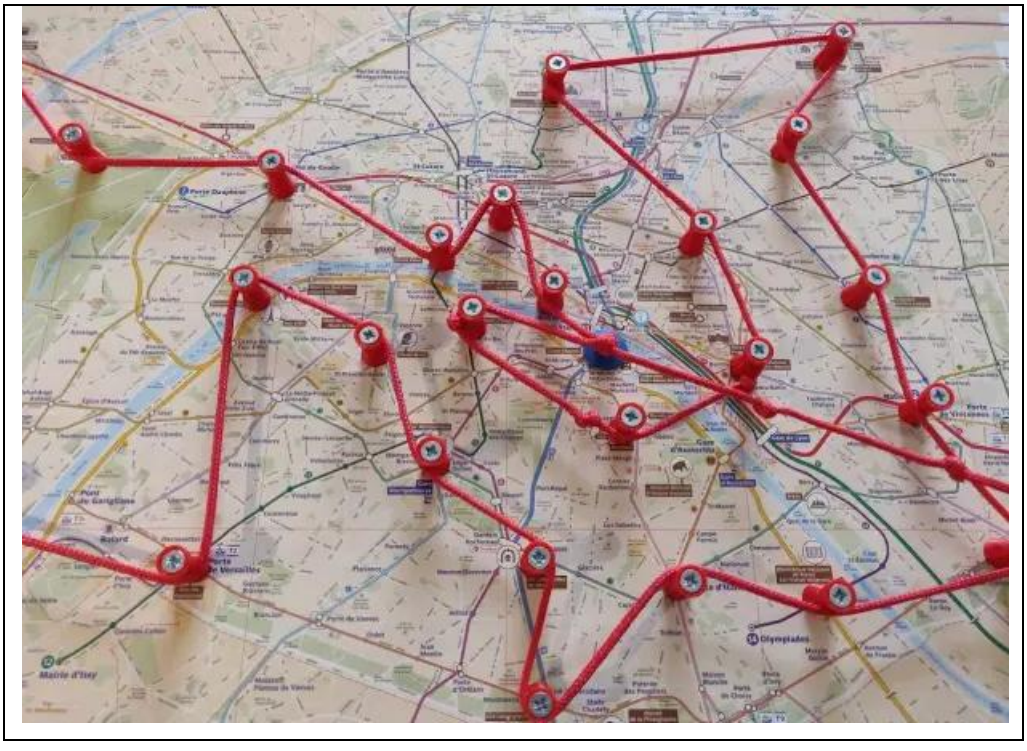
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Το πρόβλημα του ταξιδεύοντος πωλητή: Παρουσίαση και ανάλυση αλγορίθμων επίλυσης – Μελέτη περίπτωσης»



Του φοιτητή
ΧΕΙΛΙΤΗ ΣΤΑΥΡΟΥ
Αρ. Μητρώου: 164778

Επιβλέπων
ΚΑΡΑΜΗΤΟΠΟΥΛΟΣ ΛΕΩΝΙΔΑΣ
Ε.Δ.Ι.Π.

Θεσσαλονίκη, 12 Σεπτεμβρίου 2025

Το πρόβλημα του ταξιδεύοντος πωλητή: Παρουσίαση και ανάλυση αλγορίθμων επίλυσης – Μελέτη περίπτωσης

Κωδικός Δ.Ε. 23265

ΧΕΙΛΙΤΗΣ ΣΤΑΥΡΟΣ

ΚΑΡΑΜΗΤΟΠΟΥΛΟΣ ΛΕΩΝΙΔΑΣ

Ημερομηνία ανάληψης Δ.Ε. 08/10/2023

Ημερομηνία περάτωσης Δ.Ε. 12/09/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Σε όσους είναι πάντα δίπλα μου»

Πρόλογος

Η διπλωματική εργασία είναι το τελικό στάδιο για να μπορέσει ένας φοιτητής να ολοκληρώσει τις σπουδές του και να λάβει το πτυχίο του. Η εργασία η οποία μου κέντρισε το ενδιαφέρον και επέλεξα να ασχοληθώ έχει ως θέμα «Το πρόβλημα του ταξιδεύοντος πωλητή: Παρουσίαση και ανάλυση αλγορίθμων επίλυσης – Μελέτη περίπτωσης» και το παράδειγμα που θα χρησιμοποιήσω έχει σχέση με την Εταιρεία που εργάστηκα. Το πρόβλημα ορίζεται ως εξής : ένας οδηγός καλείται να επισκεφθεί μία σειρά από πόλεις όπου θα ακολουθήσει τη πιο σύντομη διαδρομή, για να μπορέσει να περάσει από κάθε πόλη ακριβώς μία φορά και να επιστρέψει στο σημείο από όπου ξεκίνησε. Σκοπός της εργασίας είναι να κινητοποιήσει τις επόμενες γενιές και να κεντρίσει το ενδιαφέρον των αναγνωστών, δίνοντάς τους κίνητρο να κάνουν περαιτέρω έρευνα. Εν κατακλείδι, θα προσπαθήσουμε να αναδείξουμε τα πλεονεκτήματα και τα μειονεκτήματα των διαφορετικών αλγορίθμων που υπάρχουν και θα εστιάσουμε σε έναν αλγόριθμο που θα ακολουθήσουμε στο παράδειγμά μας.

Περίληψη

Το πρόβλημα του περιπλανώμενου πωλητή είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης που μελετάται στην θεωρητική επιστήμη των υπολογιστών και στην έρευνα εφαρμογών. Με δεδομένη μία λίστα από πόλεις και των μεταξύ τους αποστάσεων, το ζητούμενο του προβλήματος είναι να βρεθεί η συντομότερη διαδρομή που θα περιλαμβάνει όλες τις πόλεις από μία μόνο φορά. Η παρούσα εργασία έχει ως στόχο να παρουσιάσει τα προβλήματα που θα βρεθεί αντιμέτωπος ο πλανώδιος πωλητής που αφορούν το μέγεθος και τη δομή. Έτσι, θα παρουσιαστούν διάφοροι ακριβείς και προσεγγιστικοί αλγόριθμοι που σχετίζονται με το πρόβλημα, αλλά θα δοθεί ιδιαίτερη έμφαση στον ακριβή αλγόριθμο Branch and Bound που χρησιμοποιεί τεχνικές διακλάδωσης και περιορισμού για τη σταδιακή εξερεύνηση του δέντρου λύσεων με σκοπό τη μείωση του υπολογιστικού κόστους.

Στο πλαίσιο της εργασίας πραγματοποιείται αναλυτική παρουσίαση της μεθοδολογίας Branch and Bound, ενώ στη συνέχεια γίνεται εφαρμογή με ένα μεσαίου μεγέθους παράδειγμα από μία Εταιρεία. Η επίλυση του προβλήματος βήμα προς βήμα επιτρέπει τη πλήρη κατανόηση του τρόπου λειτουργίας του αλγορίθμου και της αποδοτικότητάς του. Τέλος, θα αναφερθούν κάποια συμπεράσματα σχετικά με τη χρησιμότητα της μεθόδου σε μικρού και μεσαίου μεγέθους προβλήματα, καθώς και προτάσεις που θα τη βελτιώσουν.

«The Traveling Salesman Problem: Presentation and Analysis of Solving Algorithms – Case Study»

«STAVROS CHEILITIS»

Abstract

The traveling salesman problem is a combinatorial optimization problem studied in theoretical computer science and applied research. Given a list of cities and the distances between them, the problem is to find the shortest route that includes all the cities in a single trip. This study aims to present the problems that the street vendor will face regarding size and structure. Thus, various exact and approximate algorithms related to the problem will be presented, but special emphasis will be given to the exact Branch and Bound algorithm that uses branching and constraint techniques to gradually explore the solution tree in order to reduce the computational cost.

In the context of the work, a detailed presentation of the Branch and Bound methodology is carried out, while it is then applied with a medium-sized example from a Company. Solving the problem step by step allows for a full understanding of how the algorithm works and its efficiency. Finally, some conclusions will be reported regarding the usefulness of the method in small and medium-sized problems, as well as suggestions that will improve it.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω μέσα από τη καρδιά μου όλους όσους συνέβαλαν στην εκπλήρωση αυτής της Πτυχιακής. Ιδιαίτερα τον αρχικό επιβλέπον καθηγητή κ. Κώστογλου Βασίλη, στη συνέχεια το κ. Καραμητόπουλο Λεωνίδα για τις συμβουλές και τις αντοχές να με αντέξει, το Γιάννη και τη Μάγδα για την ώθηση που δε σταμάτησαν να μου δίνουν και στο φίλο μου το Νίκο που στα δύσκολα είναι πάντα εκεί.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Εισαγωγικά Στοιχεία

1.1 Γενικά

Το πρόβλημα του περιοδεύοντος πωλητή (Travelling Salesman Problem - TSP) αποτελεί ένα από τα πιο χαρακτηριστικά και μελετημένα προβλήματα στη συνδυαστική βελτιστοποίηση. Θεωρείται συνέχεια του προβλήματος του κύκλου Hamilton, εξίσου δύσκολου προβλήμστος, καθώς εισέρχεται η παράμετρος του κόστους για κάθε μετακίνηση και η απαίτηση να βρεθεί η διαδρομή με το μικρότερο δυνατό κόστος. Ως εκ τούτου, δεν χρειάζεται μόνο η εύρεση μιας τέτοιας διαδρομής, αλλά του συνόλου τους έτσι ώστε να πραγματοποιηθούν οι απαραίτητες συγκρίσεις και να επιλεγεί η μικρότερη - άριστη. Στη βασική του μορφή, το πρόβλημα ζητά τον προσδιορισμό της συντομότερης δυνατής διαδρομής που διατρέχει ένα σύνολο πόλεων, περνώντας από κάθε πόλη μία και μόνο φορά και επιστρέφοντας στο αρχικό σημείο εκκίνησης. Αν και η διατύπωση του προβλήματος φαίνεται απλή, η επίλυσή του είναι υπολογιστικά δύσκολη, καθώς συγκαταλέγεται στα NP-πλήρη προβλήματα.

Το πρόβλημα αρχικά διατυπώθηκε σαν ένα μαθηματικό πρόβλημα το 1930 και είναι ένα από τα πιο εντατικά μελετημένα προβλήματα βελτιστοποίησης με αποτέλεσμα να θεωρείται σημείο αναφοράς για πολλές μεθόδους βελτιστοποίησης. Παρόλη τη μαθηματική δυσκολία του προβλήματος, ένας μεγάλος αριθμός εμπειρικών αλλά και επακριβών μεθόδων είναι γνωστός και έτσι προβλήματα που περιλαμβάνουν δεκάδες χιλιάδες πόλεις μπορούν να λυθούν.

Το πρόβλημα του περιπλανώμενου πωλητή έχει πληθώρα εφαρμογών ακόμη και στην πιο αρχική διατύπωση, όπως στον σχεδιασμό, στην λογιστική και στην κατασκευή μικροτσίπ. Με λίγες αλλαγές, εμφανίζεται σαν ένα υπο-πρόβλημα πολλών τομέων όπως για παράδειγμα της γενετικής αλληλουχίας. Σε αυτές τις εφαρμογές, η έννοια της πόλης αντιστοιχεί για παράδειγμα σε πελάτες ή κομμάτια DNA και η έννοια απόσταση αντιστοιχεί σε χρόνο ταξιδιού ή κόστος ή σε ομοιότητες μεταξύ κομματιών DNA. Σε πολλές εφαρμογές, επιπλέον περιορισμοί όπως περιορισμένοι πόροι ή παράθυρα χρόνου καθιστούν το πρόβλημα ακόμη δυσκολότερο.

Στην θεωρία της υπολογιστικής πολυπλοκότητας, η εκδοχή του προβλήματος του περιπλανώμενου πωλητή ανήκει στην κλάση των *NP-problems*, δηλαδή *nondeterministic polynomial time problems*. Τα προβλήματα αυτά χαρακτηρίζονται κυρίως από την εξής ιδιότητά τους:

Παρόλο που οποιαδήποτε λύση στο πρόβλημα μπορεί να επαληθευτεί πολύ γρήγορα, δεν υπάρχει κανένας γνωστός τρόπος για να βρεθεί μία λύση. Το πιο χαρακτηριστικό γνώρισμα των NP-problems είναι ότι καμία γρήγορη λύση δεν είναι γνωστή. Άρα, ο χρόνος που απαιτείται για να λυθεί ένα πρόβλημα χρησιμοποιώντας οποιοδήποτε γνωστό αλγόριθμο αυξάνεται πολύ γρήγορα όσο το μέγεθος του προβλήματος αυξάνεται, με αποτέλεσμα ο χρόνος που χρειάζεται για να λυθούν ακόμη και μεσαία προς μεγάλα προβλήματα να είναι στο επίπεδο των εκατομμυρίων ή δισεκατομμυρίων χρόνων, χρησιμοποιώντας την υπολογιστική δύναμη που είναι διαθέσιμη σήμερα. Αυτό έχει ως συνέπεια, ότι μόνο και ο καθορισμός αν είναι δυνατόν να λυθεί το πρόβλημα ή όχι να είναι από τα κύρια άλυτα προβλήματα στην επιστήμη των ηλεκτρονικών υπολογιστών.

Μέχρι σήμερα, όλοι οι αλγόριθμοι που χρησιμοποιούνται για την επίλυση του TSP εφρμόζονται και σε πολλά άλλα πεδία εφαρμογών. Αυτοί οι αλγόριθμοι χωρίζονται κυρίως σε δύο κατηγορίες: στους ακριβείς αλγόριθμους και στους ευρετικούς αλγορίθμους. Η βασική διαφορά μεταξύ των ακριβών και των ευρετικών αλγορίθμων είναι το γεγονός ότι οι ευρετικοί αλγόριθμοι δε δίνουν τη βέλτιστη λύση

(αλλά δεν είναι και αδύνατο να καταλήξουν τελικά σε αυτή), αλλά δίνουν μια προσέγγιση που είναι αρκετά κοντά (άλλοτε λιγότερο και άλλοτε περισσότερο) στη βέλτιστη λύση.

1.2 Κίνητρο της Εργασίας

Η επιλογή του θέματος για τη παρούσα Πτυχιακή Εργασία προέκυψε από τη προσωπική μου ενασχόληση με την εφαρμογή των μαθηματικών και της πληροφορικής στην επίλυση πρακτικών προβλημάτων που συναντώνται στο πραγματικό κόσμο. Το πρόβλημα του ταξιδεύοντος πωλητή (TSP) μου τράβηξε το ενδιαφέρον, καθώς συσχετίζει την απλότητα στη διατύπωση με την υπολογιστική πολυπλοκότητα, προσφέροντας πρόσφορο έδαφος για ανάλυση και πειραματισμό με διάφορες αλγοριθμικές προσεγγίσεις. Το επιπρόσθετο ενδιαφέρον δημιουργήθηκε από μία μελέτη περίπτωσης που έκανα στην Εταιρεία που εργάζομαι : Ένας οδηγός πρέπει να μεταβεί σε 10 αποθήκες της Εταιρείας, ξεκινώντας από τα Κετρικά Γραφεία. Η αναζήτηση της βέλτιστης διαδρομής σε αυτό το πλαίσιο δεν αποτελεί ένα θεωρητικό ερώτημα, αλλά μία πραγματική ανάγκη που επηρεάζει το λειτουργικό κόστος, την αποδοτικότητα και το προγραμματισμό της επιχείρησης. Μέσα από αυτή την εργασία μου δόθηκε η δυνατότητα να εμβαθύνω σε αλγοριθμικές τεχνικές, να εφαρμόσω γνώσεις που απέκτησα κατά τη διάρκεια των σπουδών μου και να διαπιστώσω τη σημασία της τεχνολογίας στη λήψη πρακτικών αποφάσεων.

1.3 Δομή της Εργασίας

Το 1^{ο} κεφάλαιο αποτελεί το εισαγωγικό μέρος της εργασίας, όπου γίνεται μία συνοπτική αναφορά του προβλήματος , καθώς επίσης περιγράφεται το κίνητρο της εργασίας.

Στο 2^{ο} κεφάλαιο αρχικά γίνεται μία ιστορική αναδρομή και ακολουθεί μία πιο λεπτομερής περιγραφή του προβλήματος. Επιπλέον, αναφέρονται και κάποια παραδείγματα με εφαρμογές σε διάφορους τομείς της καθημερινότητας.

Το 3^{ο} κεφάλαιο είναι αφιερωμένο στην παρουσίαση των κυριότερων αλγορίθμων επίλυσης του TSP.

Όσον αφορά το 4^{ο} κεφάλαιο, εκεί παρουσιάζεται η μελέτη περίπτωσης που χρησιμοποιήθηκε Περιγράφεται το σενάριο, τα δεδομένα(αποστάσεις, σημεία) και γίνεται μοντελοποίηση του προβλήματος. Εφαρμόζονται ο επιλεγμένος αλγόριθμος για την εύρεση της συντομότερης διαδρομής και παρουσιάζονται τα αποτελέσματα, τόσο σε όρους βέλτιστης απόστασης όσο και υπολογιστικού χρόνου.

Τέλος, στο 5^{ο} κεφάλαιο παρατίθενται τα συμπεράσματα της εργασίας, αξιολογούνται οι μέθοδοι που εφαρμόστηκαν και προτείνονται κατευθύνσεις για μελλοντική έρευνα ή επέκταση της μελέτης.

Κεφάλαιο 2ο: Το Πρόβλημα του Ταξιδεύοντος/Περιοδεύοντος Πωλητή

Η δεύτερη ενότητα της παρούσας μελέτης αφιερώνεται στην παρουσίαση του ιστορικού και θεωρητικού υποβάθρου του προβλήματος του Ταξιδεύοντος Πωλητή (Traveling Salesman Problem – TSP), το οποίο αποτελεί ένα από τα κλασικότερα και πλέον μελετημένα προβλήματα στη Θεωρία Γράφων και την Υπολογιστική Πολυπλοκότητα. Η αναφορά εκκινεί με μια συνοπτική επισκόπηση της ιστορικής πορείας του προβλήματος και της εξέλιξής του στο πέρασμα των αιώνων, καθώς και της συμβολής του στη θεμελίωση κρίσιμων ερευνητικών περιοχών.

Στη συνέχεια, το TSP εξετάζεται ως πρόβλημα Θεωρίας Γράφων, αναλύοντας τις βασικές έννοιες που το περιγράφουν, ενώ ακολουθεί παρουσίαση της υπολογιστικής του πολυπλοκότητας και της κατηγοριοποίησής του ως NP-πλήρους προβλήματος. Ιδιαίτερη έμφαση δίνεται στη μαθηματική διατύπωσή του, τόσο μέσω κλασικών προσεγγίσεων βελτιστοποίησης όσο και μέσω μοντέλων γραμμικού προγραμματισμού, τα οποία επιτρέπουν την αυστηρή περιγραφή των περιορισμών και των στόχων του προβλήματος.

Ακολούθως, εξετάζονται σημαντικές παραλλαγές του προβλήματος, οι οποίες αντανακλούν διαφορετικές πρακτικές συνθήκες, όπως το ασύμμετρο TSP και το Γραφικό TSP. Η ενότητα ολοκληρώνεται με την ανάδειξη των πραγματικών εφαρμογών του TSP σε ποικίλους επιστημονικούς και τεχνολογικούς τομείς, όπως η δρομολόγηση οχημάτων, η εφοδιαστική αλυσίδα, η ηλεκτρονική σχεδίαση και η ανάλυση βιολογικών δεδομένων, υπογραμμίζοντας τη διττή φύση του προβλήματος ως θεωρητικής πρόκλησης και πρακτικού εργαλείου.

2.1 Ιστορικό Υπόβαθρο

Η προέλευση του TSP δεν έχει διατυπωθεί με ακρίβεια. Η πρώτη αναφορά συναντάται το 1832, σε ένα εγχειρίδιο για πλανόδιους πωλητές, το οποίο περιλαμβάνει μερικά παραδείγματα περιηγήσεων στη Γερμανία και στην Ελβετία. Σε αυτή τη πρώτη αναφορά, δεν εντοπίζονται μαθηματικές αποδείξεις.

Οι πρώτοι Μαθηματικοί που όρισαν το TSP ήταν ο Ιρλανδός William Rowan Hamilton και ο Βρετανός Thomas Kirkman. Το Icosian Puzzle, το οποίο δημιουργήθηκε από τον Hamilton, περιελάμβανε την εύρεση ενός κύκλου από τις άκρες ενός δωδεκαέδρου, ούτως ώστε κάθε σημείο να επισκέπτεται μόνο μία φορά και το τελικό σημείο, να είναι ίδιο με το αρχικό. Ο κύκλος αυτός πήρε το όνομα του δημιουργού του και αποτελεί ένα μονοπάτι κατέχοντας την παραπάνω ιδιότητα. Επομένως, ένα μονοπάτι του Hamilton, εμπεριέχει ένα μονοπάτι σε μη κατευθυνόμενο γράφημα, το οποίο επισκέπτεται κάθε κόμβο ακριβώς μια φορά.

Η γενική μορφή του προβλήματος του Πλανόδιου Πωλητή, φαίνεται να μελετήθηκε για πρώτη φορά την δεκαετία του 1930 στη Βιέννη και στο Πανεπιστήμιο του Χάρβαρντ, κατά βάση από τον Karl Menger, ο οποίος καθόρισε και το πρόβλημα.

Η πρώτη απόπειρα μαθηματικής προσέγγισης έγινε από τον Merrill Meeks Flood τη δεκαετία του ’50, που προσπάθησε να λύσει ένα πρόβλημα σχετικά με τα δρομολόγια που εκτελολυσαν τα σχολικά λεωφορεία. Τη περίοδο εκείνη, ο Hasslier Whitney από το Πανεπιστήμιο του Princeton, πρότεινε το όνομα Travelling Salesman Problem (TSP). Με τη πάροδο των χρόνων, το πρόβλημα γινόταν όλο και πιο δημοφιλές στους επιστημονικούς κύκλους της Ευρώπης και της Αμερικής, με αποκορύφωμα το συνέδριο RAND στην Santa Monica το 1959, όπου θα δινόταν χρηματική έπαθλο σε όποιον κατάφερε

να προσφέρει λύση στο πρόβλημα. Οι George Dantzig, Delbert R. Fulkerson και Selmer M. Johnson, συμμετέχοντας στο συνέδριο, διατύπωσαν το TSP ως ένα πρόβλημα ακέραιου γραμμικού προγραμματισμού και εφάρμοσαν τη μέθοδο αποκοπής επιπέδων(cutting plane). Πέρα από αυτή τη μέθοδο, που δεν έδωσε μία άμεση αλγοριθμική λύση στο πρόβλημα του Πλανόδιου Πωλητή, οι ιδέες τους βοήθησαν να χρησιμοποιηθεί για πρώτη φορά ο αλγόριθμος διακλάδωσης και οριοθέτησης (branch and bound).

Χάρη σε αυτές τις νέες μεθόδους που εφαρμόστηκαν, δόθηκε λύση στη περίπτωση με τις 49 πόλεις σε επίπεδο βέλτιστο, κατασκευάζοντας μία διαδρομή και αποδεικνύοντας ότι καμία άλλη δεν μπορεί να είναι συντομότερη. Τις επόμενες δεκαετίες, το πρόβλημα ανάγκασε αρκετούς ευρενητές να κάνουν μελέτες από διάφορους κλάδους όπως των Μαθηματικών, της Φυσικής, της επιστήμης των υπολογιστών, αλλά και της Χημείας. Τότε έγινε η πρώτη αναφορά στην έννοια του Ελάχιστου Επικαλυπτόμενου Δέντρου (Minimum Spanning Tree, MST), που αποτελεί ένα υποσύνολο ακμών ενός γράφου, ενώνει όλους τους κόμβους και τα σημεία μεταξύ τους και στο οποίο το συνολικό βάρος των ακμών, είναι το ελάχιστο.

Το 1972, ο Richard Karp απέδειξε ότι το πρόβλημα με το κύκλο του Hamilton ήταν ένα NP-πλήρες πρόβλημα (δηλαδή μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο). Από αυτό επιβεβαιώθηκε ότι και το TSP ανήκει στα NP-δύσκολα προβλήματα λόγω της πολυπλοκότητάς του. Αύτο δίνει μία μαθηματική εξήγηση για τη προφανή υπολογιστική δυσκολία εύρεσης βέλτιστων μονοπατιών. Στη συνέχεια, αναπτύχθηκαν νέες αλγοριθμικές τεχνικές που εφαρμόστηκαν στο πρόβλημα του Πλανόδιου Πωλητή και απέδειξαν την αποτελεσματικότητά τους. Παραδείγματα τέτοιων τεχνικών, πέρα από τον αλγόριθμο B&B είναι η μέθοδος χαλάρωσης του Lagrange, ο αλγόριθμος και η ευριστίκη συνάρτηση των Lin-Kernighan, η Προσομοιωμένη Ανόπτηση και το πεδίο των συνδυαστικών πολύεδρων για δύσκολα συνδυαστικά προβλήματα βελτιστοποίησης.

Μεγάλη ανάπτυξη υπήρξε στα τέλη των δεκαετιών του 1970 και του 1980, όταν πολλοί επιστήμονες μεταξύ των οποίων οι Grotschel, Padberg, Rinaldi, έδωσαν λύση σε περιπτώσεις με έως και 2.392 πόλεις, χρησιμοποιώντας τις μεθόδους Cutting Plane και B&B.

Στη διάρκεια της δεκαετίας του 1990, οι Applegate, Bixby, Chvatal και Cook ανέπτυξαν ένα πρόγραμμα που ονομάστηκε “Concorde TSP Solver” και χρησιμοποιείται σε πολλές πρόσφατες καταγεγραμμένες λύσεις ενώ προσφέρεται σε δωρεάν μορφή για ακαδημαϊκή χρήση. Ο Gerhard Reinelt το 1991, δημοσίευσε τη βιβλιοθήκη TSPLIB. Πρόκειται για μία συλλογή από ενδεικτικές περιπτώσεις μεταβαλλόμενης δυσκολίας που χρησιμοποιήθηκε από πολλές ερευνητικές ομάδες για σύγκριση των αποτελεσμάτων.

Το 2006 δόθηκε η λύση με τις περισσότερες πόλεις μέχρι σήμερα. Μία ομάδα συνεργατών αποτελούμενη από τους Applegate, Bixby, Chvatal & Cook, υπολόγισαν μία περίπτωση βέλτιστης διαδρομής διαμέσου 85.900 πόλεων μέσα από ένα πρόβλημα διάταξης ενός microchip. Για διάφορα άλλα προβήματα με εκατομμύρια πόλεις, οι λύσεις που μπορούν να προκύψουν, βρίσκονται μέσα στο 1% από τη βέλτιστη διαδομή.

Το 2009 έγινε ακόμη μία απόπειρα για εύρεση λύσης του προβλήματος. Ο Robert Bosch, με αφορμή τη Μόνα Λίσα του Λεονάρντο Ντα Βίντσι, έχοντας 100.000 κουκίδες σαν σημεία, εφηύρε ένα παρόμοιο πρόβλημα περιπλανώμενου εμποράκου. Σκοπός ήταν η λύση του προβλήματος να αποτελεί τη δημιουργία του πίνακα σαν μία συνεχή γραμμή. Η βέλτιστη λύση για αυτό το πρόβλημα παραμένει μυστήριο για δυνατούς λύτες και εφόσον βρεθεί, θα αποτελεί παγκόσμιο ρεκόρ λύσης προβλήματος περιπλανώμενου εμποράκου.

2.2 Το TSP ως πρόβλημα Θεωρίας Γράφων

Το πρόβλημα του περιοδεύοντος πωλητή μπορεί να προσεγγιστεί αρχικά μέσω της θεωρίας γράφων, καθώς αφορά τη βέλτιστη διαδρομή σε ένα δίκτυο σημείων(πόλεων) συνδεδεμένων μεταξύ τους. Συγκεκριμένα, το TSP διατυπώνεται σε πλήρη γράφο G = (V,E), όπου:

- Το σύνολο κορυφών V = {1,2,…, n} αναπαριστά τις πόλεις που πρέπει να επισκεφθεί ο πωλητής.
- Το σύνολο ακμών E περιέχει όλες τις δυνατές διαδρομές μεταξύ ζευγών πόλεων.
- Κάθε ακμή (i,j) ∈ E φέρει ένα θετικό βάρος $c_{ij,}$ το οποίο εκφράζει το κόστος, την απόσταση ή το χρόνο μετάβασης από τη πόλη i στη πόλη j.
- Αν δεν υπάρχει ακμή που να συνδέει δύο κόμβους, τότε το κόστος θεωρείται άπειρο ($c_{ij} = \infty$).

Αντικείμενο του προβλήματος είναι ο προσδιορισμός ενός κυκλικού μονοπατιού (Hamiltonian cycle) που περνά ακριβώς μία φορά από κάθε κόμβο και επιστρέφει στο αρχικό σημείο εκκίνησης, με τέτοιο τρόπο ώστε να ελαχιστοποιείται το συνολικό κόστος της διαδρομής[1].

Δομικά χαρακτηριστικά του γράφου:

- Πληρότητα: Ο γράφος θεωρείται πλήρης, δηλαδή κάθε πόλη συνδέεται απευθείας με κάθε άλλη μέσω μιας ακμής. Αυτό ισχύει επειδή υποθέτουμε ότι ο πωλητής μπορεί να μετακινηθεί μεταξύ οποιωνδήποτε δύο πόλεων.
- Κατευθυνώμενος ή μη κατευθυνόμενος: Αν το κόστος c_{ij} δεν είναι ίσο με το c_{ji}, (δηλαδή αν η διαδρομή από την πόλη i στην πόλη j έχει διαφορετικό κόστος από την αντίστροφη), τότε ο γράφος είναι κατευθυνόμενος (directed TSP). Αν το κόστος είναι συμμετρικό, τότε ο γράφος είναι μη κατευθυνόμενος (symmetric TSP).
- Συνεκτικότητα: Ο γράφος είναι συνεκτικός, δηλαδή υπάρχει διαδρομή μεταξύ κάθε ζεύγους κόμβων.

Εξαιρερικό ενδιαφέρον βρίσκουμε όταν παρουσιάζονται Ευκλείδειες αποστάσεις μεταξύ των κόμβων ενός συμμετρικού TSP((Symmetric TSP – sTSP). Σε αυτές τις περιπτώσεις, οι κόμβοι που καθορίζουν το πρόβλημα αντιστοιχούν στα σημεία του επιπέδου δύο διαστάσεων και η απόσταση μεταξύ 2 κόμβων είναι η Ευκλείδεια απόσταση μεταξύ των αντίστοιχων σημείων. Επίσης, όταν συναντάμαι αυτή τη περίπτωση πρέπει να τηρείται η τριγωνική ανισότητα, δηλαδή:

$$c_{ij} + c_{jg} \geq c_{ig} \text{ για κάθε i,g,j.}$$

Η ευκλείδεια απόσταση ορίζεται ως d = $\sqrt{(\chi 2 - \chi 1)^2 + (y2 - y1)^2}$, όπου (χ1, y1) & (χ2, y2) οι συντεταγμένες δύο πόλεων. Η συγκεκριμένη τακτική, έχει το πλεονέκτημα ότι σε ένα σύνδεσμο από πόλεις μαζί με τις χωρικές συντεταγμένες τους, παρέχει ένα σύνολο από συνδέσμους κόμβων, οι οποίοι μας επιτρέπουν να έχουμε πάντοτε ένα πλήρες συνδεδεμένο γράφημα.

Σχέση με άλλα προβλήματα θεωρίας γράφων:

Το TSP σχετίζεται στενά με το Πρόβλημα του Hamiltonian κύκλου**,** δηλαδή τον εντοπισμό κυκλικής διαδρομής που περνά από κάθε κόμβο ακριβώς μία φορά. Το TSP διαφοροποιείται στο ότι αναζητά τον βέλτιστο (ελάχιστου κόστους) Hamiltonian κύκλο**,** καθιστώντας το πρόβλημα όχι μόνο συνδυαστικό, αλλά και βέλτιστης απόφασης.

2.3 Υπολογιστική πολυπλοκότητα και NP-πληρότητα

Αν και η διατύπωσή του είναι απλή, το TSP είναι ένα από τα πιο μελετημένα προβλήματα στον χώρο της θεωρίας υπολογισμού και του πεδίου της υπολογιστικής πολυπλοκότητας. Επιπλέον, ανήκει στην κατηγορία των NP-πλήρων προβλημάτων [2]. Αυτό σημαίνει ότι για γενική περίπτωση και μεγάλα n, δεν υφίσταται γνωστός αλγόριθμος επίλυσης σε πολυωνυμικό χρόνο που να εγγυάται εύρεση της βέλτιστης λύσης.

Για ένα σύνολο n πόλεων, ο συνολικός αριθμός δυνατών διαδρομών που επισκέπτονται κάθε πόλη ακριβώς μία φορά και επιστρέφουν στο σημείο εκκίνησης (δηλαδή Hamiltonian κύκλοι), είναι:

- Για κατευθυνόμενο γράφο (Directed TSP) : (n−1)!
- Για μη κατευθυνόμενο γράφο (Symmetric TSP) : $\frac{(n-1)!}{2}$

Αυτό σημαίνει ότι η υπολογιστική προσπάθεια για εύρεση της βέλτιστης διαδρομής αυξάνεται εκθετικά με τον αριθμό των πόλεων. Για παράδειγμα, για n = 10 υπάρχουν 181,440 πιθανές διαδρομές (στο συμμετρικό TSP), ενώ για n = 20, ξεπερνούν τα 10 δισεκατομμύρια.

Πολυπλοκότητα και εκθετική αύξηση υποψήφιων λύσεων:

Η αποφασιστική μορφή του προβλήματος (decision version) του TSP διατυπώνεται ως εξής:

Δοθέντος ενός συνόλου πόλεων και ενός θετικού ακέραιου k, υπάρχει κυκλική διαδρομή που επισκέπτεται κάθε πόλη μία μόνο φορά, με συνολικό κόστος μικρότερο ή ίσο του k.

Αυτή η μορφή του προβλήματος είναι NP-πλήρης (NP-complete), δηλαδή:

- Ανήκει στην κλάση NP (οι λύσεις του μπορούν να επαληθευτούν σε πολυωνυμικό χρόνο),
- Είναι εξίσου δύσκολο με οποιοδήποτε άλλο πρόβλημα στην NP (κάθε πρόβλημα στην NP μπορεί να αναχθεί σε αυτό σε πολυωνυμικό χρόνο).

Η ταξινόμηση αυτή αποδείχθηκε από τον Richard M. Karp το 1972, στο πλαίσιο της διάσημης λίστας των 21 NP-πλήρων προβλημάτων.

Αντίθετα, η βελτιστοποιητική μορφή του TSP (εύρεση της βέλτιστης διαδρομής) θεωρείται NP-δύσκολη (NP-hard), καθώς είναι τουλάχιστον τόσο δύσκολη όσο οποιοδήποτε πρόβλημα στην NP, αλλά δεν είναι απαραίτητα στην ίδια την κλάση NP (δεν μπορεί να επαληθευτεί εύκολα αν η λύση είναι η καλύτερη δυνατή).

2.4 Μαθηματική διατύπωση του προβλήματος

Η μαθηματική μοντελοποίηση του TSP επιτρέπει την ακριβή του περιγραφή και την επίλυσή του μέσω μεθόδων ακέραιου γραμμικού προγραμματισμού.[3]

- Έστω σύνολο πόλεων V = {1,2,…, n}.
- Έστω συνάρτηση κόστους c : V × V → R+, με c_{ij} να δηλώνει το κόστος μετακίνησης από τη πόλη i στη πόλη j.
- Ζητείται κυκλική διαδρομή που ελαχιστοποιεί το άθροισμα των μεταβάσεων και καλύπτει όλα τα στοιχεία του V μία φορά.

Μεταβλητές απόφασης:

$X_{i,j} = \begin{cases} 1, \text{αν η διαδρομή περιλαμβάνει μετάβαση από τη πόλη i στη πόλη j} \\ 0, \text{διαφορετικά} \end{cases}$ $\forall$ i,j $\in$ V, i $\neq$ j

Αντικειμενική συνάρτηση : $\min \sum \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} c(i,j) * x(i,j)$

Περιορισμοί :

- Κάθε πόλη να εξέρχεται μία φορά : $\sum_{j=1, j \neq i}^{n} x(i,j) = 1 \quad \forall$ i $\in$ V
- Κάθε πόλη να εισέρχεται μία φορά : $\sum_{i=1, i \neq j}^{n} x(i,j) = 1 \quad \forall$ j $\in$ V

Αποφυγή υποκύκλων (Miller–Tucker–Zemlin υποκύκλικοί περιορισμοί):

Εισάγονται βοηθητικές μεταβλητές $u_i \in Z$, i=2,...n , με τους παρακάτω περιορισμούς:

$u_i - u_j + n * x_{ij} \leq n - 1 \quad \forall$ i $\neq$ j, i,j $\in \{2,\ldots, n\}$

$1 \leq u_i \leq n-1 \; \forall$ i $\in \{2,\ldots, n\}$

2.5 Παραλλαγές του TSP

Το κλασικό πρόβλημα του Ταξιδεύοντος Πωλητή (TSP), όπως περιγράφηκε στην αρχική του μορφή, υποθέτει έναν πωλητή που επιθυμεί να επισκεφθεί ένα σύνολο πόλεων ακριβώς μία φορά και να επιστρέψει στο σημείο εκκίνησης, με σκοπό την ελαχιστοποίηση της συνολικής διαδρομής. Ωστόσο, σε πλήθος ρεαλιστικών εφαρμογών, οι απαιτήσεις και οι περιορισμοί του προβλήματος διαφοροποιούνται σημαντικά, οδηγώντας στην ανάπτυξη πολλών παραλλαγών του βασικού προβλήματος. Οι κυριότερες παραλλαγές του προβλήματος ενσωματώνουν χρονικούς περιορισμούς, πολλαπλούς πωλητές, αβεβαιότητα ή ειδικές γεωμετρικές συνθήκες.

2.5.1 Το TSP σε γενικά γραφήματα

Η γενική μορφή του προβλήματος του Ταξιδεύοντος Πωλητή (TSP) ορίζεται πάνω σε έναν πλήρη γράφο G = (V,E) όπου:

- V είναι το σύνολο των κόμβων (ή πόλεων),
- E το σύνολο των ακμών, που αντιπροσωπεύουν τις δυνατές μεταβάσεις μεταξύ κόμβων, και
- κάθε ακμή (i,j) $\in$ E συνοδεύεται από ένα μη αρνητικό βάρος c_{ij}, που αναπαριστά απόσταση, χρόνο ή κόστος.

Υπάρχουν περιπτώσεις που συναντάμε γραφήματα που δεν είναι πλήρη και επιθυμούμε να υπολογίσουμε τους μικρότερους κύκλους Hamilton. Ανάλογα με τη περίπτωση, έχουμε τη δυνατότητα να διαχειριστούμε αυτές τις καταστάσεις με δύο τρόπους.

Αν απαιτείται να επισκεφτούμε κάθε κόμβο μόνο μία φορά και χρησιμοποιήσουμε μόνο τις διαθέσιμες ακμές, τότε ακολουθούμε τα εξής βήματα : Προσθέτουμε όσες ακμές λείπουν, δίνοντάς τες έναν αρκετά μεγάλο αριθμό M ως βάρος και εφαρμόζουμε έναν αλγόριθμο πλήρων Γραφημάτων. Εφόσον ο αλγόριθμος τερματίσει με μια βέλτιστη λύση που δεν περιέχει καμία από τις ακμές με βάρος M, τότε η λύση που προκύπτει είναι και η βέλτιστη. Αν στη βέλτιστη λύση συναντήσουμε μία ακμή με βάρος M, τότε το αρχικό γράφημα δεν περιέχει ένα κύκλο Hamilton. Από τους διαθέσιμους αλγορίθμους που υπάρχουν και μπορούν να εγγυηθούν την εύρεση ενός κύκλου Hamilton στο

γράφημα G, έστω και αν αυτός υπάρχει, μόνο οι ακριβείς μπορούν να το επιτύχουν, εν αντιθέσει με τις ευρετικές συναρτήσεις.

Ο δεύτερος τρόπος για να διαχειριστούμε τέτοιου είδους προβλήματα αφορά την επίσκεψη στους κόμβους περισσότερες από μία φορές και το πέρασμα από τις ακμές επίσης περισσότερες από μία φορές. Αν το γράφημα που μας δίνεται είναιι πλήρες, τότε είναι πάντοτε εφικτό να βρεθεί μία διαδρομή μετ επιστροφής, υπό αυτή τη χαλάρωση. Αυτό αποκαλείται Γραφικό TSP και θα αναλυθεί στη συνέχεια.

2.5.2 Το Γραφικό TSP (Graph Traveling Salesman Problem GTSP)

Το Γραφικό TSP αποτελεί μια παραλλαγή του προβλήματος του Ταξιδεύοντος Πωλητή, όπου οι αποστάσεις μεταξύ των πόλεων (κόμβων) προέρχονται από τη δομή ενός απλού, μη πλήρους γράφου G = (V,E) αντί να δίνονται απευθείας μέσω ενός πλήρους πίνακας αποστάσεων. Περιλαμβάνει την εύρεση ενός μονοπατιού για τον πωλητή, όπου θα επισκέπτεται κάθε πόλη, με την προϋπόθεση ότι θα διανύσει την μικρότερη δυνατή απόσταση.

Στον αρχικό γράφο G, κάθε ακμή (i,j) ∈ E μπορεί να έχει μοναδιαίο ή γενικά θετικό βάρος, και δεν θεωρείται δεδομένο ότι κάθε κόμβος συνδέεται άμεσα με όλους τους υπόλοιπους. Παρ' όλα αυτά, προκειμένου να εφαρμόσουμε το πρόβλημα TSP, το οποίο απαιτεί μετάβαση από κάθε πόλη σε κάθε άλλη, κατασκευάζεται ένας πλήρης γράφος G′ = (V,E′), στον οποίο:

- Οι κορυφές παραμένουν ίδιες (V),
- Οι ακμές E′ περιλαμβάνουν όλα τα δυνατά ζεύγη (i,j), και
- Το βάρος κάθε ακμής c_{ij} ισούται με το μήκος της συντομότερης διαδρομής μεταξύ των κόμβων i και j στον αρχικό γράφο G = $c_{i\,j}$ = shortest_path(i,j) στον G

Με αυτόν τον τρόπο, το πρόβλημα διατυπώνεται τελικά ως TSP σε πλήρη γράφο, στον οποίο όμως τα κόστη δεν προκύπτουν αυθαίρετα, αλλά βασίζονται σε μια υποκείμενη δομή δικτύου (όπως οδικό ή επικοινωνιακό δίκτυο).

Το Graph TSP βρίσκει εφαρμογές σε περιβάλλοντα όπου η φυσική διασύνδεση μεταξύ των σημείων είναι περιορισμένη, όπως :

- Υποδομές πόλεων και κυκλοφοριακά δίκτυα,
- Τοπολογίες τηλεπικοινωνιακών ή υπολογιστικών συστημάτων,
- Ρομποτική πλοήγηση σε αποθηκευτικούς ή εργοστασιακούς χώρους.

Η ιδιότητα της τριγωνικής ανισότητας ικανοποιείται πάντα στον πλήρη γράφο G′, αφού τα βάρη του βασίζονται σε συντομότερες διαδρομές. Αυτό επιτρέπει τη χρήση αλγορίθμων προσέγγισης, όπως ο αλγόριθμος του Christofides, με εγγυημένα όρια απόκλισης από τη βέλτιστη λύση.

2.5.3 Το Ασύμμετρο TSP (Asymmetric Traveling Salesman Problem - aTSP)

Αυτή η εκδοχή του προβλήματος του ταξιδεύοντος πωλητή περιλαμβάνει περιπτώσεις όπου το κόστος μετάβασης από έναν κόμβο σε έναν άλλο δεν είναι απαραίτητα ίσο με το αντίστροφο. Δηλαδή, το γράφημα στο οποίο ορίζεται το πρόβλημα είναι κατευθυνόμενο (directed graph) και το κόστος ικανοποιεί γενικά :

$$c_{i,j} \neq c_{j,i}, \text{ για ορισμένα } i,j \in V$$

Το Ασύμμετρο TSP εμφανίζεται σε πρακτικά προβλήματα όπου η μετακίνηση ή η επικοινωνία έχει κατεύθυνση και η τιμή της εξαρτάται από την πορεία. Ενδεικτικά κάποια παραδείγματα :

- Μονόδρομοι ή δίκτυα με κυκλοφοριακούς περιορισμούς,
- Υψομετρικές διαφορές, όπου η κατανάλωση ενέργειας για ανάβαση διαφέρει από αυτή για κατάβαση,
- Δίκτυα επικοινωνιών ή μεταφορών, με διαφορετικά κόστη ανάλογα με τη ροή ή τον φόρτο.

Η μαθηματική διατύπωση είναι παρόμοια με αυτή του συμμετρικού TSP, με τη διαφορά ότι ο πίνακας κόστους C = (c_{ij}) δεν είναι συμμετρικός. Το πρόβλημα εξακολουθεί να απαιτεί την εύρεση ενός ελαχίστου κατευθυνόμενου Hamiltonian κύκλου, δηλαδή μιας διατεταγμένης ακολουθίας κόμβων που επισκέπτεται κάθε κόμβο ακριβώς μία φορά, ακολουθώντας κατευθυνόμενες ακμές και επιστρέφει στην αφετηρία.

Ο τρόπος επίλυσης του aTSP είναι υπολογιστικά ισοδύναμος με το κλασικό TSP. Ωστόσο, πολλές τεχνικές για συμμετρικά προβλήματα δεν εφαρμόζονται άμεσα στην ασύμμετρη περίπτωση λόγω της έλλειψης ιδιοτήτων όπως η τριγωνική ανισότητα ή η αντιστροφή διαδρομής.

Σε εφαρμογές πραγματικού κόσμου, το aTSP προσφέρει πιο ρεαλιστικά μοντέλα και για τον λόγο αυτό αποτελεί αντικείμενο σημαντικής ερευνητικής δραστηριότητας.

2.5.4 Το Πολλαπλό TSP (Multiple Traveling Salesman Problem - mTSP)

Η συγκεκριμένη παραλλαγή του προβλήματος αποτελεί γενίκευση του κλασικού TSP, όπου αντί για έναν μόνο πωλητή, πολλοί πωλητές (ή οχήματα) ξεκινούν από ένα ή περισσότερα κοινά σημεία και πρέπει να επισκεφθούν από κοινού όλα τα σημεία εξυπηρέτησης.

Ο στόχος είναι να προσδιοριστούν k διαδρομές, μία για κάθε πωλητή, ώστε:

- Κάθε κόμβος (πελάτης) να επισκεφθεί ακριβώς από έναν πωλητή,
- Κάθε πωλητής να ξεκινά και να τερματίζει (συνήθως) στο ίδιο σημείο,
- Το συνολικό κόστος (ή χρόνος) όλων των διαδρομών να ελαχιστοποιηθεί.

Το πολλαπλό TSP μπορεί να εφαρμοστεί σε δρομολόγηση στόλου οχημάτων (vehicle routing), διανομή εμπορευμάτων με πολλούς οδηγούς και ανάθεση αποστολών σε πολλούς ρομποτικούς πράκτορες. Η πολυπλοκότητα του mTSP αυξάνεται ταχύτατα με τον αριθμό πωλητών και σημείων. Συνήθως επιλύεται με ευρετικούς ή μεταευρετικούς αλγορίθμους, όπως είναι οι Γενετικοί Αλγόριθμοι(Genetic Algorithms)[4].

2.5.5 Το Πρόβλημα του Ελάχιστου Μονοπατιού Hamilton(Minimum Hamiltonian Path Problem)

Το Πρόβλημα του Ελάχιστου Μονοπατιού Hamilton αποτελεί παραλλαγή του TSP κατά την οποία δεν ζητείται το μονοπάτι να είναι κυκλικό (δηλαδή να επιστρέφει στο αρχικό σημείο), αλλά ένα απλό μονοπάτι που επισκέπτεται κάθε κόμβο ακριβώς μία φορά, χωρίς απαίτηση επιστροφής στο αρχικό σημείο.

Έστω ένας γράφος G = (V,E), με σύνολο κορυφών V, ακμών E και κόστος c_{ij} σε κάθε ακμή (i,j). Το πρόβλημα συνίσταται στην εύρεση μιας ακολουθίας κορυφών : $u_1,u_2,\ldots,u_n$ τέτοιας ώστε:

- κάθε κόμβος να εμφανίζεται ακριβώς μία φορά,
- η διαδρομή να είναι ανοικτή (δηλαδή δεν επιστρέφει στο σημείο εκκίνησης),
- το συνολικό κόστος να ελαχιστοποιείται :
- $\min\sum_{i=1}^{n-1} cuicui + 1$

Σε σχέση με το κλασικό TSP, το Πρόβλημα του Ελάχιστου Μονοπατιού Hamilton είναι δυνατό να οριστεί με καθορισμένους αρχική και τελική κορυφή(π.χ. από s προς t) ή χωρίς περιορισμούς.

Σε πολλές περιπτώσεις, το πρόβλημα του μονοπατιού μπορεί να μετατραπεί σε ισοδύναμο πρόβλημα TSP προσθέτοντας μία τεχνητή κορυφή με μηδενικό κόστος προς όλες τις άλλες. Αυτή η παραλλαγή του προβλήματος μπορεί να εφαρμοστεί σε περιπτώσεις όπου δεν απαιτείται κυκλική διαδρομή, όπως:

- Δρομολόγηση οχημάτων με διαφορετικό σημείο εκκίνησης και τερματισμού,
- Αποστολή ρομπότ ή drones από βάση σε στόχο μέσω ενδιάμεσων σημείων,
- Βελτιστοποίηση σειριακής επεξεργασίας (π.χ. σε παραγωγικές γραμμές ή βιοπληροφορική).

Για την επίλυσή του σε γενικούς Γράφους απαιτείται εκθετικός χρόνος, ενώ μπορεί να λυθεί και με διάφορους προσεγγιστικούς και ευρετικούς αλγορίθμους.

2.6 Πραγματικές εφαρμογές του TSP

Το πρόβλημα του ταξιδεύοντος πωλητή (TSP), παρά τον αφηρημένο μαθηματικό του ορισμό, βρίσκει ευρεία εφαρμογή σε ποικίλους τομείς της επιστήμης, της βιομηχανίας και της τεχνολογίας. Η βασική του δομή, δηλαδή η εύρεση της βέλτιστης διαδρομής για την επίσκεψη ενός συνόλου σημείων ακριβώς μία φορά, εμφανίζεται σε πληθώρα πραγματικών καταστάσεων, από τη φυσική δρομολόγηση οχημάτων έως τη μοριακή βιολογία.

Οι παρακάτω υποενότητες παρουσιάζουν χαρακτηριστικές περιπτώσεις εφαρμογών του TSP.

2.6.1 Δρομολόγηση Οχημάτων και Παραδόσεις (Vehicle Routing & Logistics)

Μία από τις πιο άμεσες εφαρμογές του TSP αφορά τον σχεδιασμό αποδοτικών διαδρομών για οχήματα μεταφορών ή διανομών. Εταιρείες logistics, ταχυμεταφορών και μεταφορών επιδιώκουν καθημερινά να να καθορίσουν αποδοτικά δρομολόγια για την εξυπηρέτηση πελατών ή την παράδοση προϊόντων. Ο στόχος είναι η ελαχιστοποίηση του λειτουργικού κόστους, του χρόνου διανομής και της κατανάλωσης καυσίμων. Εφαρμόζοντας αλγορίθμους TSP ή παραλλαγές όπως το Vehicle Routing Problem(VRP) οι οργανισμοί αυτοί επιτυγχάνουν αυξημένη αποδοτικότητα, καλύτερη εξυπηρέτηση πελατών και μικρότερο περιβαλλοντικό αποτύπωμα[5].

Παράδειγμα Εταιρειών που υλοποιούν εξελιγμένους αλγορίθμους για τη βελτιστοποίηση των διαδρομών διανομής τους:

UPS: ανέπτυξε το σύστημα ORION (On-Road Integrated Optimization and Navigation), το οποίο εφαρμόζει αλγορίθμους τύπου TSP και VRP σε πραγματικό χρόνο για να μειώσει την απόσταση και τα κόστη παράδοσης. Το ORION εξοικονομεί περίπου 100 εκατομμύρια μίλια διαδρομών ετησίως, μειώνοντας το κόστος καυσίμων κατά εκατομμύρια δολάρια και περιορίζοντας τις εκπομπές CO_2.

Η Amazon χρησιμοποιεί ρομπότ για να μεταφέρουν προϊόντα μέσα στις αποθήκες της. Οι διαδρομές που ακολουθούν τα ρομπότ για την παραλαβή και τοποθέτηση αντικειμένων είναι βελτιστοποιημένες μέσω προβλημάτων τύπου TSP, λαμβάνοντας υπόψη δυναμικούς περιορισμούς και εμπόδια.

2.6.2 Προγραμματισμός Παραγωγής (Manufacturing & Scheduling)

Στη βιομηχανική παραγωγή, το TSP εφαρμόζεται στον προγραμματισμό εργασιών πάνω σε γραμμές παραγωγής ή CNC(Computer Numerical control) μηχανές, όπου πρέπει να καθοριστεί η σειρά

επεξεργασίας πολλών τεμαχίων ή σταθμών εργασίας με ελαχιστοποίηση χρόνου ή κόστους αλλαγών εργαλείων.

Παρόμοια προβλήματα εμφανίζονται στην κατασκευή κυκλωμάτων, στον βέλτιστο προγραμματισμό βημάτων συναρμολόγησης, και στη διαχείριση αποθηκών.

2.6.3 Βιοπληροφορική και Γονιδιωματική

Το TSP βρίσκει επίσης εφαρμογή στον τομέα της βιοπληροφορικής, ιδίως σε προβλήματα που σχετίζονται με την ευθυγράμμιση και ανάλυση αλληλουχιών DNA. Για παράδειγμα, η ανακατασκευή γονιδιώματος από θραύσματα (genome assembly) μπορεί να διατυπωθεί ως ένα παραλλαγμένο TSP, όπου οι ακολουθίες DNA αντιστοιχούν σε πόλεις, και η "απόσταση" μετράται με βάση την επικαλυπτόμενη ομοιότητα. Ομοίως, χρησιμοποιείται σε προβλήματα εύρεσης βέλτιστης σειράς πειραμάτων ή ανάλυσης βιολογικών δεδομένων με τη μορφή ταξινομήσεων ή ομαδοποιήσεων. Η αποτελεσματική επίλυση τέτοιων προβλημάτων οδηγεί σε ταχύτερες και ακριβέστερες βιοϊατρικές αναλύσεις.

2.6.4 Ρομποτική και Τεχνητή Νοημοσύνη

Το πρόβλημα του ταξιδεύοντος πωλητή (TSP) βρίσκει ευρεία εφαρμογή στη ρομποτική και στην τεχνητή νοημοσύνη, κυρίως στο πλαίσιο προβλημάτων πλοήγησης, βελτιστοποίησης διαδρομών και κάλυψης περιοχών. Τα ρομπότ που κινούνται σε φυσικά ή εικονικά περιβάλλοντα συχνά καλούνται να επισκεφθούν μια σειρά από σημεία ενδιαφέροντος (targets, waypoints, περιοχές επιθεώρησης) με όσο το δυνατόν αποδοτικότερο τρόπο, τόσο από άποψη απόστασης όσο και χρόνου ή ενεργειακής κατανάλωσης.

Προβλήματα Πλοήγησης και Επιθεώρησης:

Ένα από τα πιο χαρακτηριστικά παραδείγματα αφορά τη σχεδίαση διαδρομών επιθεώρησης για ρομποτικά οχήματα, επίγεια ή εναέρια, τα οποία πρέπει να επισκεφθούν καθορισμένα σημεία για παρακολούθηση, χαρτογράφηση, λήψη δεδομένων ή συντήρηση εξοπλισμού. Αυτό μπορεί να περιλαμβάνει είτε Drones που επιθεωρούν φωτοβολταϊκά πάνελ, ανεμογεννήτριες ή γραμμές μεταφοράς, είτε Ρομπότ σε βιομηχανικά περιβάλλοντα που ελέγχουν αισθητήρες ή εξοπλισμό. Η βασική απαίτηση σε τέτοιες περιπτώσεις είναι η κάλυψη όλων των απαιτούμενων σημείων με την ελάχιστη δυνατή διαδρομή, χωρίς απαραίτητα επιστροφή στο σημείο εκκίνησης, κάτι που καθιστά το πρόβλημα ισοδύναμο με το Hamiltonian Path ή το TSP χωρίς κύκλο.

Προβλήματα Κάλυψης (Coverage Planning):

Στην περίπτωση που ένα ρομπότ πρέπει να καλύψει πλήρως έναν χώρο (π.χ. για καθαρισμό ή απολύμανση), το πρόβλημα διατυπώνεται συχνά ως TSP σε διακριτοποιημένα σημεία (π.χ. κεντρικά σημεία κελιών). Αν και η πλήρης κάλυψη ανήκει σε άλλη κατηγορία βελτιστοποίησης, το TSP χρησιμοποιείται για τη βελτιστοποίηση της σειράς κάλυψης.

Συνεργατική Ρομποτική (Multi-agent systems):

Σε συστήματα πολλαπλών ρομπότ, συχνά προκύπτουν πολλαπλά TSPs (mTSP), όπου το σύνολο των σημείων διαμοιράζεται μεταξύ των ρομπότ. Το ζητούμενο μπορεί να είναι είτε:

- η ελαχιστοποίηση του συνολικού κόστους όλων των διαδρομών, ή
- η εξισορρόπηση των επιμέρους διαδρομών ώστε να αποφεύγονται ανισοκατανομές φόρτου.

Η διατύπωση αυτή χρησιμοποιείται σε ρομποτικές αποστολές επιτήρησης, συγχρονισμένα drones σε γεωργία ακριβείας και για συνεργατική εξερεύνηση σε άγνωστα περιβάλλοντα.

Αλγόριθμοι Τεχνητής Νοημοσύνης για την Επίλυση του TSP:

Η ρομποτική αξιοποιεί εντατικά αλγορίθμους εμπνευσμένους από τη φύση και τη μάθηση για την επίλυση του TSP:

- Αλγόριθμοι αποικιών μυρμηγκιών (Ant Colony Optimization): χρησιμοποιούνται ευρέως για δυναμικά περιβάλλοντα και προσαρμοστικές διαδρομές.
- Γενετικοί αλγόριθμοι: κατάλληλοι για την εξεύρεση λύσεων κοντά στο βέλτιστο σε προβλήματα με μεγάλο αριθμό σημείων.
- Εξελικτικοί αλγόριθμοι και reinforcement learning (ενισχυτική μάθηση): εφαρμόζονται σε καταστάσεις όπου το ρομπότ "μαθαίνει" σταδιακά το περιβάλλον και βελτιώνει τις διαδρομές του.

Αυτοί οι αλγόριθμοι ενσωματώνονται σε πραγματικό χρόνο σε συστήματα πλοήγησης και σχεδιασμού αποστολών, καθιστώντας το TSP πυρήνα των υποκείμενων υπολογισμών.

2.6.5 Τηλεπικοινωνίες και Σχεδίαση Δικτύων

Το πρόβλημα του ταξιδεύοντος πωλητή (TSP) βρίσκει εφαρμογή και στον χώρο των τηλεπικοινωνιών, της σχεδίασης υποδομών δικτύου και της βελτιστοποίησης ροών δεδομένων. Πολλές από τις απαιτήσεις που τίθενται κατά την ανάπτυξη, λειτουργία και συντήρηση σύγχρονων δικτύων μπορούν να μοντελοποιηθούν ως παραλλαγές του TSP, με στόχο τη βελτιστοποίηση της απόδοσης, του κόστους και της αξιοπιστίας.

Σχεδίαση Δικτύων Οπτικών Ινών και Καλωδιώσεων:

Κατά τη φάση σχεδίασης ενός φυσικού τηλεπικοινωνιακού δικτύου — για παράδειγμα, ενός δικτύου οπτικών ινών — είναι κρίσιμο να καθοριστεί η βέλτιστη διαδρομή που πρέπει να ακολουθήσει η εγκατάσταση του καλωδίου ώστε να συνδεθούν όλοι οι κόμβοι (π.χ. υποσταθμοί, κέντρα διανομής, τερματικοί χρήστες), ελαχιστοποιώντας το συνολικό μήκος του καλωδίου ή το κόστος εγκατάστασης.

Η διατύπωση αυτή είναι ιδιαίτερα συγγενής με το TSP σε γεωμετρικό γράφο, ενώ σε πιο σύνθετες περιπτώσεις μπορεί να εξελιχθεί σε προβλήματα όπως το m-TSP (με πολλαπλούς τεχνικούς ή συνδέσεις).

Δρομολόγηση Πακέτων σε Δίκτυα Δεδομένων:

Σε δίκτυα δεδομένων (όπως τα IP δίκτυα), η ανάγκη αποδοτικής δρομολόγησης πακέτων μεταξύ κόμβων συνιστά μια δυναμική μορφή του TSP. Ειδικά σε περιπτώσεις αποστολής δεδομένων σε πολλούς παραλήπτες (multicast) ή προγραμματισμένων αποστολών (scheduled transmissions), η εύρεση της βέλτιστης ακολουθίας αποστολής ή της βέλτιστης σειράς επεξεργασίας κόμβων αποτελεί παραλλαγή του TSP. Αντίστοιχα, σε ασύρματα δίκτυα αισθητήρων, χρησιμοποιούνται TSP-παραλλαγές για την εύρεση διαδρομών που : ελαχιστοποιούν την ενεργειακή κατανάλωση, περιορίζουν τις καθυστερήσεις και διασφαλίζουν την κάλυψη και επικοινωνία με όλους τους κόμβους.

Εφαρμογές σε Κινητά Δίκτυα και Πλοήγηση Δορυφόρων/Drones:

Σε περιβάλλοντα με κινητούς κόμβους (π.χ. δορυφόρους, Μη επανδρωμένα αεροσκάφοι, αισθητήρες σε κίνηση), το TSP εφαρμόζεται για την επιλογή βέλτιστων σημείων επικοινωνίας σε δεδομένα

χρονικά διαστήματα. Για παράδειγμα, ένα δορυφορικό σύστημα πρέπει να επικοινωνήσει με πολλούς σταθμούς εδάφους εντός περιορισμένου χρόνου. Τέτοιου τύπου προβλήματα ορίζονται συχνά ως Time-Dependent TSP, όπου τα κόστη αλλάζουν δυναμικά, ανάλογα με τον χρόνο.

Συντήρηση και αναβάθμιση υποδομών:

Η τακτική συντήρηση ή αναβάθμιση εξοπλισμού (π.χ. switches, routers, repeaters) από τεχνικά συνεργεία, ειδικά σε απομακρυσμένες περιοχές, απαιτεί σχεδιασμό βέλτιστων διαδρομών επισκέψεων. Το πρόβλημα αυτό είναι κλασικό TSP και εφαρμόζεται από παρόχους τηλεπικοινωνιών για τη βελτιστοποίηση κόστους και χρόνου υποστήριξης.

Πεδίο Εφαρμογής	**Μορφή TSP**	**Αντικειμενικός Σκοπός**
Καλωδιώσεις Οπτικών Ινών	Geometric TSP	Ελαχιστοποίηση μήκους/κόστους υλοποίησης
Multicast routing	Time-Dependent TSP	Ελαχιστοποίηση καθυστέρησης μετάδοσης
Δίκτυα Αισθητήρων	Energy-aware TSP	Ελαχιστοποίηση κατανάλωσης ενέργειας
Τεχνική Συντήρηση Κόμβων	Multiple TSP	Ελαχιστοποίηση χρόνου διαδρομής και εξισορρόπηση

Πίνακας 2.6.5: Ενδεικτική εφαρμογή TSP σε δίκτυα.

2.6.6 Γραφικά Υπολογιστών και Τέχνη

Το πρόβλημα του ταξιδεύοντος πωλητή (TSP), παρά τον μαθηματικό και αλγοριθμικό του χαρακτήρα, έχει βρει ενδιαφέρουσες και πρωτότυπες εφαρμογές και στον τομέα των γραφικών υπολογιστών (computer graphics), αλλά και της υπολογιστικής τέχνης (computational art). Η ικανότητα του TSP να περιγράφει τη βέλτιστη ακολουθία επίσκεψης σε σημεία οδηγεί στη δημιουργία αποδοτικών, αισθητικά συνεκτικών και δομικά βελτιστοποιημένων απεικονίσεων, μοτίβων και σχεδίων.

Αναπαράσταση Εικόνας ως Διαδρομή (TSP Art):

Μία από όπως πιο χαρακτηριστικές καλλιτεχνικές εφαρμογές είναι η μετατροπή μιας εικόνας σε μονογραμμικό σχέδιο, μέσω όπως επίλυσης όπως προβλήματος TSP. Η διαδικασία έχει ως εξής :

- Η είσοδος είναι μία εικόνα (συνήθως πορτρέτο ή αντικείμενο υψηλής αντίθεσης).
- Μέσω τεχνικών επεξεργασίας εικόνας (π.χ. Canny edge detection) εξάγονται σημεία ελέγχου (control points).
- Τα σημεία θεωρούνται κόμβοι σε γράφο και επιλύεται το TSP για να βρεθεί η βέλτιστη διαδρομή που τα επισκέπτεται μία μόνο φορά.
- Το τελικό αποτέλεσμα είναι ένα μονοκονδυλιά σχέδιο, το οποίο αναπαριστά την αρχική εικόνα με υψηλή αισθητική αξία.

Αυτή η τεχνική είναι γνωστή ως TSP art και έχει εφαρμοστεί τόσο για καλλιτεχνικούς όσο και για ερευνητικούς σκοπούς, αξιοποιώντας αλγορίθμους όπως ο Lin–Kernighan, οι γενετικοί αλγόριθμοι, και η αποικία μυρμηγκιών.

Βελτιστοποίηση Σχεδίασης σε Εκτυπώσεις και Ρομποτικό Σχέδιο:

Στην ψηφιακή κατασκευή και σχεδίαση, το TSP χρησιμοποιείται για τη βελτιστοποίηση της σειράς με την οποία ένα μηχάνημα (όπως ένας ρομποτικός βραχίονας ή ένας εκτυπωτής CNC/laser cutter) θα εκτελέσει τις κινήσεις του ώστε να ελαχιστοποιηθεί ο χρόνος σχεδίασης, να μειωθούν οι περιττές κινήσεις (non-printing moves) και να διασφαλιστεί σταθερή και συνεχής εκτέλεση της διαδρομής.

Τεχνολογία Εικονικής και Επαυξημένης Πραγματικότητας (VR/AR):

Σε εφαρμογές εικονικής περιήγησης ή αλληλεπίδρασης με γραφικά περιβάλλοντα, ο χρήστης καλείται συχνά να ακολουθήσει μια συγκεκριμένη διαδρομή. Ο σχεδιασμός αυτών των διαδρομών, ιδίως όταν αφορούν επισκέψεις σε πολλαπλά "σημεία ενδιαφέροντος" μέσα σε έναν τρισδιάστατο χώρο, μπορεί να μοντελοποιηθεί ως πρόβλημα TSP ώστε να επιτυγχάνεται η βέλτιστη εμπειρία χρήστη και ομαλή ροή.

Κεφάλαιο 3ο: Αλγόριθμοι Επίλυσης TSP

Η επίλυση του προβλήματος του ταξιδεύοντος πωλητή (TSP) έχει αποτελέσει επί δεκαετίες αντικείμενο εντατικής μελέτης στον χώρο της θεωρητικής πληροφορικής, των μαθηματικών βελτιστοποίησης και της επιχειρησιακής έρευνας. Η υπολογιστική δυσκολία του προβλήματος, σε συνδυασμό με το πλήθος των πραγματικών εφαρμογών του, έχει οδηγήσει στην ανάπτυξη μιας ευρείας ποικιλίας αλγορίθμων για την προσέγγιση ή την ακριβή επίλυσή του.

Δεδομένου ότι το TSP ανήκει στην κατηγορία των NP-πλήρων προβλημάτων, η εύρεση μιας βέλτιστης λύσης απαιτεί, στη χειρότερη περίπτωση, εκθετικό χρόνο ως προς τον αριθμό των πόλεων (ή κόμβων). Συνεπώς, για μικρά προβλήματα (π.χ. έως 30-40 κόμβους) μπορούν να χρησιμοποιηθούν ακριβείς αλγόριθμοι, οι οποίοι εγγυώνται την εύρεση της βέλτιστης διαδρομής, έστω και με υψηλό υπολογιστικό κόστος.

Ωστόσο, για μεγαλύτερα προβλήματα πραγματικής κλίμακας, οι ακριβείς μέθοδοι καθίστανται μη εφαρμόσιμες στην πράξη. Σε αυτές τις περιπτώσεις προτιμώνται ευρετικοί ή μεταευρετικοί (προσεγγιστικοί) αλγόριθμοι, οι οποίοι παρέχουν «καλές» λύσεις εντός λογικού χρονικού διαστήματος, χωρίς να εξασφαλίζουν την παγκόσμια βέλτιστη λύση.

Σε αυτή την ενότητα θα ασχοληθούμε με τις δύο κύριες κατηγορίες αλγορίθμων, τους ακριβείς και τους ευρετικούς/προσεγγιστικούς.

3.1 Ακριβείς Αλγόριθμοι

Ένας αλγόριθμος ονομάζεται ακριβής όταν μας αποδίδει την άριστη λύση για κάθε είσοδο του προβλήματος. Παρότι οι μέθοδοι αυτές είναι εγγυημένα ορθές, το κόστος σε χρόνο και μνήμη αυξάνεται εκθετικά με το πλήθος των κόμβων, γεγονός που τις καθιστά κατάλληλες κυρίως για μικρής ή μεσαίας κλίμακας προβλήματα.

Στην ενότητα αυτή παρουσιάζονται οι βασικότεροι ακριβείς αλγόριθμοι που έχουν προταθεί για το TSP, με έμφαση τόσο στη θεωρητική τους βάση όσο και στην πρακτική τους αξιοποίηση.

3.1.1 Αλγόριθμος Branch and Bound

Ο αλγόριθμος Branch & Bound (B&B) θα μπορούσε να μεταφραστεί στα ελληνικά ως Αλγόριθμος Διακλάδωσης και Οριοθέτησης ή Επέκτασης και Οριοθέτησης όμως επιλέγεται στην παρούσα εργασία να χρησιμοποιείται, κατά κύριο λόγο, ο πρωτότυπος αγγλικός όρος. Είναι ένας γενικός αλγόριθμος για εύρεση βέλτιστων λύσεων για διάφορα προβλήματα βελτιστοποίησης, ειδικά για διακριτά ή συνδυαστικά προβλήματα. Αποτελείται από μια συστηματική απογραφή των υποψήφιων λύσεων, όπου μεγάλα υποσύνολα από αχρείαστα δεδομένα απορρίπτονται μαζικά, χρησιμοποιώντας άνω και κάτω εκτιμώμενα όρια της ποσότητας που βελτιστοποιείται. Η μέθοδος προτάθηκε από τους A. H. Land και A. G. Doig το 1960 και στηρίζεται στην χρήση άνω ή κάτω ορίων για τον περιορισμό του χώρου λύσεων.

Ο αλγόριθμος Branch and Bound ανήκει στην κατηγορία των ακριβών αλγορίθμων και χρησιμοποιείται ευρέως για την επίλυση του TSP. Έχουν αναπτυχθεί επίσης διάφοροι αλγόριθμοι Branch and Cut που αποτελούν γενίκευση του αλγορίθμου Branch and Bound. Είναι αρκετά περίπλοκο για τους Branch and Bound αλγορίθμους το πώς μπορεί να απλοποιηθεί το Πρόβλημα του Περιοδεύοντος Πωλητή.

Η βασική μέθοδος που ακολουθείται είναι η διαδιχική διάσπαση του συνόλου των διαδρομών σε όλο και μικρότερα υποσύνολα και ο υπολογισμός για κάθε ένα απ' αυτά του κατώτατου ορίου του μήκους της αντίστοιχης άριστης λύσης. Με βάση τα κατώτατα όρια διαχωρίζονται τα υποσύνολα και τελικά προσδιορίζεται μία άριστη διαδρομή. Αυτό γίνεται αν βρεθεί μία κλειστή διαδρομή, της οποίας το μήκος είναι μικρότερο ή ίσο από τα κατώτατα όρια όλων των άλλων υποσυνόλων. Τότε η διαδρομή αυτή είναι η άριστη.

Τα υποσύνολα διαδρομών παριστάνονται με τους κόμβους ενός δένδρου και η διαδικασία διαχωρισμού με τις διακλαδώσεις του δένδρου. Γι 'αυτό η μέθοδος αυτή ονομάστηκε ¨διακλάδωση και όριο¨.

Προκειμένου να διαμορφωθεί μαθηματικά το πρόβλημα ορίζονται :

$$t = [(i_1,i_2),\ (i_2,i_3),\ \ldots..,\ (i_{n-1},i_n),\ (i_n,i_1)]$$

είναι δηλαδή, μία κλειστή διαδρομή, η οποία μπορεί να παρασταθεί σαν ένα σύνολο από n διατεταγμένα ζεύγη θέσεων.

$$z(t) = \sum d_{i,j} \quad (i,j) \in t$$

Είναι το μήκος της διαδρομής t και ισούται με το άθροισμα των στοιχείων του πίνακα αποστάσεων D, τα οποία ανήκουν σ' αυτήν. Αξίζει να σημειωθεί ότι η διαδρομή t περνάει μία και μόνο μία φορά από κάθε θέση. Επομένως, στον πίνακα αποστάσεων τα στοιχεία που αποτελούν τη διαδρομή t είναι τοποθετημένα έτσι ώστε να υπάρχει ένα και μόνο ένα σε κάθε γραμμή και κάθε στήλη.

X, Y, $\bar{Y}$ = κόμβοι του δένδρου λύσεων.

ω(X) = το κατώτατο όριο του μήκους των διαδρομών του κόμβου X.

Δηλαδή, ισχύει : $z(t) \geq \omega(X)$ για κάθε διαδρομή του κόμβου X.

z_0 = το μήκος της άριστης διαδρομής που έχει βρεθεί σε ένα αποιοδήποτε στάδιο της αλγοριθμικής διαδικασίας.

<u>Υπολογισμός κατώτατων ορίων – Μετασχηματισμός πίνακα.</u>

Τα κατώτατα όρια υπολογίζονται μετασχηματίζοντας τον πίνακα αποστάσεων. Ένας πίνακας D ονομάζεται μετασχηματισμένος αν όλα τα στοιχεία του είναι μη αρνητικά και υπάρχει τουλάχιστον ένα μηδενικό στοιχείο σε κάθε γραμμή και στήλη.

Ο μετασχηματισμός γίνεται ως εξής :

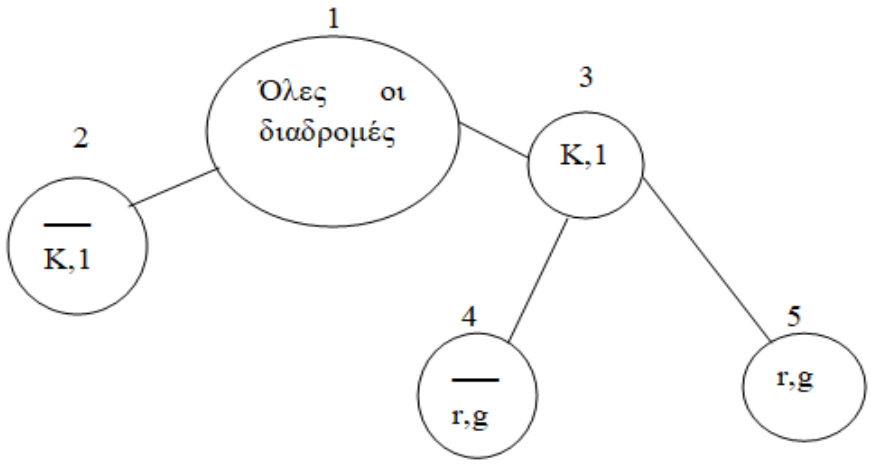
Αφαιρείται το μικρότερο στοιχείο κάθε γραμμής από όλα τα στοιχεία της γραμμής αυτής. Επαναλμβάνεται το ίδιο για κάθε στήλη.

Αν z(t) είναι το μήκος μιας διαδρομής t στον αρχικό πίνακα D, $z_1(t)$ το μήκος της διαδρομής στον μετασχηματισμένο πίνακα D και h το άθροισμα των σταθερών που χρησιμοποιήθηκαν για τον μετασχηματισμό, τότε ισχύει : $z(t) = h + z_1(t)$

Εφόσον ο μετασχηματισμένος πίνακας περιέχει μόνο μη αρνητικά στοιχεία, η τιμή h αποτελεί το κατώτατο όριο του μήκους της διαδρομής t στον αρχικό πίνακα.

Διακλαδώσεις και κόμβοι του δένδρου λύσεων.

Ο διαμερισμός του συνόλου των διαδρομών σε ξένα μεταξύ τους υποσύνολα δημιουργείται από την διακλάδωση ενός δένδρου και παρουσιάζεται στο Σχήμα 3.1.1



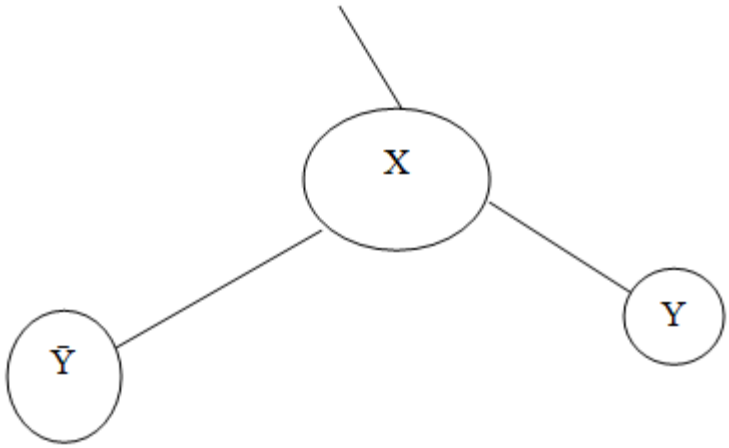
Σχήμα 3.1.1: Διακλαδώσεις και Κόμβοι Δένδρου Λύσεων

Για την επεξήγηση της λειτουργείας της διακλάδωσης με βάση τα στοιχεία του σχήματος 1 έχουμε :

Αρχικά όλες οι δυνατές διαδρομές Hamilton περιέχονται στο σύνολο που αντιστοιχεί στον κόμβο ¨Όλες οι διαδρομές¨. Στη συνέχεια το σύνολο αυτό διασπάται στα σύνολα των διαδρομών που αντιστοιχούν στους κόμβους 2 και 3. Πιο συγκεκριμένα, ο κόμβος 2 περιέχει όλες τις διαδρομές Hamilton που δεν περιέχουν τη μετάβαση K→1 και ο κόμβος 3 τις διαδρομές Hamilton που περιέχουν τη μετάβαση K→1. Κατά τον ίδιο τρόπο το σύνολο των διαδρομών που αντιστοιχούν στον κόμβο 3 διασπάται σε 2 υποσύνολα, ξένα μεταξύ τους, και τα οποία αντιστοιχούν στους κόμβους 4 και 5. Έτσι κάθε διαδρομή Hamilton του συνόλου που αντιστοιχεί στο κόμβο 4 δεν περιέχει τη μετάβαση r→g και περιέχει την K→1. Ομοίως κάθε διαδρομή Hamilton του συνόλου που αντιστοιχεί στον κόμβο 5 περιέχει τη μετάβαση r→g και την K→1.

Στα παρακάτω υιοθετείται η εξής παρουσίαση της διακλάδωσης:

Όταν ένας κόμβος X διασπάται σε δύο άλλους, ο κόμβος που βρίσκεται δεξιά του X συμβολίζεται με Y και αυτός που βρίσκεται αριστερά με $\bar{Y}$ (Σχήμα 3.1.2).



Σχήμα 3.1.2: Διάσπαση κόμβου σε άλλους

Προσδιορισμός των κόμβων Y και Ȳ και των αντίστοιχων κατώτατων ορίων.

Έστω ο πίνακας που αντιστοιχεί στον κόμβο X. Ο πίνακας αυτός περιέχει ένα τουλάχιστον μηδενικό στοιχείο σε κάθε γραμμή και στήλη.

Για κάθε μηδενικό στοιχείο του πίνακα που αντιστοιχεί στο ζεύγος (i,j) προσδιορίζονται τα μεγέθη α(i), β(j), και θ(i,j) ως εξής :

α(i) = η επιπλέον απόσταση που διανύεται λόγω της μη εκλογής του μικρότερου στοιχείου της γραμμής i. Ισούται με το αμέσως μικρότερο στοιχείο της γραμμής i(1,2,…,n).

β(j) = η επιπλέον απόσταση που διανύεται λόγω της μη εκλογής του μικρότερου στοιχείου της στήλης j. Ισούται με το αμέσως μικρότερο στοιχείο της στήλης j(1,2,…,n).

$$\theta(i,j) = \alpha(i) + \beta(j), \forall\, d_{ij} = 0$$

και προσδιορίζεται η θέση (K,1) έτσι ώστε:

$$\theta(K,1) = \max\theta(i,j), \forall(i,j) \text{ έτσι που } d_{ij} = 0.$$

Ο κόμβος X λοιπόν διακλαδίζεται στους κόμβους (K,1) και (K,1). Για τον κόμβο (K,1) υπολογίζεται το κατώτατο όριο:

$$\omega(K,1) = \omega(X) + \theta(K,1)$$

Εφόσον η επί μέρους διαδρομή (K,1) περιέχεται στον κόμβο Y, διαγράφονται η γραμμή K η στήλη 1 από τον πίνακα που αντιστοιχεί στον κόμβο X. Η γραμμή K διαγράφεται για να αποκλειστεί η πιθανότητα εκλογής σε οποιδήποτε μεταγενέστερο στάδιο μιας διαδρομής (K,p) όπου $p \neq 1$. Ομοίως, η διαγραφή της στήλης 1 αποκλείει την εκλογή μιας διαδρομής (q,1) όπου $q \neq K$.

Κατ' αυτόν τον τρόπο προκύπτει πίνακας ελαττωμένος κατά μία γραμμή και κατά μία στήλη.

Το στοιχείο (1,K) στον νέο αυτό πίνακα τίθεται ίσο προς το άπειρο για να μην εμφανιστεί ο κλειστός βρόγχος K→1→K. Επίσης, αν η διαδρομή (K,1) σχηματίζει αλυσίδα με μία προηγούμενη διαδρομή του δένδρου λύσεων, π.χ. με την (s,K), τότε και το στοιχείο (1,s) τίθεται ίσο προς το άπειρο, για τον ίδιο λόγο, όπως και προηγουμένως.

Μετά τις παραπάνω τροποποιήσεις, ο πίνακας μετασχηματίζεται(αν χρειάζεται) ούτως ώστε να υπάρχει ένα τουλάχιστον μηδέν σε κάθε γραμμή και στήλη και υπολογίζεται το άθροισμα h των σταθερών που χρησιμοποιήθηκαν για τον μετασχηματισμό.

Για τον κόμβο (K,1) υπολογίζεται το κατώτατο όριο ως εξής:

$$\omega(K,1) = \omega(X) + h(K,1)$$

Το κατώτατο όριο του κόμβου (K,1) συγκρίνεται με το κατώτατο όριο του κόμβου (K,1) καθώς και με τα κατώτατα όρια των τελικών κόμβων του δένδρου. Η διακλάδωση συνεχίζεται από τον κόμβο με το μικρότερο κατώτατο όριο.

Η διαδικασία προσδιορισμού των κόμβων Y και $\bar{Y}$ συνεχίζεται με τον ίδιο τρόπο, έως ότου ο πίνακας που προκύπτει είναι διαστάσεων 2x2. Στον πίνακα αυτόν υπάρχουν δύο μόνο επί μέρους διαδρομές, οι οποίες συμπληρώνουν τη κλειστή διαδρομή.

3.1.2 Αλγόριθμος Cutting Plane

Ο αλγόριθμος Cutting Plane αποτελεί μία από τις πιο σημαντικές μεθόδους στην επίλυση του TSP ως προβλήματος ακέραιου γραμμικού προγραμματισμού, ειδικά όταν συνδυάζεται με τεχνικές όπως το Branch and Bound. Ο αλγόριθμος Cutting Plane είναι μια μέθοδος επίλυσης ακέραιων γραμμικών προβλημάτων που βασίζεται στην προσθήκη κοπών (cuts) επιπλέον περιορισμών, οι οποίοι εξαλείφουν ανεπιθύμητες (μη ακέραιες ή ανεφάρμοστες) λύσεις της χαλαρωμένης μορφής του προβλήματος, χωρίς να αποκλείουν καμία εφικτή ακέραιη λύση. Για το TSP, η προσέγγιση αυτή ξεκινά με τη γραμμική χαλάρωση του προβλήματος (δηλαδή, επιτρέποντας $x_{ij} \in [0,1]$), και προοδευτικά προσθέτει περιορισμούς που αποκλείουν υποκύκλους (subtours) ή άλλες μη αποδεκτές δομές.

Η βασική ιδέα για επίλυση με τον Αλγόριθμο Cutting Plane περιλαμβάνει:

- Λύση της αρχικής χαλάρωσης του TSP (χωρίς περιορισμούς υποκύκλων).
- Έλεγχος για ύπαρξη υποκύκλων: Αν η λύση περιέχει περισσότερους από έναν κύκλους, προστίθεται ένας ή περισσότεροι subtour elimination constraints.
- Προσθήκη των cuts και επίλυση ξανά του τροποποιημένου γραμμικού προβλήματος.
- Επανάληψη μέχρι η βέλτιστη λύση να είναι ένας ενιαίος κύκλος (δηλαδή έγκυρη λύση του TSP).

Η μέθοδος Cutting Plane εφαρμόστηκε επιτυχώς από την Concorde TSP Solver, τον ταχύτερο και πιο αποδοτικό επιλυτή για το συμμετρικό TSP, ο οποίος συνδυάζει Cutting Planes με Branch and Bound. Το 2006, η Concorde χρησιμοποίησε αυτήν την τεχνική για να επιλύσει ένα TSP με 85.900 πόλεις που αντιστοιχούσαν σε αποστάσεις μεταξύ σημείων σε chip της VLSI(Very Large-Scale Integration) τεχνολογίας!

<u>Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:</u>

Πλεονεκτήματα:

- Πολύ ισχυρή μέθοδος για ακριβή επίλυση του TSP.
- Μπορεί να αξιοποιηθεί με επιλυτές ILP (CPLEX, Gurobi).
- Παράγει πολύ καλές κάτω φραγές (lower bounds).

Μειονεκτήματα:

- Υπολογιστικά ακριβή για μεγάλα n.
- Χρειάζεται ειδικό αλγόριθμο για τον εντοπισμό υποκύκλων.
- Περίεργη υλοποίηση.

3.1.3 Αλγόριθμος Held-Karp (Δυναμικός Προγραμματισμός)

Ο αλγόριθμος Held-Karp αποτελεί μια από τις πιο γνωστές ακριβείς μεθόδους επίλυσης του προβλήματος του ταξιδεύοντος πωλητή, βασισμένος στην τεχνική του δυναμικού προγραμματισμού. Η βασική ιδέα είναι η κατανομή του προβλήματος σε μικρότερα, επαναλαμβανόμενα υποπροβλήματα, τα οποία αποθηκεύονται ώστε να αποφεύγεται ο επαναυπολογισμός. Αντί να εξετάζονται όλες οι πιθανές διαδρομές (όπως στην εξαντλητική αναζήτηση), ο Held-Karp υπολογίζει το ελάχιστο κόστος για να φτάσει κανείς σε κάθε πόλη, επισκεπτόμενος ένα συγκεκριμένο υποσύνολο πόλεων, καταλήγοντας τελικά στη συνολική βέλτιστη διαδρομή.

Η μείωση του αριθμού υπολογισμών επιτυγχάνεται μέσω της χρήσης πινάκων αποθήκευσης (memoization), στους οποίους καταγράφεται το βέλτιστο κόστος κάθε υποπροβλήματος. Η υπολογιστική του πολυπλοκότητα ανέρχεται σε $O(n^2 \cdot 2^n)$, γεγονός που τον καθιστά σημαντικά πιο αποδοτικό από τον Brute Force αλγόριθμο (με πολυπλοκότητα O(n!)), αλλά η τιμή του εξακολουθεί να είναι εκθετική και κατά συνέπεια καθιστά τον αλγόριθμο μη πρακτικό για πολύ μεγάλες τιμές του n.

Παρόλα αυτά, ο Held-Karp χρησιμοποιείται ευρέως για προβλήματα μεσαίου μεγέθους (μέχρι περίπου 20-25 πόλεις), ενώ παράλληλα αποτελεί σημαντικό σημείο αναφοράς για την αξιολόγηση της ποιότητας λύσεων που παράγουν οι προσεγγιστικοί και μεταευρετικοί αλγόριθμοι. Η δυνατότητα παροχής της βέλτιστης λύσης τον καθιστά πολύτιμο εργαλείο τόσο σε θεωρητικό επίπεδο όσο και για συγκριτικές αναλύσεις αλγορίθμων.

<u>Μαθηματική διατύπωση του Αλγορίθμου Held-Karp:</u>

Έστω ένα σύνολο πόλεων : $V=\{1,2,...,n\}$ και ένας πίνακας αποστάσεων :

D[i][j] = απόσταση από την πόλη i στην πόλη j.

Ο αλγόριθμος υπολογίζει το ελάχιστο κόστος διαδρομής χρησιμοποιώντας δυναμικό προγραμματισμό.

Ορισμός
ΚατάστασηςC(S,j)=ελάχιστο κόστος για να φτάσουμε στην πόλη j από την πόλη 1, επισκεπτόμενοι ακ ριβώς τις πόλεις στο σύνολο $S \subseteq V$, με $1 \in S$ και $j \in S$.

Αρχικοποίηση: Για κάθε πόλη $j \neq 1$: $C(\{1,j\},j) = D[1][j]$

Αναδρομική Σχέση: Για κάθε υποσύνολο $S \subseteq V$ με $1 \in S$ και $|S| > 2$ και για κάθε $j \in S$, $j \neq 1$:

$C(S,j) = \min_{i \in S, i \neq j}[C(S \setminus \{j\}, i) + D[i][j]]$

Δηλαδή: για να πάω στην πόλη j αφού έχω επισκεφθεί τις πόλεις στο σύνολο S, το καλύτερο μονοπάτι είναι να φτάσω σε κάποια πόλη $i \in S \setminus \{j\}$ και μετά να μετακινηθώ από το i στο j.

Τελική Λύση:

Το ελάχιστο συνολικό κόστος της περιοδείας είναι:

$$\min_{j \neq 1}[C(V,j) + D[j][1]]$$

όπου C(V,j) είναι το κόστος για να φτάσουμε στην πόλη j επισκεπτόμενοι όλες τις πόλεις και D[j][1] είναι η επιστροφή στην αρχική πόλη(1).

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Εγγυάται βέλτιστη λύση.
- Εύκολη υλοποίηση με πίνακες memoization.
- Κατάλληλος για προβλήματα με 20-25 κόμβους.

Μειονεκτήματα :

- Εκθετική πολυπλοκότητα.
- Υψηλές απαιτήσεις μνήμης.
- Δεν επεκτείνεται αποδοτικά για μεγάλα n.

3.1.4 Αλγόριθμος Εξαντλητικής Αναζήτησης(Brute Force)

Ο αλγόριθμος Brute Force, ή αλλιώς εξαντλητικής αναζήτησης, είναι η πιο απλή αλλά και υπολογιστικά ακριβή μέθοδος για την επίλυση του προβλήματος του ταξιδεύοντος πωλητή (TSP). Η βασική ιδέα του αλγορίθμου είναι να υπολογίσει όλες τις δυνατές διαδρομές που περνούν από κάθε πόλη ακριβώς μία φορά και επιστρέφουν στην αρχική, και να επιλέξει εκείνη με το μικρότερο συνολικό κόστος. Επιπλέον, σε σχέση με άλλες τεχνικές, χρησιμεύει ως σημείο αναφοράς (benchmark) για την αξιολόγηση της απόδοσης ευρετικών και ακριβών αλγορίθμων. Η σύγκριση της λύσης που βρίσκει ένας ευρετικός αλγόριθμος με τη βέλτιστη λύση του Brute Force, επιτρέπει την εκτίμηση της απόκλισης από τη βέλτιστη τιμή.

Δεδομένου ενός σύνολου n πόλεων και ενός πίνακα αποστάσεων ή βαρών c_{ij}, ο αλγόριθμος παράγει όλες τις πιθανές κυκλικές διαδρομές της μορφής : $P=\{\upsilon_1,\upsilon_2,...,\upsilon_n,\upsilon_1\}$

όπου κάθε υ_i είναι μία διαφορετική πόλη από το σύνολο και η διαδρομή ξεκινά και τελειώνει στην ίδια πόλη. Για κάθε τέτοια διαδρομή, υπολογίζεται το συνολικό κόστος:

Cost(P) = $\sum_{i=1}^{n} c$ui ui + 1 και αποθηκεύεται η διαδρομή με το ελάχιστο κόστος.

Η υπολογιστική πολυπλοκότητα του αλγορίθμου είναι: O(n!) (για μεγέθη πόλεων n > 1).

Αυτό σημαίνει ότι ο αριθμός των διαδρομών αυξάνεται εκθετικά με τον αριθμό των πόλεων. Π.χ. Για n>10, η πλήρης απαρίθμηση γίνεται πρακτικά ανεφάρμοστη, λόγω των χρονικών και υπολογιστικών πόρων που απαιτούνται.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Απόλυτη ακρίβεια – εγγυάται βέλτιστη λύση.
- Εύκολη υλοποίηση και κατανόηση.

- Χρήσιμος για μικρά παραδείγματα ή επαλήθευση άλλων αλγορίθμων.
- Χρήσιμος για εκπαιδευτικούς σκοπούς.

Μειονεκτήματα :

- Πολύ αργός για μεγάλα n.
- Δεν αξιοποιεί δομές ή φράγματα.
- Δεν είναι πρακτικό.

3.2 Προσεγγιστικοί Αλγόριθμοι

Καθώς το πρόβλημα του ταξιδεύοντος πωλητή (TSP) είναι NP-πλήρες, η εύρεση της βέλτιστης λύσης μέσω ακριβών αλγορίθμων γίνεται υπολογιστικά απαγορευτική για προβλήματα με μεγάλο αριθμό πόλεων. Ως εκ τούτου, υχνά καταφεύγουμε σε ευρετικούς ή προσεγγιστικούς αλγορίθμους, οι οποίοι προσφέρουν καλές λύσεις σε αποδεκτό χρόνο, χωρίς να εγγυώνται ότι αυτές είναι βέλτιστες.

Οι ευρετικοί αλγόριθμοι χρησιμοποιούνται είτε:

- Για την κατασκευή μίας αρχικής λύσης (constructive heuristics), είτε
- Για τη βελτίωση μιας υπάρχουσας λύσης (improvement heuristics),
- Ενώ άλλες τεχνικές, όπως οι μεταευρετικές μέθοδοι (metaheuristics), στοχεύουν στη συνολική εξερεύνηση του χώρου λύσεων, με τεχνικές εμπνευσμένες από φυσικά ή βιολογικά φαινόμενα.

3.2.1 Απλοί Ευρετικοί Αλγόριθμοι

Στην παρούσα εργασία, θα εξετάσουμε δύο βασικούς Απλούς ευρετικούς Αλγορίθμους:

- Nearest Neighbor (Κοντινότερου/Πλησιέστερου Γείτονα),
- Greedy Heuristic Algorithm (Άπληστος Αλγόριθμος).

3.2.1.1 Nearest Neighbor (Κοντινότερου/Πλησιέστερου Γείτονα)

Ο αλγόριθμος Nearest Neighbor (NN) είναι μία από τις απλούστερες και πιο δημοφιλείς ευρετικές προσεγγίσεις για την επίλυση του TSP. Η βασική του αρχή είναι να ξεκινάει από μία αρχική πόλη και να μετακινείται διαδοχικά προς την πλησιέστερη μη επισκεφθείσα πόλη, έως ότου όλες οι πόλεις να έχουν επισκεφθεί, και τέλος να επιστρέφει στην αρχική.

Βήματα του Αλγορίθμου:

- Επιλογή αρχικής πόλης (τυχαία ή συγκεκριμένη).
- Από την τρέχουσα πόλη, επιλέγεται η πόλη που δεν έχετε ακόμη επισκεφθεί.
- Επισκεπτόμαστε την πόλη και τη σημειώνουμε ως επισκεφθείσα.
- Αν υπάρχουν πόλεις που δεν έχουμε επισκεφθεί, επαναλαμβάνουμε το βήμα 2.
- Επιστρέφουμε στην αρχική πόλη.

Υπολογιστική Πολυπλοκότητα:O(n2), καθώς για κάθε μία από τις n πόλεις εξετάζουμε έως και n−1 γείτονες για να βρούμε τον κοντινότερο.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Εξαιρετικά απλός και γρήγορος.
- Παράγει αποδεκτές λύσεις για μικρού/μεσαίου μεγέθους προβλήματα.
- Χρήσιμος για αρχικοποίηση πιο σύνθετων ευρετικών αλγορίθμων.

Μειονεκτήματα :

- Η λύση μπορεί να αποκλίνει σημαντικά από τη βέλτιστη.
- Υπάρχει πιθανότητα «κακής αρχικής επιλογής» που οδηγεί σε κακή συνολική διαδρομή.
- Δεν παρέχει βελτιωτική λύση: μόλις ολοκληρωθεί, δεν προσπαθεί να διορθώσει/βελτιστοποιήσει το αποτέλεσμα.

Ο αλγόριθμος Nearest Neighbor παρέχει καλύτερα αποτελέσματα όταν εφαρμόζεται πολλαπλές φορές, ξεκινώντας κάθε φορά από διαφορετική αρχική πόλη και κρατείται κάθε φορά η καλύτερη λύση.

3.2.1.2 Greedy Heuristic Algorithm (Άπληστος Αλγόριθμος)

Ο Άπληστος Αλγόριθμος ανήκει στην κατηγορία ευρετικών αλγορίθμων και αναζητά τοπικά βέλτιστα, βελτιστοποιώντας περαιτέρω μία τοπικά βέλτιστη λύση αναζητώντας το ολικό βέλτιστο του προβλήματος. Βασίζεται στην απλή ιδέα της προσθήκης της μικρότερης δυνατής ακμής στο μονοπάτι κάθε φορά, αποφεύγοντας κύκλους και εξασφαλίζοντας ότι καμία πόλη δεν επισκέπτεται περισσότερες από δύο φορές. Ο αλγόριθμος συνεχίζει μέχρι να σχηματιστεί ένας πλήρης κύκλος που επισκέπτεται όλες τις πόλεις ακριβώς μία φορά. Χρησιμοποιείται συχνά ως σημείο εκκίνησης για πιο εξελιγμένες μεθόδους (π.χ. 2-opt, Tabu Search).

Βασική ιδέα: Ο αλγόριθμος διαλέγει την επόμενη διαθέσιμη ακμή με το ελάχιστο κόστος, υπό τις εξής συνθήκες:

- Δεν δημιουργείται πρόωρα κύκλος (δηλαδή πριν να περιληφθούν όλες οι πόλεις),
- Καμία πόλη δεν έχει βαθμό μεγαλύτερο του 2, ώστε να διατηρείται η δυνατότητα δημιουργίας κύκλου Hamilton στο τέλος.

Βήματα Αλγορίθμου:

- Κατασκευάζουμε μία λίστα με όλες τις ακμές ταξινομημένες κατά αύξον κόστος.
- Ξεκινώντας από τη μικρότερη, προσθέτουμε σταδιακά ακμές στο μονοπάτι αν :
- Δεν δημιουργούν πρόωρο κύκλο.
- Δεν οδηγούν σε πόλη με βαθμό > 2.
- Η διαδικασία ολοκληρώνεται όταν σχηματιστεί κύκλος Hamilton.

Συνολική πολυπλοκότητα: $O(n^2 \log n)$

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Πολύ απλός και γρήγορος.
- Δίνει αποδεκτές λύσεις σε λογικό χρόνο ακόμα και για μεγαλύτερα n.

Μειονεκτήματα:

- Είναι άπληστος: Δεν εξετάζει τη συνολική βελτιστοποίηση, αλλά μόνο τοπικές επιλογές.
- Η τελική διαδρομή μπορεί να απέχει σημαντικά από τη βέλτιστη.
- Δεν είναι προσαρμοστικός ούτε επαναληπτικός.

3.2.2 Βελτιωτικοί Ευρετικοί Αλγόριθμοι

Οι βελτιωτικές ευρετικές μέθοδοι (improvement heuristics) στοχεύουν στην αναδιαμόρφωση μιας ήδη υπάρχουσας λύσης του TSP με σκοπό τη βελτίωση του συνολικού κόστους της διαδρομής. Σε αντίθεση με τις κατασκευαστικές μεθόδους που ξεκινούν «από το μηδέν», οι βελτιωτικές ξεκινούν από μια αρχική διαδρομή (συνήθως προερχόμενη από κάποια άλλη ευρετική, όπως Nearest Neighbor ή Greedy) και εφαρμόζουν τοπικές αλλαγές για να την κάνουν καλύτερη. Η γενική ιδέα αυτών των μεθόδων βασίζεται στην εύρεση μιας <<γειτονικής>> λύσης, που θα έχει μικρότερο συνολικό κόστος, αντικαθιστώντας τη τρέχουσα λύση με τη βελτιωμένη και συνεχίζοντας.

Στην παρούσα εργασία, θα εξετάσουμε τρεις βασικούς Βελτιωτικούς ευρετικούς Αλγορίθμους:

- Αλγόριθμος 2-opt
- Αλγόριθμος 3-opt και γενικά k-opt
- Αλγόριθμος Lin-Kernighan
- Αλγόριθμος LK-H (Lin-Kernighan-Helsgaun)

3.2.2.1 Αλγόριθμος 2-opt

Ο αλγόριθμος 2-opt είναι ένας από τουςς πιο κλασικούς και ευρέως χρησιμοποιούμενους αλγόριθμους για το Πρόβλημα του Ταξιδεύοντος Πωλητή (TSP). Ανήκει στην κατηγορία με τους Αλγορίθμους τοπικής αναζήτησης και εφαρμόζεται με στόχο τη βελτίωση μιας υπάρχουσας διαδρομής, εξαλείφοντας κυκλικά μοτίβα και περιττές διασταυρώσεις που αυξάνουν το συνολικό κόστος. Αποτελεί θεμέλιο λίθο για πιο σύνθετους αλγορίθμους όπως οι k-opt, Lin-Kernighan.

Η βασική ιδέα της μεθόδου 2-opt έγκειται στην επιλογή δύο ακμών της τρέχουσας διαδρομής, την αφαίρεσή τους και την αντικατάσταση με άλλες δύο, ώστε να διατηρείται η εγκυρότητα του κύκλου και ιδανικά να προκύπτει μια διαδρομή με μικρότερο συνολικό κόστος. Αν η νέα διαδρομή έχει μικρότερο συνολικό μήκος, η αντικατάσταση διατηρείται. Η διαδικασία επαναλαμβάνεται για όλα τα πιθανά ζεύγη ακμών, έως ότου δεν είναι δυνατή περαιτέρω βελτίωση.

Βήματα του Αλγορίθμου:

Έστω ότι η αρχική διαδρομή περνάει από τις πόλεις με τη σειρά : $P = (\upsilon_1,\upsilon_2,\ldots,\upsilon_n,\upsilon_1)$

- Επανάληψη για κάθε δυνατή δυάδα θέσεων (i,j) με $1 < i < j \leq n$
- Αντικατάσταση του υπομονοπατιού $(\upsilon_i,\upsilon_{i+1},\ldots,\upsilon_j)$ με το αντεστραμμένο $(\upsilon_j,\upsilon_{j-1},\ldots,\upsilon_i)$.
- Δημιουργείται νέα διαδρομή P′.
- Υπολογισμός του συνολικού κόστους της νέας διαδρομής :
- $C(P') = \sum_{k=1}^{n} d(\upsilon k, \upsilon k + 1)$
- Αν C(P′) < C(P) τότε:
- Θέτουμε P = P′
- Επαναλαμβάνουμε τη διαδικασία.
- Η διαδικασία ολοκληρώνεται όταν δεν προκύπτει καμία βελτίωση.

Υπολογιστική Πολυπλοκότητα: Ένας πλήρης γύρος 2-opt απαιτεί $O(n^2)$ εναλλαγές.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Πολύ απλός στην υλοποίηση.
- Ικανός να βελτιώσει σημαντικά αρχικές λύσεις.
- Χρήσιμος ως ενδιάμεσο βήμα σε πιο σύνθετους αλγορίθμους.

Μειονεκτήματα:

- Μπορεί να συγκλίνει σε τοπικά ελάχιστα.
- Δεν εξασφαλίζει βέλτιστη λύση.
- Εξαρτάται έντονα από την αρχική διαδρομή.

3.2.2.2 Αλγόριθμος 3-opt και γενικά k-opt

Ο Αλγόριθμος 3-opt και γενικότερα k-opt αποτελούν επεκτάσεις της βασικής τεχνικής 2-opt, με στόχο την περαιτέρω βελτίωση της ποιότητας της λύσης στο πρόβλημα του Ταξιδεύοντος Πωλητή (TSP). Ανήκουν στην κατηγορία των Ευρετικών Αλγορίθμων βελτίωσης, οι οποίοι ξεκινούν από μια αρχική εφικτή λύση και προσπαθούν να την αναδιαμορφώσουν, μειώνοντας το συνολικό κόστος του κύκλου μέσω τοπικών επανασυνδέσεων.

Ο Αλγόριθμος 3-opt επεκτείνει τη λογική του 2-opt επιτρέποντας την αφαίρεση τριών ακμών από τον κύκλο και την ανασύνδεση των τμημάτων με τρόπο που δημιουργεί μια νέα έγκυρη διαδρομή. Ο Εξετάζει όλες τις δυνατές ανασυνδέσεις των τριών κομμένων τμημάτων (μέχρι και 7 διαφορετικές συνδέσεις), επιλέγοντας εκείνη με το χαμηλότερο συνολικό κόστος. Ένας πλήρης γύρος 3-opt απαιτεί O(n3) επανασυνδέσεις. Χρησιμοποιείται συχνά με περιορισμούς (π.χ. μόνο όταν δύο ακμές τέμνονται)

για να μειωθεί ο υπολογιστικός φόρτος. Συνήθως βρίσκει καλύτερες λύσεις από τον Αλγόριθμο 2-opt, αλλά απαιτεί περισσότερο υπολογιστικό χρόνο και είναι πιο δύσκολος στην υλοποίηση.

Ο Αλγόριθμος k-opt είναι η γενίκευση των παραπάνω μεθόδων. Στον k-opt αφαιρούνται k ακμές από την υφιστάμενη διαδρομή και εξετάζονται οι εναλλακτικοί τρόποι επανασύνδεσης των υπομονοπατιών, ώστε να σχηματιστεί νέος έγκυρος κύκλος.

Δεδομένης μιας αρχικής έγκυρης λύσης(κύκλος Hamilton), ο Αλγόριθμος επιλέγει k ακμές του κύκλου, τις αφαιρεί και επιχειρεί να επανασυνδέσει τα τμήματα του μονοπατιού που προκύπτουν με διαφορετικό τρόπο, ώστε να σχηματιστεί νέος κύκλος μικρότερου κόστους. Αν η νέα λύση είναι βελτιωμένη, γίνεται αποδεκτή. Η διαδικασία επαναλαμβάνεται μέχρι να μην είναι πλέον δυνατή περαιτέρω βελτίωση με ανταλλαγές μεγέθους k. Ο χρόνος αναζήτησης είναι $O(n^k)$ για κάθε πλήρη αναζήτηση όλων των συνδυασμών. Ο Αλγόριθμος k-opt συνήθως εφαρμόζεται με περιορισμούς, π.χ. σε κοντινά σημεία ή μόνο για τέμνουσες ακμές, ώστε να καταστεί πρακτικά υλοποιήσιμο.

Στην πράξη, ο k-opt εφαρμόζεται κυρίως με μικρές τιμές του k (2–5) ή μέσω ευφυών παραλλαγών όπως ο Lin-Kernighan. Πολλοί εξελιγμένοι ευρετικοί και υβριδικοί αλγόριθμοι χρησιμοποιούν k-opt ως βασικό στοιχείο βελτίωσης της λύσης, ιδιαίτερα σε συνδυασμό με αναζήτηση γειτονιάς (neighborhood search) και μεταευρετικές τεχνικές.

<u>Πλεονεκτήματα-μειονεκτήματα αλγορίθμου k-opt:</u>

Πλεονεκτήματα:

- Ικανός να ξεπερνά τοπικά ελάχιστα που "παγιδεύουν" τις μεθόδους 2-opt ή 3-opt.
- Δυνατότητα παραγωγής λύσεων πολύ κοντά στο βέλτιστο.
- Βασικό συστατικό σε προηγμένους μεταευρετικούς αλγορίθμους (π.χ. Lin-Kernighan, Iterated Local Search).

Μειονεκτήματα:

- Υψηλή υπολογιστική πολυπλοκότητα για μεγάλες τιμές του k.
- Περισσότερο περίπλοκος στην υλοποίηση και στη διαχείριση των επανασυνδέσεων.
- Κίνδυνος υπερπροσαρμογής (overfitting) σε στατικά δεδομένα.

Οι Αλγόριθμοι 3-opt και k-opt προσφέρουν σημαντικές δυνατότητες βελτίωσης των λύσεων του TSP, ιδιαίτερα όταν χρησιμοποιούνται σε συνδυασμό με άλλους ευρετικούς ή μεταευρετικούς Αλγορίθμους. Αν και παρουσιάζουν αυξημένο υπολογιστικό κόστος σε σύγκριση με τον 2-opt, έχουν αποδειχθεί αποτελεσματικοί στην εύρεση υψηλής ποιότητας προσεγγιστικών λύσεων σε πλήθος πρακτικών εφαρμογών.

3.2.2.3 Αλγόριθμος Lin-Kernighan

Ο αλγόριθμος Lin-Kernighan είναι μια εξελιγμένη ευρετική μέθοδος για την επίλυση του προβλήματος του περιοδεύοντος πωλητή(TSP). Προτάθηκε από τους Shen Lin και Brian W. Kernighan το 1973 και θεωρείται μία από τις πιο αποτελεσματικές ευρετικές για προβλήματα μικρού έως μεσαίου μεγέθους (μέχρι ~10.000 κόμβους). Η μέθοδος βασίζεται στην ιδέα του k-opt, αλλά με δυναμική προσαρμογή του k κατά τη διάρκεια της εκτέλεσης.

Τα κύρια χαρακτηριστικά του Αλγορίθμου είναι:

- Βασίζεται σε τοπικές ανταλλαγές ακμών, όπως ο k-opt, αλλά δεν καθορίζεται εκ των προτέρων το k.
- Αντίθετα, επιλέγεται δυναμικά, όσο συνεχίζει να προκύπτει βελτίωση στο μονοπάτι.
- Αν συνδυαστεί με καλά αρχικά μονοπάτια (π.χ. Greedy, Nearest Neighbor), παράγει εξαιρετικά καλές λύσεις.

Βήματα του Αλγορίθμου:

- Αρχικά παίρνουμε έναν έγκυρο κύκλο, π.χ. από Nearest Neighbor.
- Ξεκινάμε με ανταλλαγή 2 ακμών (2-opt) που μειώνει το κόστος.
- Αν η ανταλλαγή είναι επιτυχής, αναζητούμε επόμενη βελτιωτική ανταλλαγή, οδηγώντας σε 3-opt, 4-opt, κ.λπ.
- Η διαδικασία συνεχίζεται έως ότου δεν παρατηρείται περαιτέρω βελτίωση.
- Επιστρέφουμε στην καλύτερη λύση που επιτεύχθηκε κατά τη διάρκεια της τρέχουσας αναζήτησης.
- Η διαδικασία επαναλαμβάνεται από νέα σημεία εκκίνησης για αποφυγή τοπικών ελαχίστων.

Ο LK χρησιμοποιεί επιπλέον τεχνικές για βελτιωμένη απόδοση:

- Candidate sets: Περιορίζει τις ακμές που εξετάζονται μόνο σε εκείνες με μικρές αποστάσεις.
- Don’t look bits: Μηχανισμός για να αποφεύγονται περιττές επαναλήψεις.
- α-β pruning: Αποφυγή υποσχόμενων αλλά ατελέσφορων ανταλλαγών.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Πολύ αποτελεσματικός για μεγάλα προβλήματα TSP.
- Δυναμική προσαρμογή του μεγέθους ανταλλαγής k.
- Αποφεύγει εύκολα τοπικά ελάχιστα.

Μειονεκτήματα:

- Πιο πολύπλοκη υλοποίηση από άλλες ευρετικές.
- Εξαρτάται από την ποιότητα της αρχικής λύσης.
- Απαιτεί προσεκτική ρύθμιση παραμέτρων για βέλτιστη απόδοση.

Ο αλγόριθμος Lin-Kernighan έχει χρησιμοποιηθεί σε Εμπορικούς και ακαδημαϊκούς λύτες TSP (π.χ. Concorde Solver) καθώς και σε βιομηχανικές εφμαρμογές για δρομολόγηση οχημάτων. Αποτελεί θεμέλιο λίθο στη σύγχρονη βιβλιογραφία του TSP. Η βελτιωμένη εκδοχή του, ο LK-H (Lin-Kernighan-Helsgaun), παραμένει ένας από τους πιο γρήγορους και ακριβείς αλγορίθμους που υπάρχουν.

3.2.2.4 Αλγόριθμος LK-H (Lin-Kernighan-Helsgaun)

Ο Lin-Kernighan-Helsgaun (LKH) είναι ένας προηγμένος ευρετικός αλγόριθμος για την προσέγγιση του TSP, σχεδιασμένος από τον Keld Helsgaun το 2000. Αποτελεί τη βελτιστοποιημένη υλοποίηση και επέκταση του αλγορίθμου Lin-Kernighan, και είναι από τους ταχύτερους και πιο αξιόπιστους αλγορίθμους για μεγάλης κλίμακας TSP προβλήματα. Ο ίδιος ο Keld Helsgaun παρέχει ελεύθερα διαθέσιμη υλοποίηση του αλγορίθμου στη Γλώσσα Προγραμματισμού C, η οποία χρησιμοποιείται ως βάση σε πλήθος ερευνητικών έργων και εμπορικών εφαρμογών.

Βασίζεται στην ίδια ιδέα με τον LK: δυναμική ακολουθία ανταλλαγών ακμών (k-opt), ενσωματώνοντας όμως νέα στρατηγική επιλογής υποψήφιων ακμών.

Οι κύριες Βελτιώσεις που έγιναν:

- Εισάγει ανορθόδοξες συναρτήσεις κόστους ώστε να οδηγήσει πιο επιθετικά τη βελτιστοποίηση.
- Αντί για απλή ευκλείδεια εγγύτητα, χρησιμοποιεί την πιο "ευφυή" προσέγγιση των α-nearness measures, με βάση το MST (Minimum Spanning Tree).
- Εστιάζει μόνο σε "υποσχόμενες" ανταλλαγές, μειώνοντας το χρόνο εκτέλεσης χωρίς μείωση ποιότητας.

<u>Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:</u>

Πλεονεκτήματα:

- Πολύ υψηλή απόδοση για μεγάλα προβλήματα.
- Ελάχιστη απόκλιση από το βέλτιστο (συχνά < 1%).
- Ευέλικτος και παραμετροποιήσιμος.

Μειονεκτήματα:

- Σχετικά περίπλοκος στην υλοποίηση.
- Δεν εξασφαλίζει βέλτιστη λύση θεωρητικά.
- Απαιτεί καλή κατανόηση παραμέτρων και fine-tuning.

Συμπερασματικά, ο αλγόριθμος LKH αποτελεί μία από τις κορυφαίες επιλογές για την επίλυση του TSP σε πρακτικό περιβάλλον. Η συνδυαστική του ισχύς, τα candidate sets και η δυναμική προσέγγιση τον καθιστούν εξαιρετικά αποτελεσματικό και επίκαιρο.

3.2.3 Μεταευρετικοί Αλγόριθμοι (Metaheuristics)

Οι μεταευρετικοί αλγόριθμοι αποτελούν εξελιγμένες τεχνικές βελτιστοποίησης που έχουν σχεδιαστεί ώστε να υπερβούν τα μειονεκτήματα των βασικών ευρετικών. Σε αντίθεση με τις τοπικές ευρετικές, που συχνά "παγιδεύονται" σε τοπικά ελάχιστα, οι μεταευρετικές ενσωματώνουν στρατηγικές εξερεύνησης και στοχαστικά στοιχεία, επιτρέποντας την αναζήτηση σε ευρύτερες περιοχές του χώρου λύσεων. Οι τεχνικές αυτές δεν εγγυώνται εύρεση της βέλτιστης λύσης, αλλά έχουν αποδειχθεί εξαιρετικά αποδοτικές στην πράξη, ειδικά για πολύπλοκα ή μεγάλης κλίμακας προβλήματα όπως το πρόβλημα του περιοδεύων πωλητή (TSP).

Χαρακτηριστικά Μεταευρετικών:

- Εξερεύνηση και εκμετάλλευση του χώρου λύσεων (exploration vs. exploitation).
- Χρήση στοχαστικών τεχνικών για αποφυγή τοπικών ακροτάτων.
- Ευελιξία και δυνατότητα προσαρμογής σε συγκεκριμένες παραλλαγές του προβλήματος.

Στην παρούσα εργασία, θα εξετάσουμε τρεις βασικούς μεταευρετικούς Αλγορίθμους:

- Simulated Annealing (Προσομοιωμένη Ανόπτηση),
- Tabu Search,
- Γενετικοί Αλγόριθμοι (Genetic Algorithms).

3.2.3.1 Αλγόριθμος Simulated Annealing (Προσομοιωμένη Ανόπτηση)

Ο αλγόριθμος Simulated Annealing (SA) είναι μια στοχαστική μεταευρετική μέθοδος βελτιστοποίησης που εμπνέεται από τη φυσική διαδικασία της ανόπτησης μετάλλων: την αργή ψύξη ενός υλικού ώστε να επιτευχθεί σταθερή κρυσταλλική δομή με ελάχιστη ενέργεια. Στο πλαίσιο της υπολογιστικής επιστήμης, η ιδέα αυτή εφαρμόζεται στην αναζήτηση προσεγγιστικά βέλτιστων λύσεων σε προβλήματα βελτιστοποίησης όπως το TSP. Παρά τη σχετική απλότητά του, ο αλγόριθμος Simulated Annealing παραμένει ένα σημαντικό εργαλείο στη συνδυαστική βελτιστοποίηση και μπορεί να δώσει λύσεις κοντά στο βέλτιστο όταν υλοποιηθεί σωστά.

Το SA ξεκινά από μια αρχική λύση και προχωρά σε διαδοχικές "γειτονικές" λύσεις μέσω μικρών μεταβολών (π.χ. ανταλλαγή δύο πόλεων). Αν η νέα λύση είναι καλύτερη, γίνεται αποδεκτή. Αν είναι χειρότερη, μπορεί να γίνει αποδεκτή με κάποια πιθανότητα, η οποία μειώνεται όσο "ψύχεται" το σύστημα. Αυτό επιτρέπει στο SA να ξεφύγει από τοπικά ελάχιστα και να εξερευνήσει τον χώρο λύσεων πιο αποτελεσματικά.

<u>Βήματα του Αλγορίθμου:</u>

- Επιλέγεται μία αρχική λύση(τυχαία ή βασισμένη σε ευρετικό Αλγόριθμο, π.χ. Nearest Neighbor).
- Ακολουθεί η δημιουργία συνάρτησης κόστους που περιλαμβάνει το μήκος της διαδρομής.
- Επιλέγεται η γειτονική λύση που δημιουργείται είτε με απλή μετάθεση, 2-opt, ή άλλη τοπική τροποποίηση.
- Στη συνέχεια εισάγουμε τη θερμοκρασία, η οποία καθορίζει την πιθανότητα αποδοχής χειρότερων λύσεων.
- Καθορίζουμε με το σχέδιο ψύξης(Cooling Schedule) πώς μειώνεται η θερμοκρασία με τον χρόνο.
- Τέλος, ορίζουμε ένα κριτήριο τερματισμού. Π.χ. Μέγιστος αριθμός επαναλήψεων.

Η πιθανότητα αποδοχής μιας χειρότερης λύσης δίνεται από τον τύπο:

$$P(\Delta E, T) = e^{-\Delta E/T}$$ όπου:

ΔE = κόστος νέας λύσης − κόστος τρέχουσας λύσης,

T = τρέχουσα «θερμοκρασία». Καθώς T→0, η πιθανότητα αποδοχής χειρότερης λύσης μειώνεται.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Μπορεί να ξεφύγει από τοπικά ακρότατα.
- Απλός αλγόριθμος, εύκολος στην υλοποίηση.
- Ευέλικτος σε συνδυασμό με τοπικές ευρετικές (π.χ. 2-opt).

Μειονεκτήματα:

- Ευαίσθητος στην επιλογή παραμέτρων (αρχική θερμοκρασία, ρυθμός ψύξης).
- Πιθανώς μεγάλος χρόνος εκτέλεσης αν δεν βελτιστοποιηθεί σωστά.
- Δεν εγγυάται βέλτιστη λύση.

Ο αλγόριθμος Simulated Annealing έχει χρησιμοποιηθεί εκτενώς στην επίλυση του TSP, ιδιαίτερα για μεσαίου μεγέθους προβλήματα. Συχνά ενσωματώνεται σε υβριδικές μεθόδους, όπου η αρχική λύση προκύπτει από γρήγορη ευρετική (π.χ. greedy), και η τοπική βελτιστοποίηση επιτυγχάνεται μέσω κινήσεων τύπου 2-opt ή 3-opt κατά τη διάρκεια της "ψύξης".

3.2.3.2 Αλγόριθμος Αναζήτησης Ταμπού(Tabu Search)

Ο Αλγόριθμος Αναζήτησης Ταμπού είναι ένας εξελιγμένος μεταευρετικός αλγόριθμος, που επινοήθηκε από τον Fred Glover τη δεκαετία του 1980. Σε αντίθεση με απλές τοπικές αναζητήσεις, που "κολλάνε" εύκολα σε τοπικά ελάχιστα, ο Tabu Search χρησιμοποιεί μηχανισμούς μνήμης για να αποφύγει την επανάληψη προηγούμενων καταστάσεων και να ενισχύσει την εξερεύνηση του χώρου λύσεων. Η κεντρική ιδέα του Tabu Search είναι η χρήση ενός "ταμπού καταλόγου" (tabu list**)**, ο οποίος αποθηκεύει πρόσφατες κινήσεις ή λύσεις που απαγορεύεται να επαναληφθούν για κάποιο χρονικό διάστημα. Με αυτόν τον τρόπο, ο αλγόριθμος αποφεύγει την ανακύκλωση και ξεφεύγει από τοπικά ελάχιστα. Η επιτυχία του Tabu Search εξαρτάται σημαντικά από την επιλογή των παραμέτρων (μήκος tabu list, στρατηγικές αποδοχής, restart policies), αλλά προσφέρει εξαιρετική ποιότητα λύσεων σε συνδυαστικά προβλήματα.

Βήματα του Αλγορίθμου:

- Επιλέγεται μία αρχική λύση(τυχαία ή Ευρετική).
- Ακολουθεί μία γειτονική αναζήτηση όπως συμβαίνει στο local search (π.χ. 2-opt).
- Στη συνέχεια εισάγεται μία δομή(tabu list) που καταγράφει πρόσφατες μετακινήσεις (π.χ. ανταλλαγές ακμών) που απαγορεύονται για ένα αριθμό βημάτων.
- Έχουμε το κριτήριο προσδοκίας που επιτρέπει τη παραβίαση του ταμπού αν η νέα λύση είναι καλύτερη από την καλύτερη γνωστή.

- Τέλος, ορίζουμε ένα κριτήριο τερματισμού. Π.χ. αριθμός επαναλήψεων χωρίς βελτίωση.

Στην περίπτωση του TSP, οι κινήσεις που καταγράφονται στην tabu list είναι συνήθως ανταλλαγές ακμών ή θέσεων μεταξύ πόλεων. Για παράδειγμα, η αφαίρεση και αντικατάσταση μιας ακμής σε λύση 2-opt μπορεί να θεωρηθεί ταμπού για μερικά βήματα. Το Tabu Search λειτουργεί αποτελεσματικά για μεσαίου και μεγάλου μεγέθους προβλήματα TSP, ειδικά όταν συνδυάζεται με ισχυρούς τοπικούς βελτιωτές όπως οι 3-opt ή Lin-Kernighan.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Μπορεί να διαφύγει από τοπικά ελάχιστα.
- Εκμεταλλεύεται πληροφορία από το παρελθόν (memory-based search).
- Ικανός να εξερευνήσει μεγάλα τμήματα του χώρου λύσεων.

Μειονεκτήματα:

- Απαιτεί προσεκτικό σχεδιασμό της tabu list και του μήκους της.
- Η επιλογή των aspiration criteria επηρεάζει την ποιότητα των λύσεων.
- Μπορεί να έχει υψηλή υπολογιστική απαίτηση σε μεγάλες περιπτώσεις.

Στην περίπτωση του TSP, οι κινήσεις που καταγράφονται στην tabu list είναι συνήθως ανταλλαγές ακμών ή θέσεων μεταξύ πόλεων. Για παράδειγμα, η αφαίρεση και αντικατάσταση μιας ακμής σε λύση 2-opt μπορεί να θεωρηθεί ταμπού για μερικά βήματα. Το Tabu Search λειτουργεί αποτελεσματικά για μεσαίου και μεγάλου μεγέθους προβλήματα TSP, ειδικά όταν συνδυάζεται με ισχυρούς τοπικούς βελτιωτές όπως οι 3-opt ή Lin-Kernighan.

3.2.3.3 Γενετικοί Αλγόριθμοι(Genetic Algorithms)

Οι Γενετικοί Αλγόριθμοι (GAs) αποτελούν μία ισχυρή κλάση μεταευρετικών μεθόδων, εμπνευσμένων από τη φυσική εξέλιξη και τη φυσική επιλογή των οργανισμών, όπως αυτές περιγράφονται από τη θεωρία του Δαρβίνου. Βασίζονται στις αρχές της αναπαραγωγής, της μετάλλαξης και της επιλογής, για να εξελίξουν πληθυσμούς πιθανών λύσεων σε δύσκολα προβλήματα βελτιστοποίησης όπως το TSP. Ο Γενετικός Αλγόριθμος λειτουργεί με έναν πληθυσμό λύσεων, κάθε μία από τις οποίες αναπαριστά έναν πιθανό γύρο στο TSP. Σε κάθε γενιά, δημιουργούνται νέες λύσεις μέσω γονιδιακών τελεστών (crossover = διασταύρωση και mutation = μετάλλαξη), ενώ οι καλύτερες λύσεις επιλέγονται για να συνεχίσουν στην επόμενη γενιά. Η επιτυχία ενός ΓΑ εξαρτάται σε μεγάλο βαθμό από την ποιότητα των τελεστών crossover και mutation, καθώς και από την ισορροπία μεταξύ εξερεύνησης και εκμετάλλευσης.

Βήματα του Αλγορίθμου:

- Αρχικά, αναπαριστούμε κάθε λύση (χρωμόσωμα) σαν μία ακολουθία πόλεων (π.χ. [3, 5, 1, 2, 4]).
- Έχουμε ένα σύνολο από υποψήφιες λύσεις.

- Εισάγουμε μία συνάρτηση καταλληλότητας με τη συνολική απόσταση της διαδρομής. Όσο μικρότερη η απόσταση, τόσο καλύτερη η λύση.
- Στη συνέχεια, γίνεται η επιλογή λύσεων για αναπαραγωγή(π.χ. roulette wheel, tournament).
- Χρησιμοποιούμε τους γονιδιακούς τελεστές(διασταύρωση και μετάλλαξη).
- Τέλος, ορίζουμε ένα κριτήριο τερματισμού. Π.χ. σταθερότητα στη βέλτιστη λύση.

Πλεονεκτήματα-μειονεκτήματα αλγορίθμου:

Πλεονεκτήματα:

- Εύκολα παραλληλοποιήσιμος.
- Εξερευνά μεγάλο μέρος του χώρου λύσεων.
- Δεν εξαρτάται από τις ιδιότητες του προβλήματος (π.χ. παραγώγους).

Μειονεκτήματα:

- Μπορεί να απαιτεί πολλούς πόρους (π.χ. πληθυσμούς, γενιές).
- Χρειάζεται προσεκτικός σχεδιασμός τελεστών και παραμέτρων.
- Δεν εγγυάται το βέλτιστο – εξαρτάται από την ποιότητα της αναπαραγωγής.

Οι Γενετικοί Αλγόριθμοι χρησιμοποιούνται ευρέως για την επίλυση του TSP, ιδίως όταν συνδυάζονται με τοπικές βελτιωτικές τεχνικές, όπως 2-opt, για τη βελτίωση των απογόνων σε κάθε γενιά (hybrid GAs). Στην πράξη, έχουν δώσει ποιοτικά αποτελέσματα σε προβλήματα με δεκάδες έως και χιλιάδες πόλεις.

3.3 Συγκριτική αξιολόγηση αλγορίθμων

Τα κύρια κριτήρια που χρησιμοποιούνται για να αξιολογηθεί ένας αλγόριθμος επίλυσης TSP είναι ο χρόνος και η ακρίβεια της επίλυσης, οι επαναλήψεις της μεθόδου που χρησιμοποιείται καθώς και η πολυπλοκότητα του προβλήματος.

Χρόνος Επίλυσης: Υπάρχουν οι ευρετικοί αλγόριθμοι που έχουν τη δυνατότητα να βρίσκουν λύσεις μέσα σε ελάχιστα δευτερόλεπτα, αλλά κατά προσέγγιση στο ολικό βέλτιστο. Από την άλλη μεριά, υπάρχουν οι ακριβείς που εντοπίζουν τη βέλτιστη λύση αλλά χρειάζονται πολύ περισσότερο χρόνο. Έτσι, έχει παγιωθεί η έννοια ότι ο χρόνος που χρειάζεται ένας αλγόριθμος για να επιλύσει ένα συγκεκριμένο TSP, καθορίζει και την αξία του έναντι άλλων αντίστοιχων αλγορίθμων.

Ακρίβεια Επίλυσης: Βασικό κριτήριο που χρησιμοποιείται κυρίως σε προβήματα της βιβλιοθήκης TSPLIB. Οι ερευνητές έχοντας στα χέρια τους τη βέλτιστη λύση, μπορούν να προσδιορίσουν πόσο μακριά ή κοντά βρίσκεται η λύση που έχει δώσει ο εκάστοτε αλγόριθμος που έχει χρησιμοποιηθεί.

Επαναλήψεις μεθόδου: Ανάλογα με το είδος του αλγορίθμου που έχει επιλεχθεί, απαιτούνται και οι ανάλογες επαναλήψεις για να εκτελεστεί σωστά. Υπάρχουν αλγόριθμοι που χρησιμοποιούν μικρό αριθμό επαναλήψεων και οδηγούν σε μία τελική λύση καθώς και άλλοι που χρειάζονται μεγαλύτερο πλήθος.

Πολυπλοκότητα προβλήματος: Ανάλογα με το πλήθος των κόμβων που χρησιμοποιούνται, προσδιορίζεται και ο χρόνος που απαιτείται για να φθάσει ο αλγόριθμος στην επίλυση του προβλήματος. Έτσι, καθορίζεται και η πολυπλοκότητα του προβλήματος.

Κατηγορία	**Αλγόριθμοι**	**Πλεονεκτήματα**	**Μειονεκτήματα**	**Καταλληλότητα**
Ακριβείς	Branch and Bound, Brute and Force, Cutting Plane, Held-Karp.	Εγγυώνται τη βέλτιστη λύση, αλλά περιλαμβάνουν αρκετή θεωρητική πληρότητα.	Επειδή έχουν εκθετική πολυπλοκότητα, πρακτικά ακατάλληλοι για μεγάλος εύρος Ν(πόλεων).	Μικρά προβλήματα(N < 20-30).
Απλοί Ευρετικοί	Greedy, Nearest Neighbor.	Πολύ γρήγοροι, εύκολοι στην υλοποίηση.	Χαμηλή ποιότητα λύσης, παγιδεύονται σε τοπικά ελάχιστα.	Μικρά προβλήματα ή ως αρχικές λύσεις σε μεσαίου – μεγάλου μεγέθους.
Βελτιωτικοί Ευρετικοί	2-opt, 3-opt, k-opt, Lin-Kernighan, Lin-Kernighan-Helsgaun	Παράγουν λύσεις πολύ κοντά στο ολικό βέλτιστο και έχουν καλή ισορροπία ποιότητας/χρόνου.	Εξαρτώνται από την αρχική λύση, έχουν αυξημένη πολυπλοκότητα με μεγάλα k.	Μεσαία έως πολύ μεγάλα προβλήματα.
Μεταευρετικοί	Genetic Algorithms, Simulated Annealing, Tabu Search	Εξερευνούν μεγάλο χώρο λύσεων, αποφεύγουν τοπικά ελάχιστα, κατάλληλες για παράλληλη επεξεργασία.	Υψηλή παραμετροποίηση και χρόνος σύγκλισης.	Μεγάλα και πολύπλοκα προβλήματα.

Πίνακας 3.3: Σύγκριση Αλγορίθμων

Από τη συγκριτική παρουσίαση προκύπτει ότι η επιλογή του κατάλληλου αλγορίθμου για την επίλυση του TSP εξαρτάται σε μεγάλο βαθμό από το μέγεθος και τις απαιτήσεις του εκάστοτε προβλήματος. Οι **ακριβείς αλγόριθμοι** έχουν θεωρητική σημασία, καθώς προσφέρουν την εγγύηση εύρεσης της βέλτιστης λύσης· ωστόσο η εκθετική πολυπλοκότητά τους τους καθιστά πρακτικά ακατάλληλους για προβλήματα μεγαλύτερου μεγέθους.

Οι **ευρετικοί αλγόριθμοι** αποτελούν μια εξαιρετικά γρήγορη και απλή προσέγγιση, παρέχοντας όμως λύσεις περιορισμένης ποιότητας. Συνεπώς, συχνά χρησιμοποιούνται ως αφετηρία για πιο σύνθετες μεθόδους. Οι **βελτιωτικοί ευρετικοί** (όπως οι 2-opt, 3-opt και Lin-Kernighan) προσφέρουν έναν

αποτελεσματικό συμβιβασμό, καθώς με σχετικά χαμηλό υπολογιστικό κόστος μπορούν να παραγάγουν λύσεις πολύ κοντά στο βέλτιστο.

Τέλος, οι **μεταευρετικοί αλγόριθμοι** ξεχωρίζουν σε μεγάλα και πολύπλοκα προβλήματα, χάρη στη δυνατότητά τους να εξερευνούν εκτενέστερα τον χώρο των λύσεων και να αποφεύγουν την παγίδευση σε τοπικά ελάχιστα. Ωστόσο, απαιτούν πιο περίπλοκη υλοποίηση, λεπτομερή ρύθμιση παραμέτρων και συνήθως μεγαλύτερο χρόνο εκτέλεσης.

Συνολικά, η βέλτιστη στρατηγική στις περισσότερες πραγματικές εφαρμογές βασίζεται σε υβριδικές προσεγγίσεις, οι οποίες συνδυάζουν την ταχύτητα των ευρετικών με τη βελτιωτική ισχύ των τοπικών αναζητήσεων ή των μεταευρετικών τεχνικών.

Κεφάλαιο 4ο: Εφαρμογή της μεθόδου Branch and Bound στο μεσαίου μεγέθους πρόβλημα

Στο παρόν Κεφάλαιο, παρουσιάζεται η εφαρμογή ενός αλγόριθμου Branch and Bound σε ένα μεσαίου μεγέθους πραγματικό πρόβλημα Ταξιδεύοντος Πωλητή. Στη ενότητα 4.1, περιγράφονται τα δεδομένα του προβλήματος, στην Ενότητα 4.2 παρουσιάζεται ο αλγόριθμος Branch and Bound που χρησιμοποιείται και στην Ενότητα 4.3 παρουσιάζεται η εφαρμογή του αλγόριθμου.

4.1 Τα δεδομένα του προβλήματος

Η ΑΡΧΕΙΟΘΗΚΗ Α.Ε. από εδώ και στο εξής ΑΡΧΕΙΟΘΗΚΗ, είναι μία Εταιρεία που ασχολείται με τη Φύλαξη και Διαχείριση του φυσικού αρχείου για επιχειρήσεις και οργανισμούς κάθε μεγέθους. Με μακρά ιστορία από το 2006 όπου ιδρύθηκε στη περιοχή της Νέας Ιωνίας Αττικής, η ΑΡΧΕΙΟΘΗΚΗ παρέχει ολοκληρωμένες λύσεις Διαχείρισης της πληροφορίας χρησιμοποιώντας τις πιο βέλτιστες διεθνείς πρακτικές. Από τη φύλαξη του Αρχείου στα ειδικά διαμορφωμένα Κέντρα που διαθέτει, μέχρι τη ψηφιοποίηση με τα πιο προηγμένα τεχνολογικά μέσα, η Αρχειοθήκη εγγυάται ότι θα καλύψει τις όποιες ανάγκες προκύψουν και θα οδηγήσει στην όσο το δυνατόν πιο ομαλή μετάβαση στο νέο Ψηφιακό Κόσμο.

Η εταιρεία αναλαμβάνει το έργο της διαχείρισης μεγάλου όγκου εγγράφων σε διάφορους τομείς (δικηγορικές εταιρείες, τράπεζες, νοσοκομεία, κλπ.), όπου η ταχύτητα, η ασφάλεια και η αποδοτικότητα στις μετακινήσεις είναι θεμελιώδους σημασίας. Διαθέτει δίκτυο αποθηκευτικών μονάδων και γραφείων σε διάφορες περιοχές και για να εξυπηρετήσει καλύτερα τους πελάτες της, πρέπει να οργανώσει τις διαδρομές των υπαλλήλων της που αναλαμβάνουν την παράδοση και παραλαβή αρχείων. Αυτό περιλαμβάνει την επίσκεψη σε 10 διαφορετικά σημεία αποθήκευσης και διαχείρισης αρχείων, με στόχο την ελαχιστοποίηση του συνολικού χρόνου ή της απόστασης που θα διανύσει κάθε υπάλληλος.

Το αρχικό στάδιο της διαδικασίας που χρησιμοποιεί η Εταιρεία, αφορά τον εγκιβωτισμό και τη μεταφορά των αρχείων στις εγκαταστάσεις της. Μία από τις προκλήσεις της διαδικασίας αυτής είναι η εύρεση της βέλτιστης διαδρομής και κατ΄ επέκταση η εξοικονόμηση χρόνου και πόρων. Στη προκειμένη περίπτωση, το Traveling Salesman Problem (TSP) αποτελεί το κλειδί για την επίλυση αυτού του προβλήματος, καθώς η εταιρεία επιθυμεί να διασφαλίσει ότι κάθε υπάλληλος θα επισκεφτεί όλα τα σημεία (ή περιοχές) με τον πιο σύντομο δυνατό τρόπο, και θα επιστρέψει στο σημείο εκκίνησης χωρίς να διανύσει επιπλέον ή άσκοπη απόσταση.

Το TSP είναι ένα κλασικό πρόβλημα στα μαθηματικά και την επιστήμη των υπολογιστών, που αφορά την εύρεση της πιο σύντομης διαδρομής για έναν ταξιδιώτη που πρέπει να επισκεφτεί όλα τα σημεία ακριβώς μία φορά και να επιστρέψει στο σημείο εκκίνησης. Στην περίπτωση της ΑΡΧΕΙΟΘΗΚΗΣ, τα σημεία αυτά μπορεί να είναι διαφορετικές αποθήκες αρχείων, γραφεία ή ακόμα και περιοχές εξυπηρέτησης πελατών, που απαιτούν τακτική επίσκεψη για τη διαχείριση των αρχείων.

Η επίλυση του TSP εξασφαλίζει ότι οι υπάλληλοι της εταιρείας θα εκτελούν τις παραδόσεις τους με τον πιο αποτελεσματικό τρόπο, μειώνοντας τον χρόνο μετακίνησης, το κόστος μεταφοράς και την ενεργειακή κατανάλωση. Επιπλέον, το πρόβλημα του TSP μπορεί να είναι χρήσιμο και για τον προγραμματισμό των εξόδων συντήρησης, καθώς η βέλτιστη διαδρομή μπορεί να μειώσει την ανάγκη για συχνές διορθώσεις ή αναπρογραμματισμούς διαδρομών.

Η ΑΡΧΕΙΟΘΗΚΗ χρησιμοποιεί προηγμένες τεχνολογίες και αλγορίθμους βελτιστοποίησης για να υπολογίσει την πιο αποδοτική διαδρομή για κάθε υπάλληλο. Οι εργαζόμενοι, εφοδιασμένοι με GPS και σύστημα διαχείρισης στόλου, ακολουθούν τις βέλτιστες διαδρομές, ελαχιστοποιώντας τον χρόνο και το κόστος κάθε αποστολής. Αυτός ο τρόπος οργάνωσης και διαχείρισης του έργου της αποθήκευσης και μεταφοράς αρχείων όχι μόνο βελτιώνει την αποδοτικότητα αλλά ενισχύει και την ικανοποίηση των πελατών, καθώς οι υπηρεσίες παραδίδονται γρήγορα και με ακρίβεια.

Με τη χρήση του TSP για την οργάνωση των διαδρομών της, η εταιρεία συνεχίζει να διατηρεί την ανταγωνιστικότητα της, ενώ παράλληλα μειώνει το κόστος, τον χρόνο και τις περιβαλλοντικές επιπτώσεις των καθημερινών της λειτουργιών. Με αυτές τις προσθήκες, η ΑΡΧΕΙΟΘΗΚΗ γίνεται μια εταιρεία που δεν προσφέρει μόνο εξαιρετικές υπηρεσίες, αλλά είναι και πρωτοπόρος στην τεχνολογική καινοτομία και περιβαλλοντική υπευθυνότητα, κάτι που προσδίδει αξία και στο ίδιο το πρόβλημα του TSP και στη λύση του.

4.2 Ο αλγόριθμος Branch and Bound: Η μέθοδος του αποκλεισμού υπο-περιοδειών

Υπάρχουν αρκετές Branch and Bound προσεγγίσεις που έχουν αναπτυχθεί για την επίλυση προβλημάτων TSP. Η μέθοδος που εφαρμόζεται στο παρόν Κεφάλαιο ονομάζεται "Μέθοδος Αποκλεισμού Υπο-περιοδειών" (Method of Excluded Subtours) [6].

Η προσέγγιση αυτή περιλαμβάνει μία κύρια λίστα προβλημάτων ανάθεσης. Στην πρώτη επανάληψη, η κύρια λίστα περιλαμβάνει ένα πρόβλημα : το αρχικό πρόβλημα TSP, το οποίο θα επιλυθεί ως πρόβλημα ανάθεσης. Το πρόβλημα αυτό ονομάζεται Πρόβλημα 1(t = 1).

Σε κάθε επανάληψη του αλγόριθμου επιλύεται ένα πρόβλημα ανάθεσης με την εφαρμογή του Ουγγρικού Αλγόριθμου. Στην αρχή κάθε επανάληψης i, υπάρχει ένα ανώτατο φράγμα (c^i) για το κόστος της βέλτιστης λύσης του προβλήματος ανάθεσης. Ένας τρόπος με τον οποίο θα μπορούσε να υπολογιστεί το ανώτατο φράγμα για την πρώτη επανάληψη είναι ο εξής :

$c^i = c_{12} + c_{23} + c_{34} + \ldots + c_{n1}$, όπου n το πλήθος των «πόλεων» και 1-2-3-…-n-1 μία εφικτή λύση του προβλήματος TSP.

Μία περιοδεία (tour) αποτελεί μία εφικτή λύση ενός πρόβλημα TSP.

Μία υπο-περιοδεία (subtour) αποτελεί ένα κύκλο που περιλαμβάνει λιγότερες από n «πόλεις».

Στη συνέχεια, περιγράφεται η διαδικασία που ακολουθείται κατά την i επανάληψη του αλγόριθμού :

Βήμα 1 : Αν η κύρια λίστα είναι κενή, ο αλγόριθμος τερματίζεται. Διαφορετικά, επιλέγεται ένα πρόβλημα από τη κύρια λίστα. Ακολουθείται η διαδικασία LIFO (Last In First Out) για την επιλογή του επόμενου προβλήματος προς επίλυση, δηλαδή, επιλέγεται εκείνο το πρόβλημα που έχει δημιουργηθεί πιο πρόσφατα.

Βήμα 2 : Επιλύεται το πρόβλημα ανάθεσης που επιλέχθηκε στο Βήμα 1. Αν το κόστος της βέλτιστης λύσης (z) είναι μεγαλύτερο ή ίσο με το c^i , θέσε $c^{i+1} = c^i$ και επέστρεψε στο Βήμα 1. Διαφορετικά, προχώρησε στο Βήμα 3.

Βήμα 3 : Αν η βέλτιστη λύση του προβλήματος ανάθεσης είναι περιοδεία, τότε κατέγραψε την, θέσε c^{i+1} ίσο με το κόστος που αντιστοιχεί στη βέλτιστη λύση και επέστρεψε στο Βήμα 1. Διαφορετικά, προχώρησε στο Βήμα 4.

Βήμα 4 : Επίλεξε από τη βέλτιστη λύση του επιλεγμένου προβλήματος ανάθεσης μία υπο-περιοδεία που περιλαμβάνει το μικρότερο αριθμό πόλεων. Για κάθε $x_{i,j} = 1$ της υπο-περιοδείας, πρόσθεσε ένα πρόβλημα στην κύρια λίστα και θέσε $c_{i,j} = M$ (ισοδύναμα $x_{i,j} = 0$), όπου M ένας πολύ μεγάλος θετικός αριθμός. Όλα τα υπόλοιπα κόστη παραμένουν τα ίδια με εκείνα του επιλεγμένου προβλήματος στο Βήμα 1. Θέσε $c^{i+1} = c^i$ και επέστρεψε στο Βήμα 1.

Συνοπτικά, η διαδικασία περιλαμβάνει την επίλυση προβλημάτων ανάθεσης. Αν η λύση του προβλήματος είναι μία περιοδεία, τότε η λύση αυτή αποτελεί μία εφικτή λύση του προβλήματος TSP. Δημιουργούνται νέα προβλήματα ανάθεσης με διακλάδωση, η οποία αποκλείει μία υπο-περιοδεία. Κάθε πρόβλημα του οποίου η βέλτιστη λύση αντιστοιχεί σε κόστος μεγαλύτερο του τρέχοντος φράγματος (της καλύτερης εφικτής λύσης που έχει βρεθεί προηγούμενα) αφαιρείται από τη διαδικασία.

4.3 Εφαρμογή του αλγόριθμου στο πρόβλημα της ΑΡΧΕΙΟΘΗΚΗΣ

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1.Νέα Ιωνία	2000	13	21	58	354	417	317	502	488	492
2.Περιστέρι	13	2000	22	71	356	419	319	504	490	493
3.Ασπρόπυργος	21	22	2000	69	341	403	328	513	498	502
4.Σχηματάρι	58	71	69	2000	403	380	262	447	433	436
5.Πρέβεζα	354	356	341	403	2000	100	212	331	342	345
6.Ιωάννινα	417	419	403	380	100	2000	122	245	251	255
7.Τρίκαλα	317	319	328	262	212	122	2000	223	209	212
8.Γιαννιτσά	502	504	513	447	331	245	223	2000	46	41
9.Καλοχώρι	488	490	498	433	342	251	209	46	2000	8
10.Βι.ΠΕ.Θ. Σίνδου	492	493	502	436	345	255	212	41	8	2000

Πίνακας 4.3.1: Δεδομένα Προβλήματος

Στον Πίνακα (Πίνακας 4.3.1) παρουσιάζονται τα δεδομένα του προβλήματος TSP. Τα δεδομένα που απεικονίζονται παρακάτω αφορούν τις αποστάσεις (σε χιλιόμετρα) μεταξύ των 10 σημείων και έχουν εισαχθεί από την εφαρμογή Χάρτες της Google.

Η κύρια λίστα προβλημάτων περιλαμβάνει το πρόβλημα αυτό (Πρόβλημα 1), το οποίο και θα επιλυθεί ως πρόβλημα ανάθεσης με την εφαρμογή του Ουγγρικού αλγόριθμου.

Το άνω φράγμα υπολογίζεται $c^1 = c_{12} + c_{23} + c_{34} + \ldots + c_{10,1} = 13 + 22 + 69 + \ldots + 492 = 1498$.

Εφαρμογή Hungarian Method:

Εύρεση του μικρότερου στοιχείου κάθε γραμμής και αφαίρεσής του από τα υπόλοιπα.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**min**
1.Νέα Ιωνία	2000	13	21	58	354	417	317	502	488	492	**13**
2.Περιστέρι	13	2000	22	71	356	419	319	504	490	493	**13**
3.Ασπρόπυργος	21	22	2000	69	341	403	328	513	498	502	**21**
4.Σχηματάρι	58	71	69	2000	403	380	262	447	433	436	**58**
5.Πρέβεζα	354	356	341	403	2000	100	212	331	342	345	**100**
6.Ιωάννινα	417	419	403	380	100	2000	122	245	251	255	**100**
7.Τρίκαλα	317	319	328	262	212	122	2000	223	209	212	**122**
8.Γιαννιτσά	502	504	513	447	331	245	223	2000	46	41	**41**
9.Καλοχώρι	488	490	498	433	342	251	209	46	2000	8	**8**
10.Βl.ΠΕ.Θ. Σίνδου	492	493	502	436	345	255	212	41	8	2000	**8**

Πίνακας 4.3.2: Ουγγρικός Αλγόριθμος – Εύρεση μικρότερου στοιχείου κάθε γραμμής.

Επόμενο βήμα:

Εύρεση του μικρότερου στοιχείου κάθε στήλης και αφαίρεσής του από τα υπόλοιπα.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1.Νέα Ιωνία	1987	0	8	45	341	404	304	489	475	479
2.Περιστέρι	0	1987	9	58	343	406	306	491	477	480
3.Ασπρόπυργος	0	1	1979	48	320	382	307	492	477	481
4.Σχηματάρι	0	13	11	1942	345	322	204	389	375	378
5.Πρέβεζα	254	256	241	303	1900	0	112	231	242	245
6.Ιωάννινα	317	319	303	280	0	1900	22	145	151	155
7.Τρίκαλα	195	197	206	140	90	0	1878	101	87	90
8.Γιαννιτσά	461	463	472	406	290	204	182	1959	5	0
9.Καλοχώρι	480	482	490	425	334	243	201	38	1992	0
10.ΒΙ.ΠΕ.Θ. Σίνδου	484	485	494	428	337	247	204	33	0	1992
min	0	0	8	45	0	0	22	33	0	0

Πίνακας 4.3.3: Ουγγρικός Αλγόριθμος – Εύρεση μικρότερου στοιχείου κάθε στήλης.

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1987	0	0	0	341	404	282	456	475	479
2.	0	1987	1	13	343	406	284	458	477	480
3.	0	1	1971	3	320	382	285	459	477	481
4.	0	13	3	1897	345	322	182	356	375	378
5.	254	256	233	258	1900	0	90	198	242	245
6.	317	319	295	235	0	1900	0	112	151	155
7.	195	197	198	95	90	0	1856	68	87	90
8.	461	463	464	361	290	204	160	1926	5	0
9.	480	482	482	380	334	243	179	5	1992	0
10.	484	485	486	383	337	247	182	0	0	1992

Πίνακας 4.3.4: Ουγγρικός Αλγόριθμος – Μετά από αφαίρεση μικρότερου στοιχείου κάθε γραμμής/στήλης.

Επόμενο βήμα:

Με τον ελάχιστο αριθμό ευθειών έστω πλήθους r(οριζόντιων ή/και καθέτων), διαγράφουμε όλα τα μηδενικά στοιχεία του πίνακα.

Αν r = m, τότε ο πίνακας περιέχει την άριστη λύση.

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1987	0	0	0	341	404	282	456	475	479
2.	0	1987	1	13	343	406	284	458	477	480
3.	0	1	1971	3	320	382	285	459	477	481
4.	0	13	3	1897	345	322	182	356	375	378
5.	254	256	233	258	1900	0	90	198	242	245
6.	317	319	295	235	0	1900	0	112	151	155
7.	195	197	198	95	90	0	1856	68	87	90
8.	461	463	464	361	290	204	160	1926	5	0
9.	480	482	482	380	334	243	179	5	1992	0
10.	484	485	486	383	337	247	182	0	0	1992

Πίνακας 4.3.5: Ουγγρικός Αλγόριθμος – Διαγραφή μηδενικών στοιχείων πίνακα.

r = 6 < 10 = m. Ο πίνακας δεν περιέχει την άριστη λύση.

Επόμενο βήμα:

- Έστω κ το μικρότερο μη διαγραμμένο στοιχείο.

- Αφαίρεση του κ από κάθε μη διαγραμμένο στοιχείο και πρόσθεσή του σε κάθε στοιχείο που είναι στη διασταύρωση δύο ευθειών.
- Επιστροφή στο προηγούμενο βήμα.

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1988	0	0	0	341	405	282	456	475	480
2.	0	1987	0	12	342	406	283	457	476	480
3.	0	0	1970	2	319	382	284	458	476	481
4.	0	12	2	1896	344	322	181	355	374	378
5.	254	255	232	257	1899	0	89	197	241	245
6.	318	319	295	235	0	1901	0	112	151	156
7.	192	196	197	94	89	0	1855	67	86	90
8.	461	462	463	360	289	204	159	1925	4	0
9.	480	481	481	379	333	243	178	4	1991	0
10.	485	485	486	383	337	248	182	0	0	1993

Πίνακας 4.3.6: Εφαρμογή Ουγγρικού Αλγορίθμου

r = 8 < 10 = m

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1990	0	0	0	341	407	282	456	475	482
2.	2	1987	0	12	342	408	283	457	476	482
3.	2	0	1970	2	319	384	284	458	476	483
4.	0	10	0	1894	342	322	179	353	372	378
5.	254	253	230	255	1897	0	87	195	239	245
6.	318	319	295	235	0	1903	0	112	151	158
7.	192	194	195	92	87	0	1853	65	84	90
8.	461	460	461	358	287	204	157	1923	2	0
9.	480	479	479	377	331	243	176	2	1989	0
10.	487	485	486	383	337	250	182	0	0	1995

Πίνακας 4.3.7: Εφαρμογή Ουγγρικού Αλγορίθμου

r = 8 < 10 = m

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1990	0	0	0	341	409	282	456	475	484
2.	2	1987	0	12	342	410	283	457	476	484
3.	2	0	1970	2	319	386	284	458	474	485
4.	0	10	2	1894	342	324	179	353	372	380
5.	252	251	228	253	1895	0	85	193	237	245
6.	318	319	295	235	0	1905	0	112	151	160
7.	190	192	193	90	85	0	1851	63	82	90
8.	459	458	459	356	285	204	155	1921	0	0
9.	478	477	477	375	329	243	174	0	1987	0
10.	487	485	486	383	337	252	182	0	0	1997

Πίνακας 4.3.8: Εφαρμογή Ουγγρικού Αλγορίθμου

r = 9 < m = 10

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1990	0	0	0	341	472	282	456	475	484
2.	2	1987	0	12	342	473	283	457	476	484
3.	2	0	1970	2	319	449	284	458	474	485
4.	0	10	2	1894	342	387	179	353	372	380
5.	189	188	165	190	1832	0	22	130	174	182
6.	318	319	295	235	0	1968	0	112	151	160
7.	127	129	130	27	22	0	1788	0	19	27
8.	459	458	459	356	285	267	155	1921	0	0
9.	478	477	477	375	329	306	174	0	1987	0
10.	487	485	486	383	337	315	182	0	0	1997

Πίνακας 4.3.9: Εφαρμογή Ουγγρικού Αλγορίθμου

r = 9 < m = 10

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1990	0	0	0	341	494	282	478	497	506
2.	2	1987	0	12	342	495	283	479	498	506
3.	2	0	1970	2	319	471	284	480	496	507
4.	0	10	2	1894	342	409	179	375	394	402
5.	167	166	143	168	1810	0	0	130	174	182
6.	318	319	295	235	0	1990	0	112	151	160
7.	105	107	108	5	0	0	1766	0	19	27
8.	437	436	437	334	263	267	133	1921	0	0
9.	456	455	455	353	307	306	152	0	1987	0
10.	465	463	464	361	315	315	160	0	0	1997

Πίνακας 4.3.10: Εφαρμογή Ουγγρικού Αλγορίθμου

r = 10 = m

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
1.	1990	0	0	0	341	494	282	478	497	506
2.	2	1987	0	12	342	495	283	479	498	506
3.	2	0	1970	2	319	471	284	480	518	507
4.	0	10	2	1894	342	409	179	375	394	402
5.	167	166	143	168	1810	0	0	130	174	182
6.	318	319	295	235	0	1990	0	112	151	160
7.	105	107	108	5	0	0	1766	0	19	27
8.	437	436	437	334	263	267	133	1921	0	0
9.	456	455	455	353	307	306	152	0	1987	0
10.	465	463	464	361	315	315	160	0	0	1997

Πίνακας 4.3.11: Τελικός Πίνακας μετά από εφαρμογή του Ουγγρικού Αλγορίθμου.

Η βέλτιστη λύση του προβλήματος ανάθεσης είναι η εξής:

1->4

2->3

3->2

4->1

5->6

6->7

7->5

8->9

9->10

10->8

$z = c_{14} + c_{23} + c_{32} + c_{41} + c_{56} + c_{67} + c_{75} + c_{89} + c_{910} + c_{108} = 58 + 22 + 22 + 58 + 100 + 122 + 212 + 46 + 8 + 41 = 689.$

Η βέλτιστη λύση του προβλήματος ανάθεσης που βρέθηκε προηγουμένως υποδεικνύει τις τέσσερεις υπο-περιοδείες που εμφανίζονται στη ρίζα του παρακάτω δέντρου.

Επιλέγεται αυθαίρετα μία από τις μικρότερες υπο-περιοδείες : 1 -> 4 -> 1.

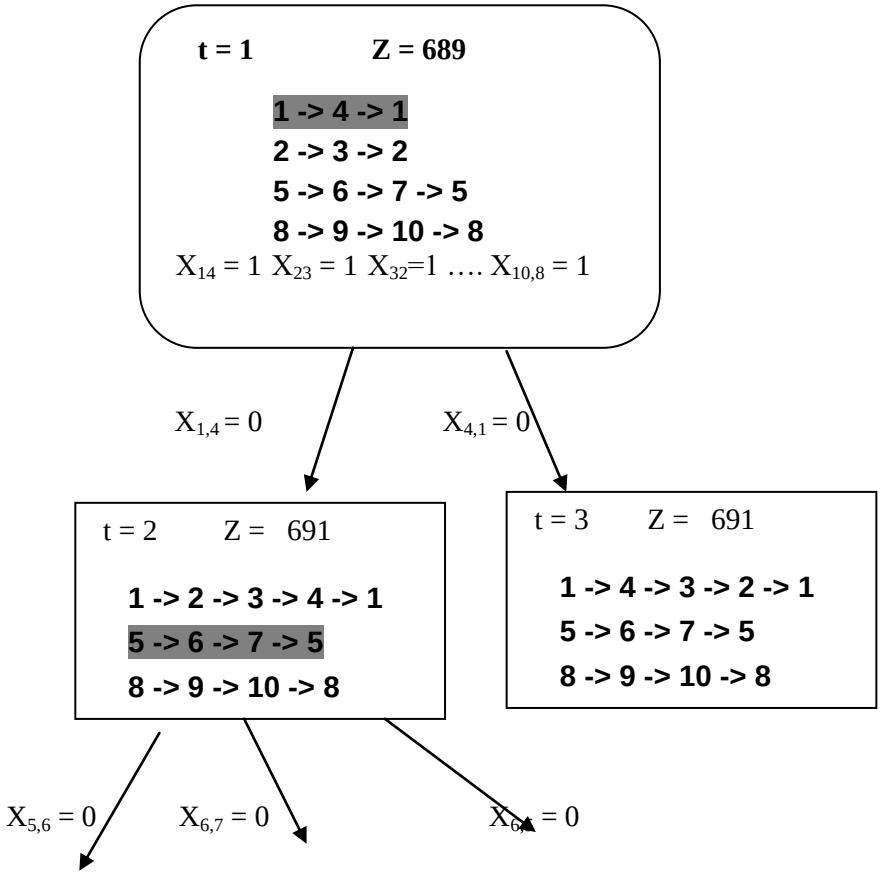
Δημιουργούνται δύο νέα προβλήματα τα οποία και προστίθενται στην κύρια λίστα:

Πρόβλημα 2: το Πρόβλημα 1 + c_{14} = M,

Πρόβλημα 3: το Πρόβλημα 1 + c_{41} = M.

Το Πρόβλημα 1 αφαιρείται από την κύρια λίστα.

$c^2 = c^1 = 1498.$



Σχήμα 4.3.1: Εφαρμογή αλγορίθμου για εύρεση αρχικής λύσης

Επιλέγεται αυθαίρετα μία από τις μικρότερες υπο-περιοδείες : 5 -> 6 -> 7 -> 5.

Δημιουργούνται τρία νέα προβλήματα τα οποία και προστίθενται στη κύρια λίστα:

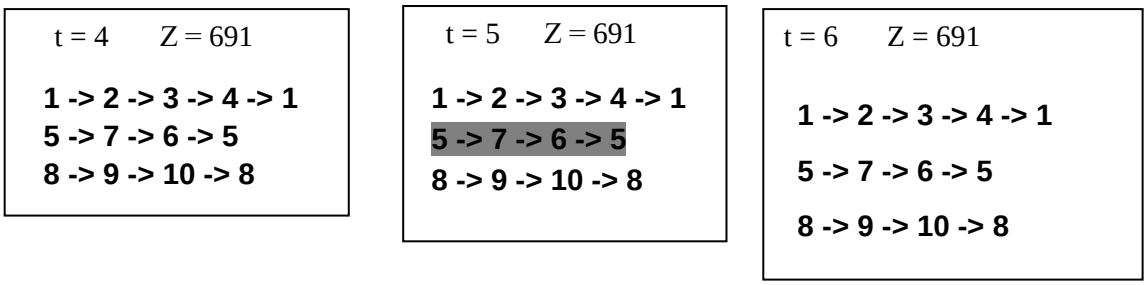
Πρόβλημα 4: το Πρόβλημα 2 + c_{56} = M,

Πρόβλημα 5: το Πρόβλημα 2 + c_{67} = M.

Πρόβλημα 6: το Πρόβλημα 2 + c_{75} = M.

Το Πρόβλημα 2 αφαιρείται από την κύρια λίστα.

$c^3 = c^2 = 1498$.



Σχήμα 4.3.2: Εφαρμογή αλγορίθμου για εύρεση αρχικής λύσης

Επιλέγεται αυθαίρετα μία από τις μικρότερες υπο-περιοδείες : 5 -> 7 -> 6 -> 5.

Δημιουργούνται τρία νέα προβλήματα τα οποία και προστίθενται στη κύρια λίστα:

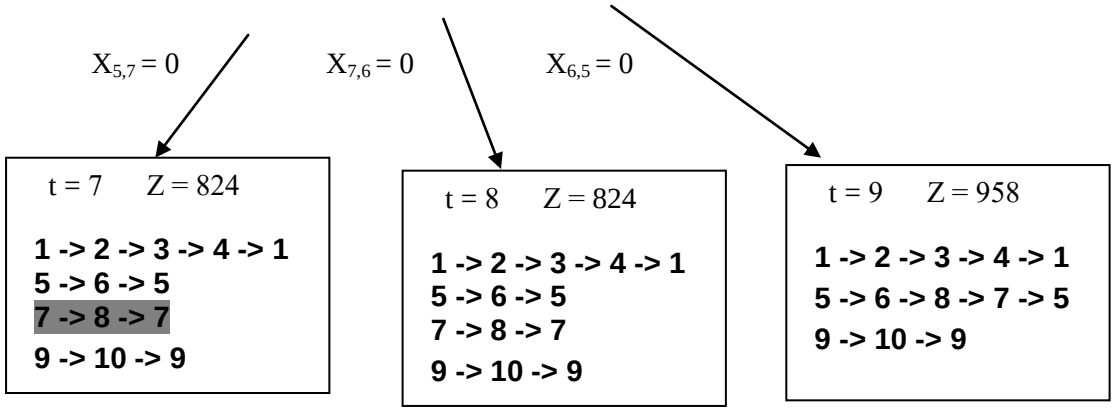
Πρόβλημα 7: το Πρόβλημα 5 + c_{57} = M,

Πρόβλημα 8: το Πρόβλημα 5 + c_{76} = M.

Πρόβλημα 9: το Πρόβλημα 5 + c_{65} = M.

Το Πρόβλημα 5 αφαιρείται από την κύρια λίστα.

$c^4 = c^3 = 1498$.



Σχήμα 4.3.3: Εφαρμογή αλγορίθμου για εύρεση αρχικής λύσης

Επιλέγεται αυθαίρετα μία από τις μικρότερες υπο-περιοδείες : 7 -> 8 -> 7.

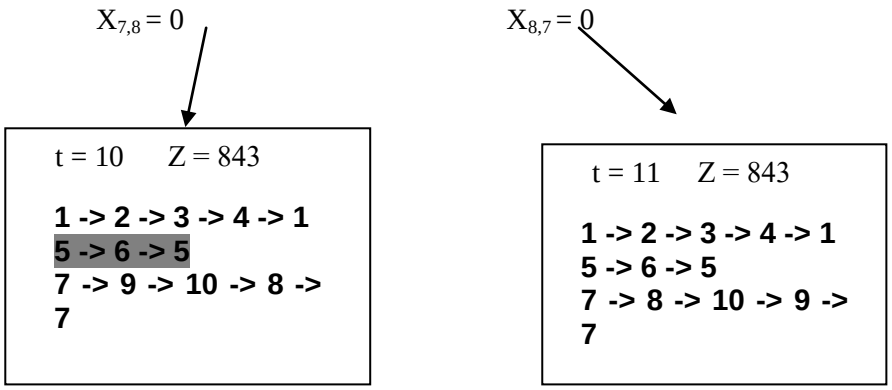
Δημιουργούνται δύο νέα προβλήματα τα οποία και προστίθενται στην κύρια λίστα:

Πρόβλημα 10: το Πρόβλημα 7 + c_{78} = M,

Πρόβλημα 11: το Πρόβλημα 7 + c_{87} = M.

Το Πρόβλημα 7 αφαιρείται από την κύρια λίστα.

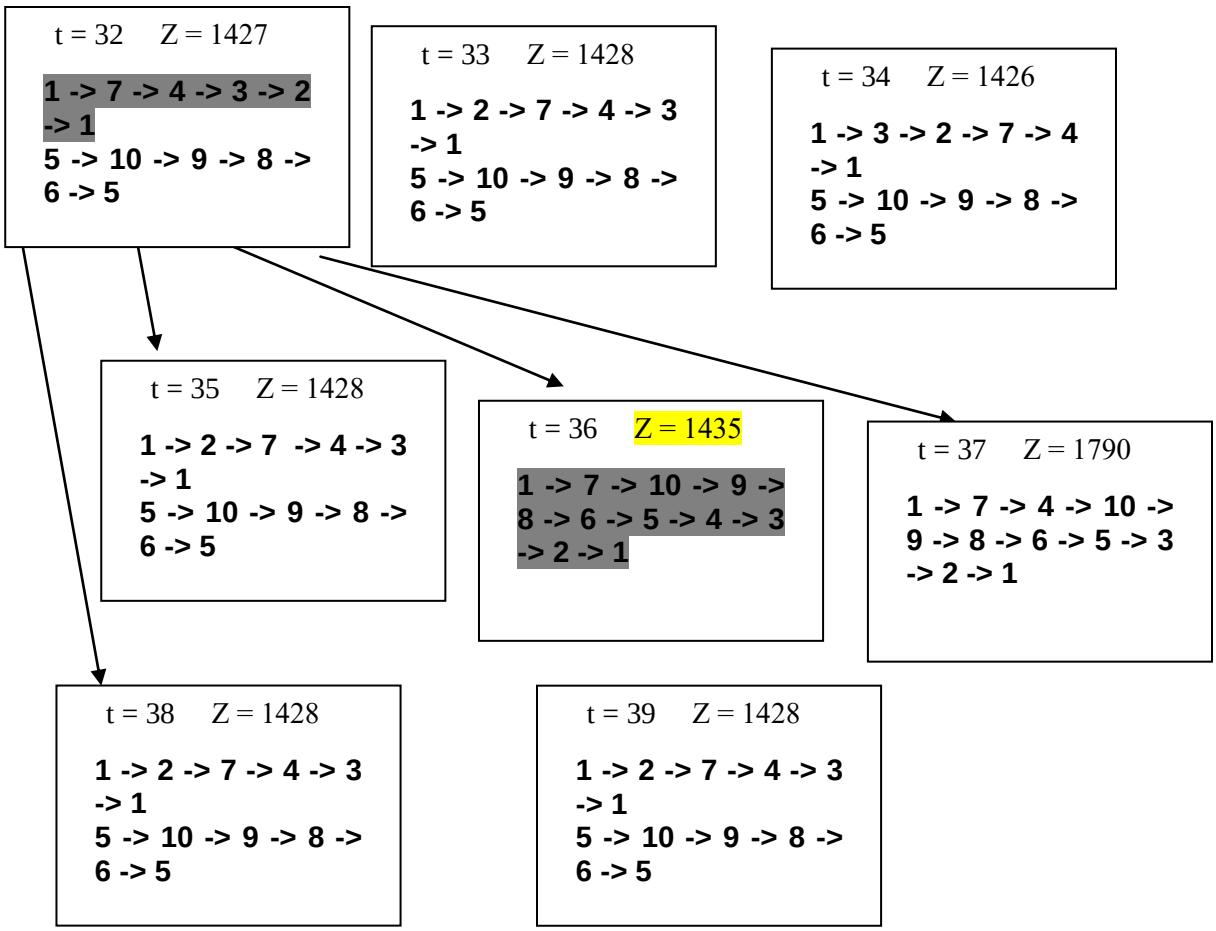
$c^5 = c^4 = 1498$.



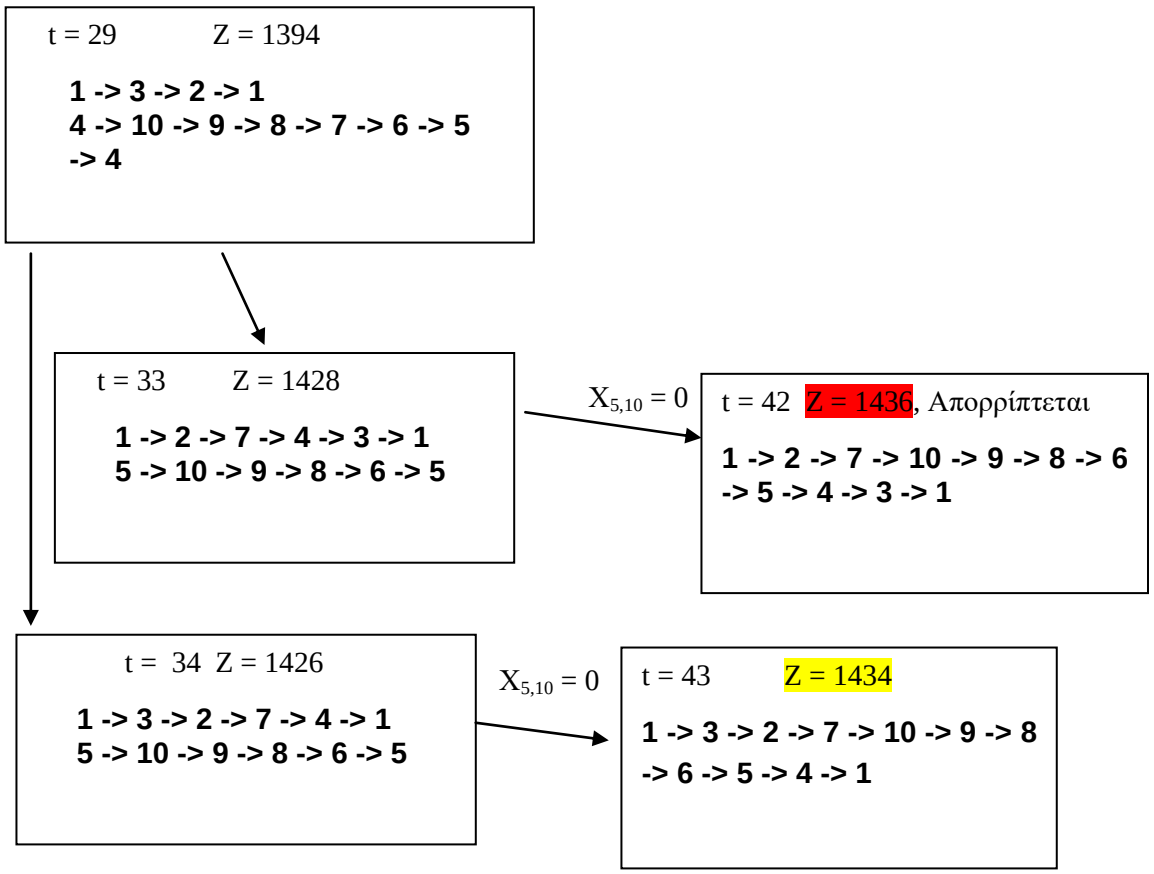
Σχήμα 4.3.4: Εφαρμογή αλγορίθμου για εύρεση αρχικής λύσης

Μετά από δεκατρείς επαναλήψεις του αλγορίθμου, φθάνουμε σε μία λύση του προβλήματος (Πρόβλημα 32) ανάθεσης, η οποία αποτελεί μία περιοδεία: **1 -> 7 -> 10 -> 9 -> 8 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1.** Συνεπώς, η λύση αυτή είναι η πρώτη εφικτή λύση του προβλήματος TSP που εντόπισε ο αλγόριθμος. Το κόστος της λύσης αυτής είναι 1435 και η τιμή αυτή ανατίθεται στο c^{13} . Όλες οι επαναλήψεις παρουσιάζονται στο Παράρτημα[].

Το επόμενο πρόβλημα που επιλύεται είναι ένα από τα προβλήματα που έχουν παραμείνει στη κύρια λίστα. Επιλέγεται κάθε φορά το Πρόβλημα εκείνο που έχει δημιουργηθεί πιο πρόσφατα. Δηλαδή, επιλύεται το Πρόβλημα 39, στη συνέχεια το Πρόβλημα 38, κ.ο.κ. Όπως φαίνεται στο παρακάτω σχήμα, το Πρόβλημα 35, το Πρόβλημα 38 και το Πρόβλημα 39 οδηγούν σε δύο υποπεριοδείες. Το Πρόβλημα 37 οδηγεί σε μία περιοδεία και συνεπώς αποτελεί εφικτή λύση του προβλήματος TSP. Όμως, το κόστος της λύσης αυτής είναι μεγαλύτερο από το κόστος της λύσης του Προβλήματος 36. Για το λόγο αυτό, αποκλείεται από περαιτέρω διερεύνηση.



Σχήμα 4.3.5: Εφαρμογή αλγορίθμου – Εύρεση αρχικής λύσης



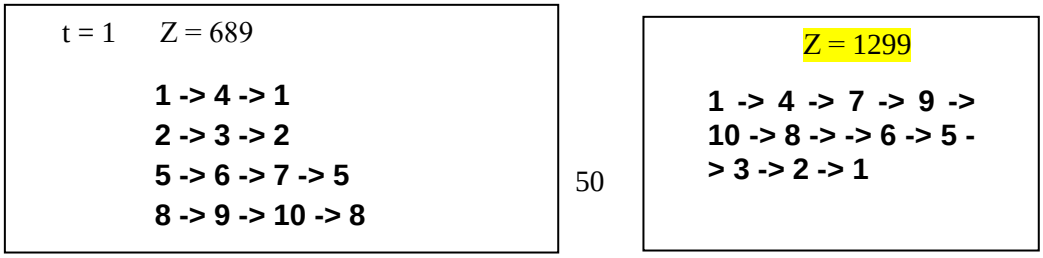
Σχήμα 4.3.6: Εφαρμογή αλγορίθμου – εύρεση βέλτιστης λύσης

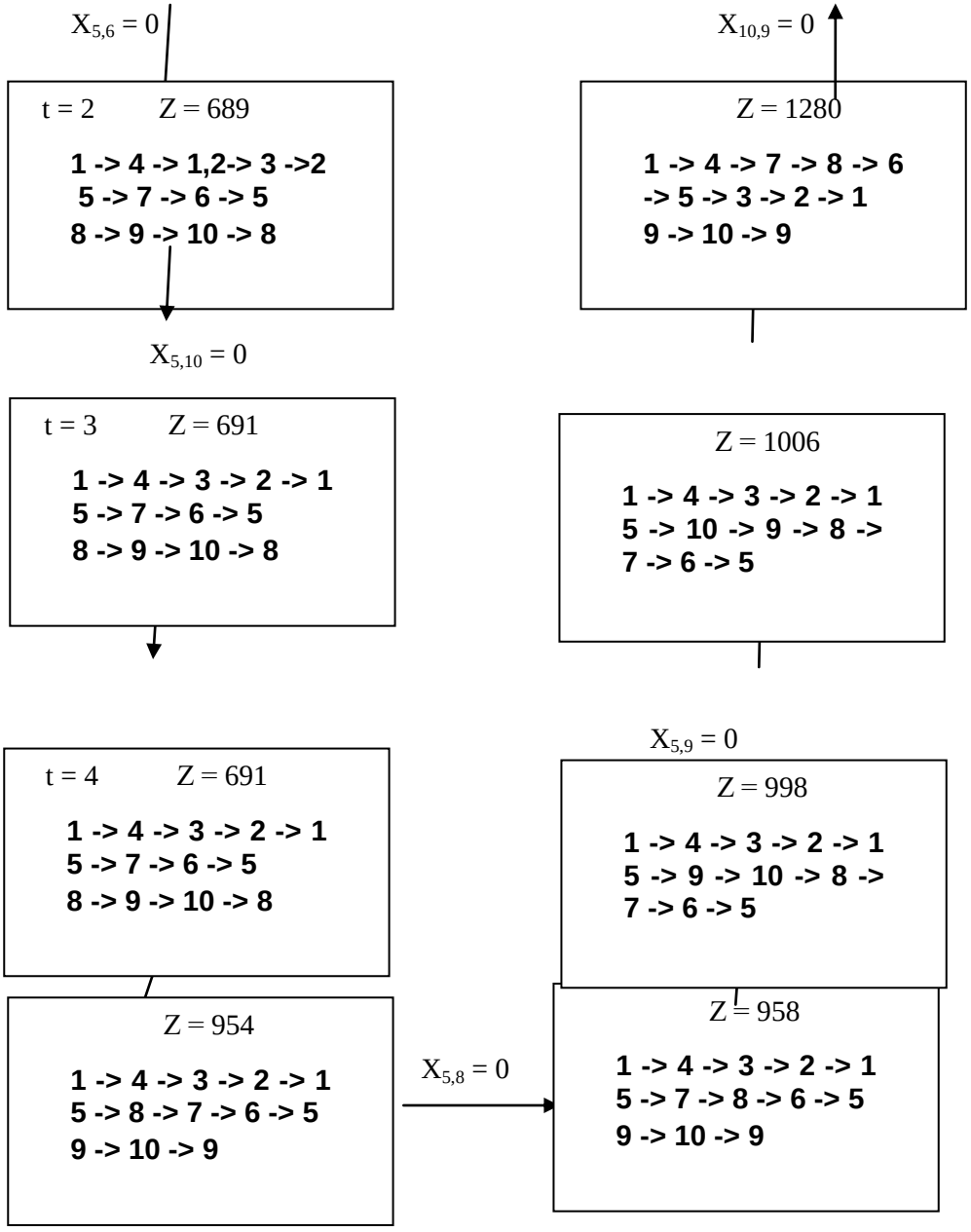
Στη συνέχεια, πηγαίνοντας προς τα πίσω, βρίσκουμε άλλη μία λύση.

Συγκεκριμένα, μετά από έντεκα επαναλήψεις του αλγορίθμου, φθάνουμε σε μία λύση του προβλήματος (Πρόβλημα 29) ανάθεσης, η οποία αποτελεί μία περιοδεία: **1** -> **3** -> **2** -> **7** -> **10** -> **9** -> **8** -> **6** -> **5** -> **4** -> **1.** Συνεπώς, η λύση αυτή είναι η δεύτερη εφικτή λύση του προβλήματος TSP που εντόπισε ο αλγόριθμος. Το κόστος της λύσης αυτής είναι 1434 και η τιμή αυτή ανατίθεται στο c^{11} .

Το επόμενο πρόβλημα που επιλύεται είναι ένα από τα προβλήματα που έχουν παραμείνει στη κύρια λίστα. Επιλέγεται κάθε φορά το Πρόβλημα εκείνο που έχει δημιουργηθεί πιο πρόσφατα. Δηλαδή, επιλύεται το Πρόβλημα 33. Όπως φαίνεται στο παρακάτω σχήμα, το Πρόβλημα 33 οδηγεί σε δύο υποπεριοδείες.

Αφού κάναμε όλες τις δυνατές επαναλήψεις του αλγορίθμου(βλέπε παράρτημα), φθάνουμε στην άριστη λύση του προβλήματος (Πρόβλημα 1) ανάθεσης, η οποία αποτελεί μία περιοδεία: **1** -> **4** -> **7** -> **9** -> **10** -> **8** -> **6** -> **5** -> **3** -> **2** -> **1.** Συνεπώς, η λύση αυτή είναι η τελική εφικτή λύση του προβλήματος TSP που εντόπισε ο αλγόριθμος. Το κόστος της λύσης αυτής είναι 1299 και η τιμή αυτή ανατίθεται στο c^{1} .





Σχήμα 4.3.7: Βέλτιστη λύση

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Η παρούσα εργασία είχε ως αντικείμενο τη μελέτη του προβλήματος του Ταξιδεύοντος Πωλητή (Traveling Salesman Problem – TSP), ενός από τα πιο κλασικά και ταυτόχρονα πιο δύσκολα προβλήματα της Υπολογιστικής Πολυπλοκότητας. Μετά από τη θεωρητική ανάλυση των αλγορίθμων που χρησιμοποιούνται αλλά και την υλοποίηση του πρακτικού μέρους, με εφαρμογή ενός αλγορίθμου σε μεσαίου μεγέθους πρόβλημα, παρακάτω ακολουθούν χρήσιμα συμπεράσματα και παρατηρήσεις που έχουν προκύψει. Τέλος, προτείνονται επιπλέον λειτουργίες οι οποίες θα μπορούσαν να βοηθήσουν στη βετλίωση της παρούσας μελέτης.

5.1 Κύρια Συμπεράσματα

Από τη μελέτη και την ανάλυση που προηγήθηκε, προκύπτουν τα ακόλουθα βασικά συμπεράσματα:

- Οι **ακριβείς αλγόριθμοι** (Brute Force, Branch and Bound, Cutting Planes) είναι σε θέση να εξασφαλίσουν τη βέλτιστη λύση, αλλά γίνονται πρακτικά ανεφάρμοστοι για μεγαλύτερα μεγέθη προβλήματος λόγω του εκρηκτικού ρυθμού αύξησης της πολυπλοκότητας.
- Οι **ευρετικοί αλγόριθμοι** (Nearest Neighbor, Greedy, 2-opt, 3-opt, Lin-Kernighan) παρέχουν γρήγορες και αρκετά καλές λύσεις, ιδανικές για μεσαίου μεγέθους προβλήματα.
- Οι **μεταευρετικοί αλγόριθμοι** (Simulated Annealing, Tabu Search, Γενετικοί Αλγόριθμοι, Lin-Kernighan-Helsgaun) συνδυάζουν την αποδοτικότητα με υψηλή ποιότητα λύσεων και αποδεικνύονται ιδιαίτερα αποτελεσματικοί σε μεγάλης κλίμακας προβλήματα.

5.2 Περιορισμοί της Μελέτης

Η εργασία αυτή εστίασε σε μια συγκεκριμένη εκδοχή του προβλήματος, χωρίς να ενσωματώνει πιο πολύπλοκους παράγοντες. Ενδεικτικά, δεν λήφθηκαν υπόψη:

- Δυναμικές συνθήκες κυκλοφορίας ή χρονικά παράθυρα επισκέψεων.
- Διαφορετικά κόστη σε σχέση με την ώρα ή το είδος του οχήματος.
- Πιθανές καθυστερήσεις λόγω κυκλοφορίας ή άλλων εξωτερικών παραγόντων.
- Επέκταση του προβλήματος σε περισσότερα από ένα οχήματα (όπως στο Vehicle Routing Problem – VRP).

5.3 Προτάσεις για Μελλοντική Έρευνα

Για τη βελτίωση και περαιτέρω εξέλιξη της παρούσας μελέτης, μπορούν να εξεταστούν οι ακόλουθες κατευθύνσεις:

- Μελέτη πιο σύνθετων παραλλαγών του TSP, όπως το TSP με χρονικά παράθυρα(Travelling Salesman Problem with Time Windows - TSPTW), ή το πρόβλημα δρομολόγησης στόλου (Vehicle Routing Problem - VRP).
- Υβριδικές προσεγγίσεις, που συνδυάζουν ακριβείς και ευρετικές τεχνικές, με στόχο την ταχύτερη εύρεση λύσεων υψηλής ποιότητας.
- Δυναμικό TSP, όπου οι συνθήκες αλλάζουν σε πραγματικό χρόνο (π.χ. κυκλοφορία, νέες παραγγελίες).

- Ανάπτυξη πρακτικού λογισμικού που θα μπορεί να χρησιμοποιηθεί από επιχειρήσεις για τη βελτιστοποίηση των δρομολογίων τους.
- Διεπιστημονικές εφαρμογές, όπως στη ρομποτική, στις τηλεπικοινωνίες και στη βιοπληροφορική, όπου παραλλαγές του TSP εμφανίζονται με διαφορετικές μορφές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (Eds.). (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley.

[2] Karp, R. M. (1972). *Reducibility among combinatorial problems*. In Complexity of Computer Computations (pp. 85–103). Springer.

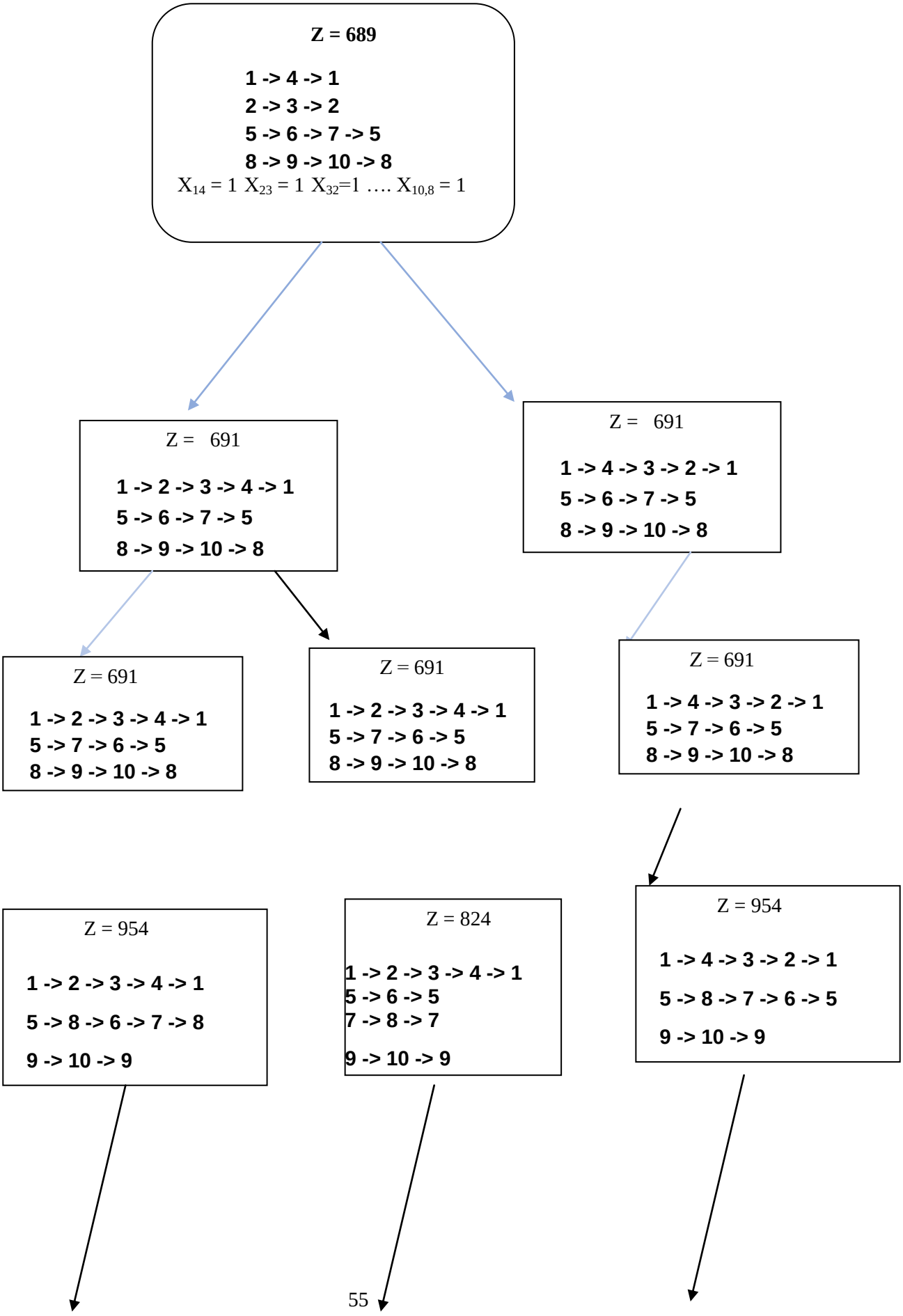
[3] Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

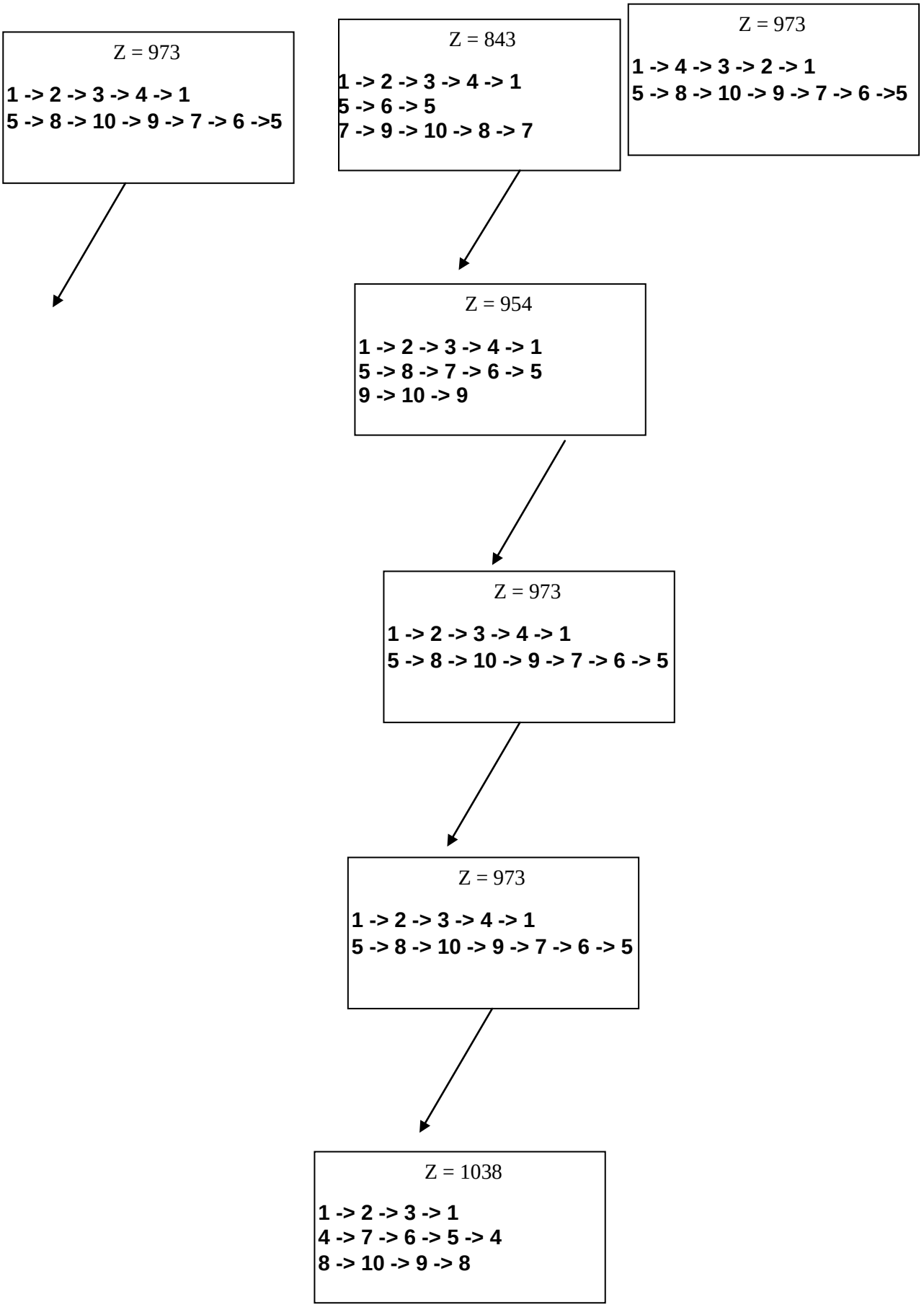
[4]Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3), 209–219.

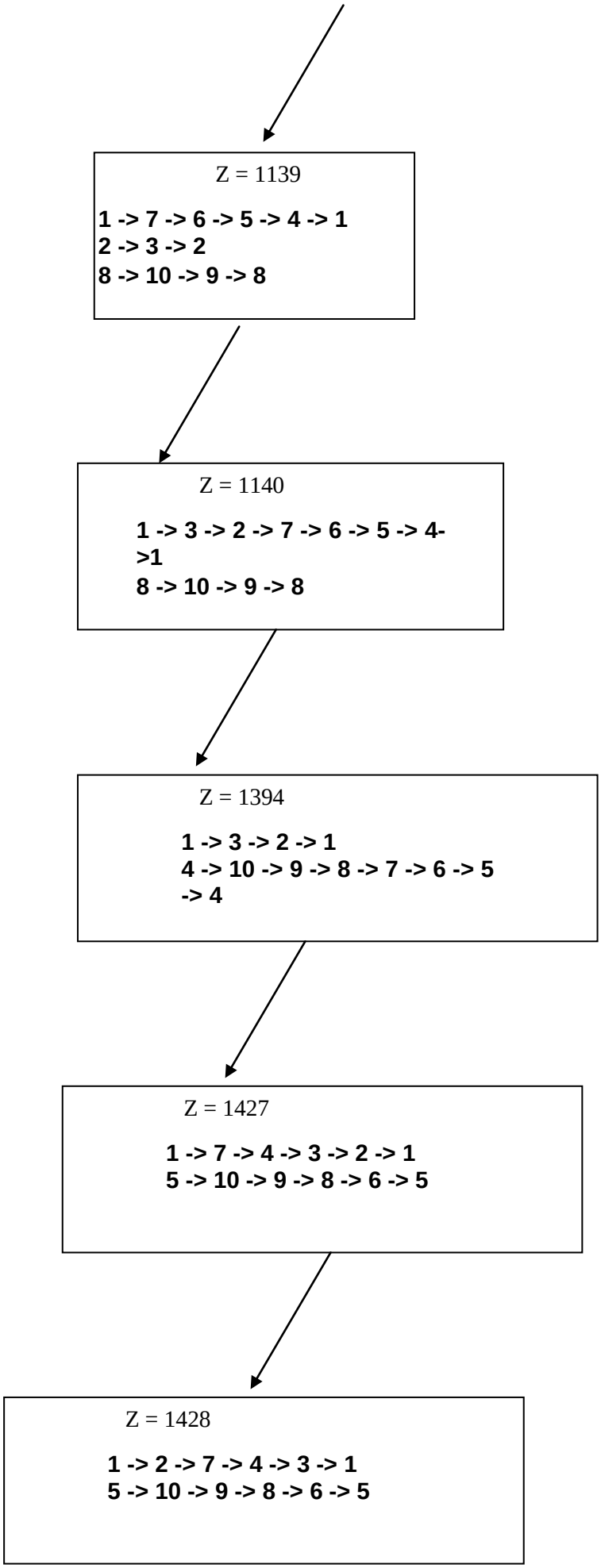
[5] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265.

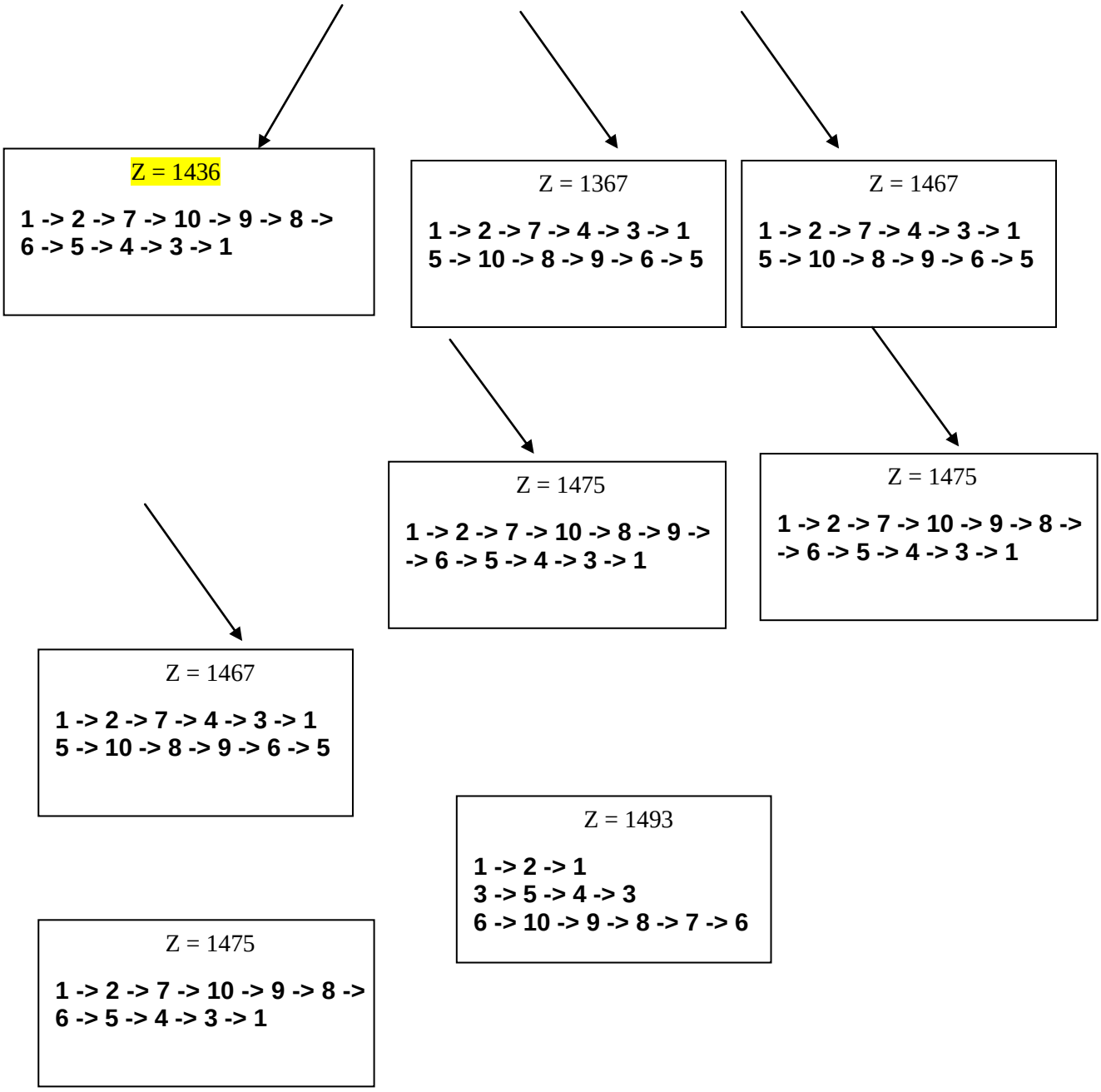
[6]Wagner, H. Principles of Operations Research, 2d ed. Englewood Cliffs, N.J.: Prentice Hall, 1975.

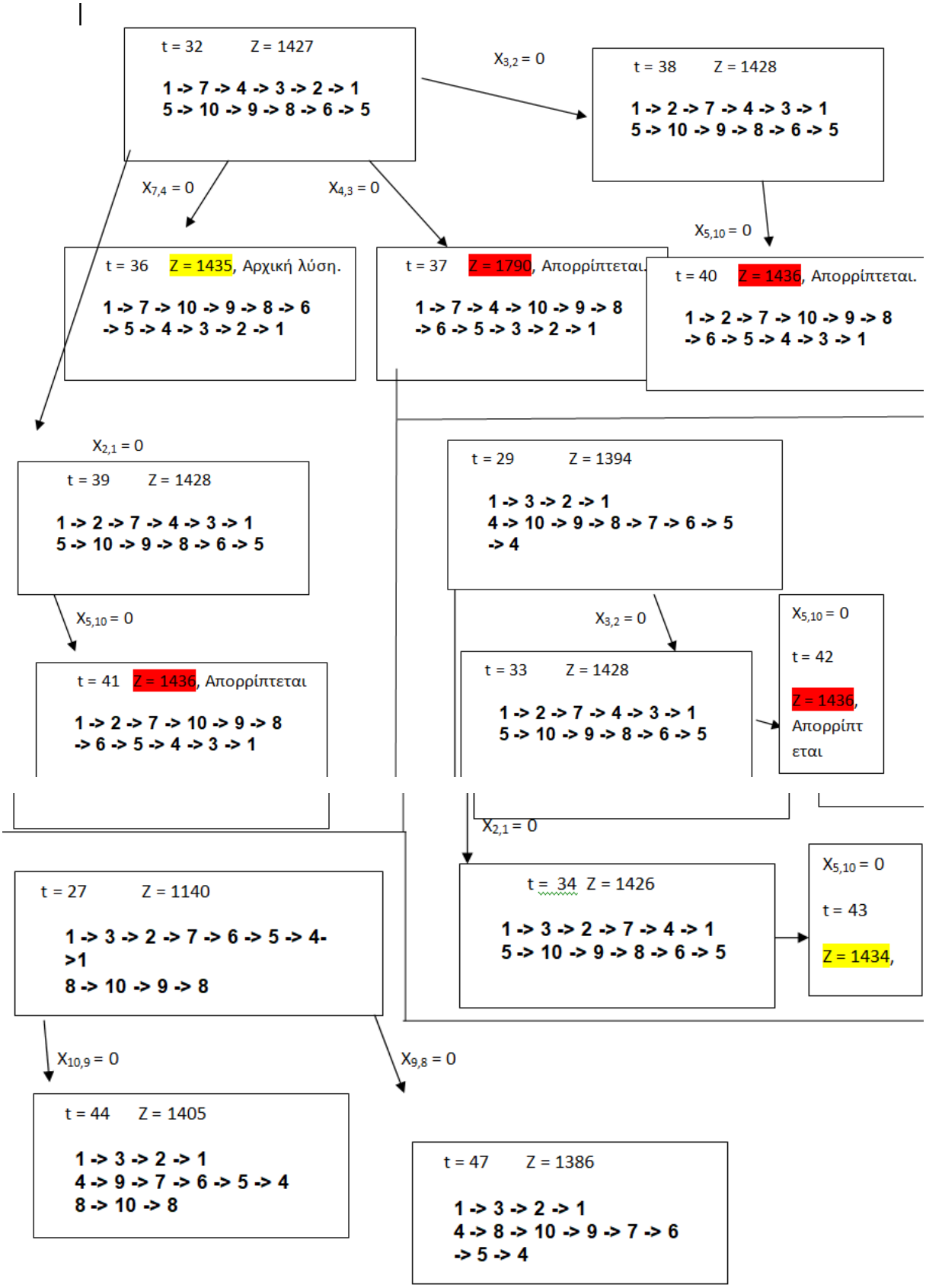
ΠΑΡΑΡΤΗΜΑ

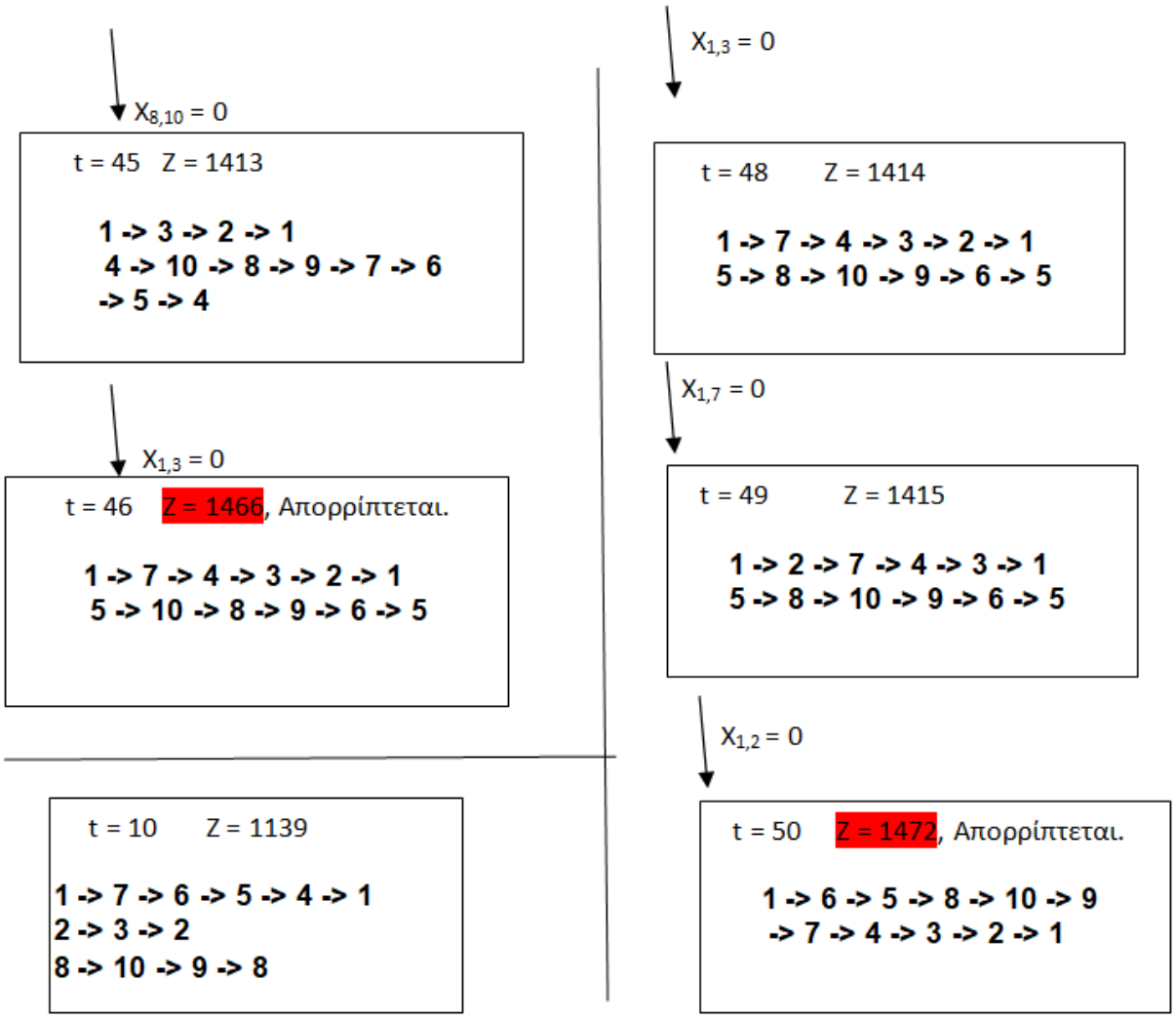

Z = 689
1 -> 4 -> 1
2 -> 3 -> 2
5 -> 6 -> 7 -> 5
8 -> 9 -> 10 -> 8
X14 = 1 X23 = 1 X32=1 …. X10,8 = 1
Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 7 -> 5
8 -> 9 -> 10 -> 8
Z = 691
1 -> 4 -> 3 -> 2 -> 1
5 -> 6 -> 7 -> 5
8 -> 9 -> 10 -> 8
Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
Z = 691
1 -> 4 -> 3 -> 2 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
Z = 954
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 6 -> 7 -> 8
9 -> 10 -> 9
Z = 824
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 5
7 -> 8 -> 7
9 -> 10 -> 9
Z = 954
1 -> 4 -> 3 -> 2 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9

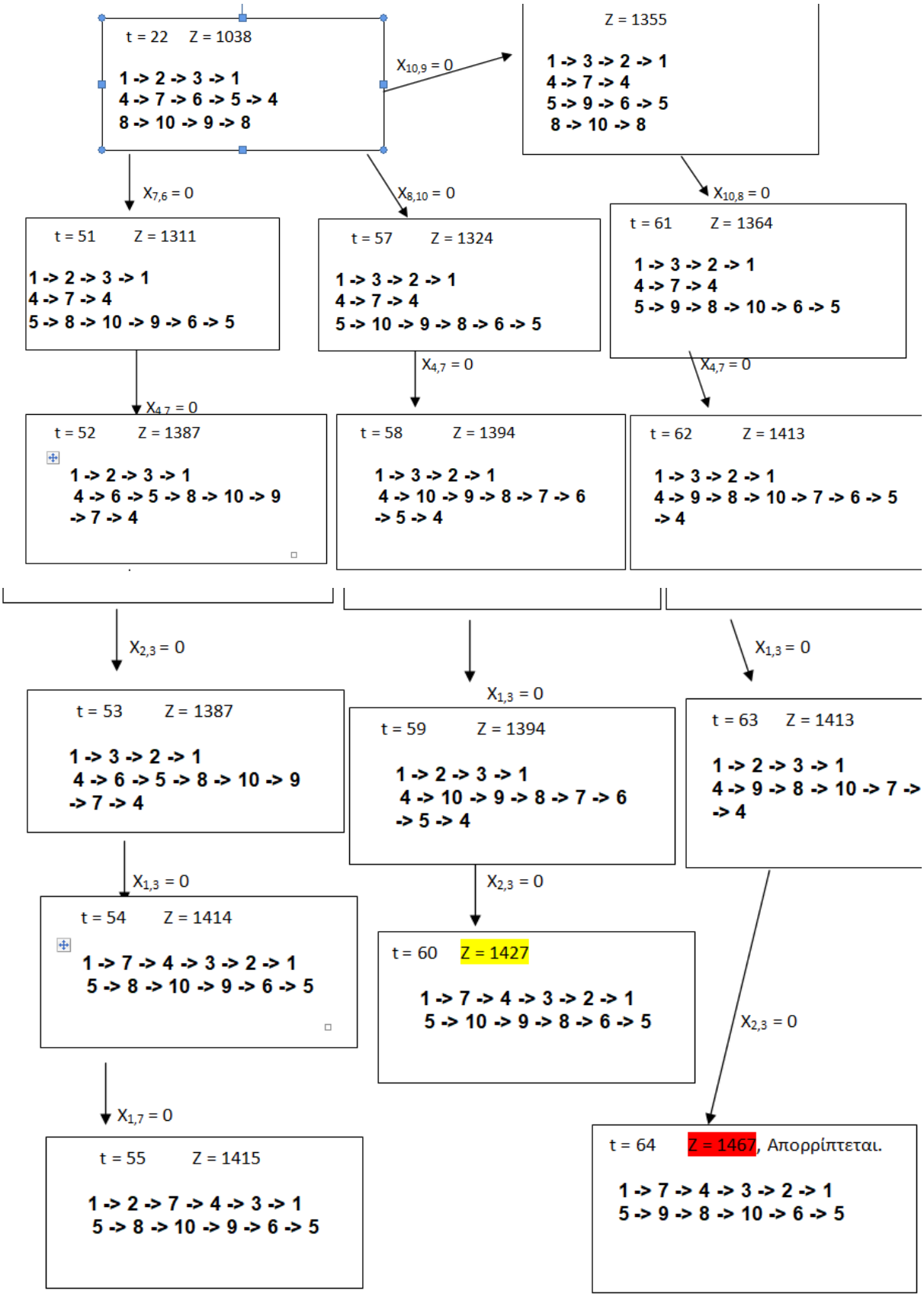

Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 ->5
Z = 843
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 5
7 -> 9 -> 10 -> 8 -> 7
Z = 973
1 -> 4 -> 3 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 ->5
Z = 954
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
Z = 1038
1 -> 2 -> 3 -> 1
4 -> 7 -> 6 -> 5 -> 4
8 -> 10 -> 9 -> 8

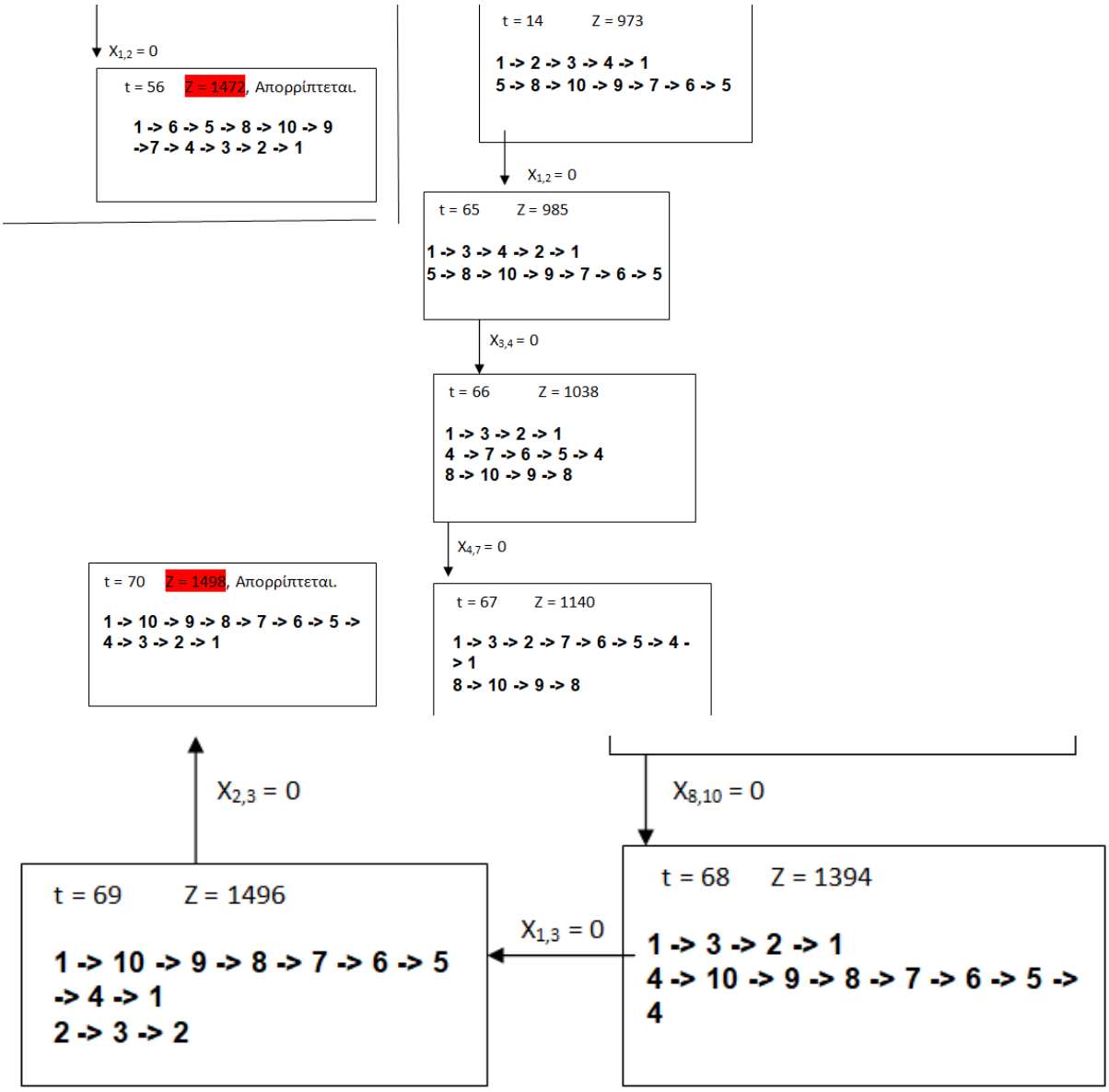

Z = 1139
1 -> 7 -> 6 -> 5 -> 4 -> 1
2 -> 3 -> 2
8 -> 10 -> 9 -> 8
Z = 1140
1 -> 3 -> 2 -> 7 -> 6 -> 5 -> 4-
>1
8 -> 10 -> 9 -> 8
Z = 1394
1 -> 3 -> 2 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5
-> 4
Z = 1427
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
Z = 1428
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5

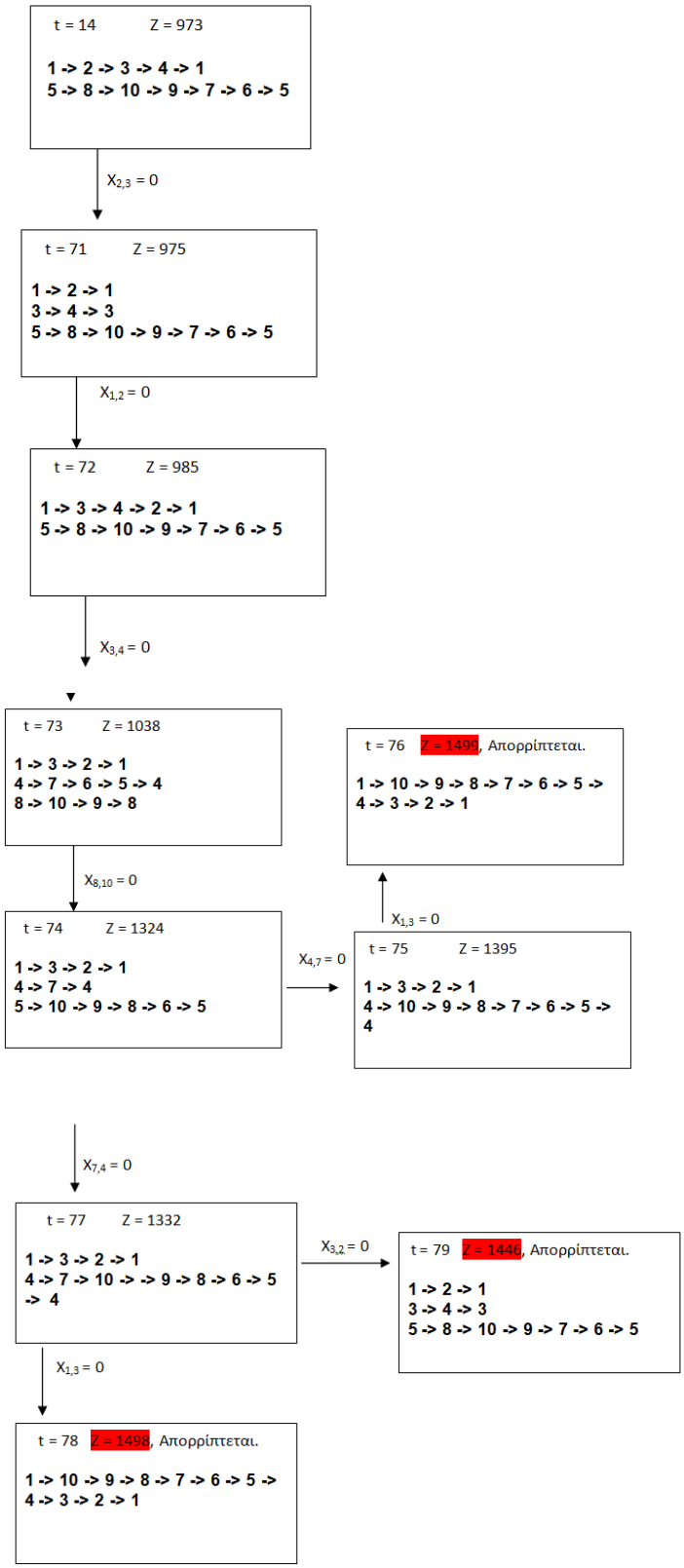

Z = 1436
1 -> 2 -> 7 -> 10 -> 9 -> 8 ->
6 -> 5 -> 4 -> 3 -> 1
Z = 1367
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 8 -> 9 -> 6 -> 5
Z = 1467
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 8 -> 9 -> 6 -> 5
Z = 1475
1 -> 2 -> 7 -> 10 -> 8 -> 9 ->
-> 6 -> 5 -> 4 -> 3 -> 1
Z = 1475
1 -> 2 -> 7 -> 10 -> 9 -> 8 ->
-> 6 -> 5 -> 4 -> 3 -> 1
Z = 1467
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 8 -> 9 -> 6 -> 5
Z = 1493
1 -> 2 -> 1
3 -> 5 -> 4 -> 3
6 -> 10 -> 9 -> 8 -> 7 -> 6
Z = 1475
1 -> 2 -> 7 -> 10 -> 9 -> 8 ->
6 -> 5 -> 4 -> 3 -> 1

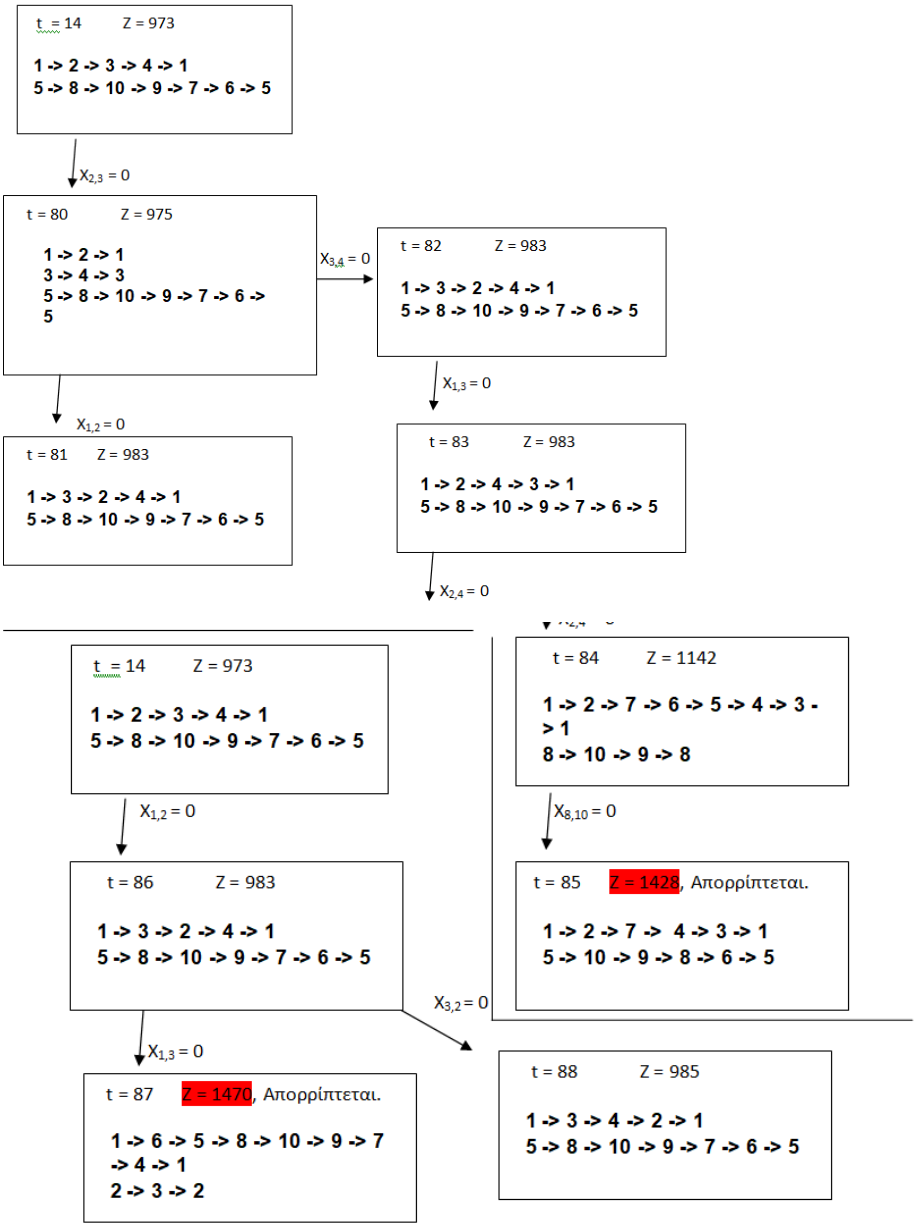

t = 32 Z = 1427
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X3,2 = 0
t = 38 Z = 1428
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X7,4 = 0
X4,3 = 0
X5,10 = 0
t = 36 Z = 1435, Αρχική λύση.
1 -> 7 -> 10 -> 9 -> 8 -> 6
-> 5 -> 4 -> 3 -> 2 -> 1
t = 37 Z = 1790, Απορρίπτεται.
1 -> 7 -> 4 -> 10 -> 9 -> 8
-> 6 -> 5 -> 3 -> 2 -> 1
t = 40 Z = 1436, Απορρίπτεται.
1 -> 2 -> 7 -> 10 -> 9 -> 8
-> 6 -> 5 -> 4 -> 3 -> 1
X2,1 = 0
t = 39 Z = 1428
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
t = 29 Z = 1394
1 -> 3 -> 2 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5
-> 4
X5,10 = 0
t = 41 Z = 1436, Απορρίπτεται
1 -> 2 -> 7 -> 10 -> 9 -> 8
-> 6 -> 5 -> 4 -> 3 -> 1
X3,2 = 0
t = 33 Z = 1428
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X5,10 = 0
t = 42
Z = 1436,
Απορρίπτ
εται
X2,1 = 0
t = 27 Z = 1140
1 -> 3 -> 2 -> 7 -> 6 -> 5 -> 4-
>1
8 -> 10 -> 9 -> 8
t = 34 Z = 1426
1 -> 3 -> 2 -> 7 -> 4 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X5,10 = 0
t = 43
Z = 1434,
X10,9 = 0
X9,8 = 0
t = 44 Z = 1405
1 -> 3 -> 2 -> 1
4 -> 9 -> 7 -> 6 -> 5 -> 4
8 -> 10 -> 8
t = 47 Z = 1386
1 -> 3 -> 2 -> 1
4 -> 8 -> 10 -> 9 -> 7 -> 6
-> 5 -> 4

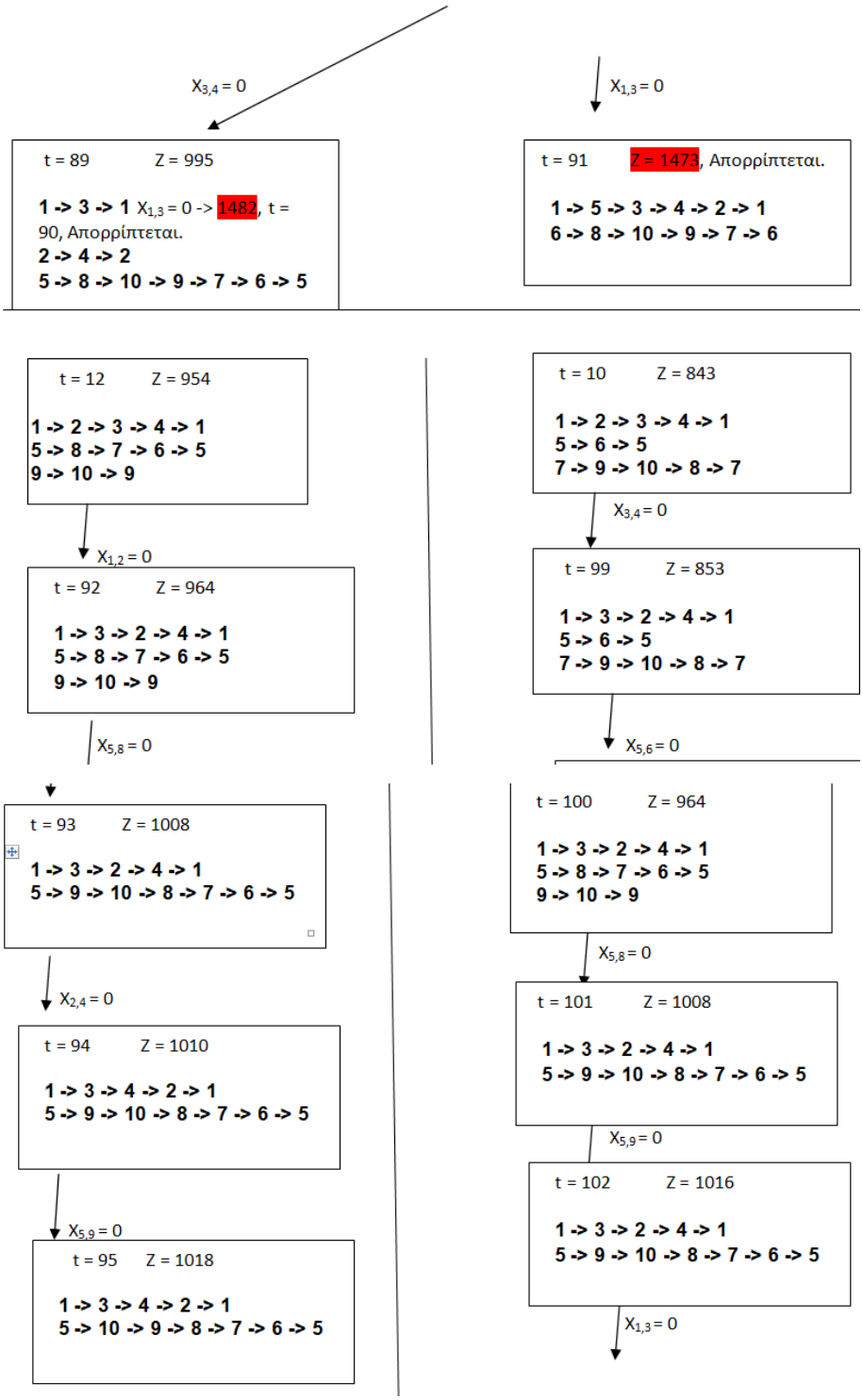

$X_{8,10} = 0$
t = 45 Z = 1413
1 -> 3 -> 2 -> 1
4 -> 10 -> 8 -> 9 -> 7 -> 6
-> 5 -> 4
$X_{1,3} = 0$
t = 46 Z = 1466, Απορρίπτεται.
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 10 -> 8 -> 9 -> 6 -> 5
t = 10 Z = 1139
1 -> 7 -> 6 -> 5 -> 4 -> 1
2 -> 3 -> 2
8 -> 10 -> 9 -> 8
$X_{1,3} = 0$
t = 48 Z = 1414
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 6 -> 5
$X_{1,7} = 0$
t = 49 Z = 1415
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 8 -> 10 -> 9 -> 6 -> 5
$X_{1,2} = 0$
t = 50 Z = 1472, Απορρίπτεται.
1 -> 6 -> 5 -> 8 -> 10 -> 9
-> 7 -> 4 -> 3 -> 2 -> 1

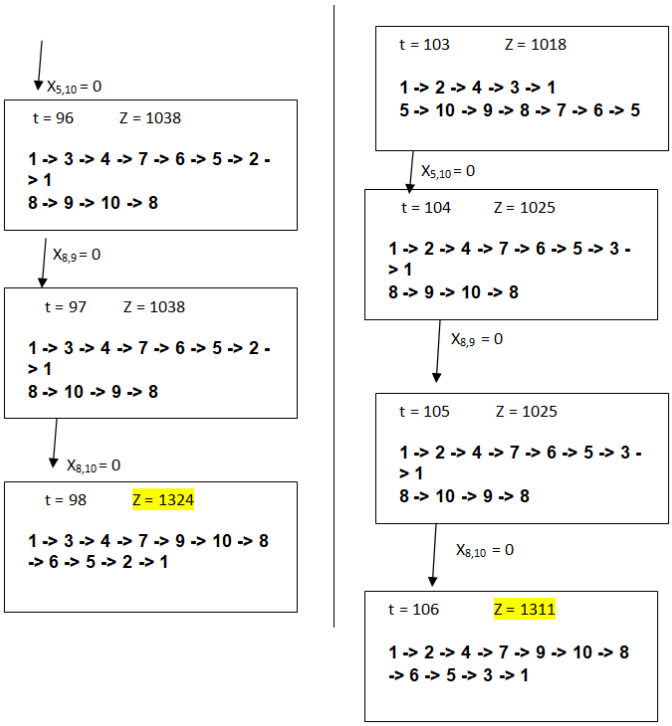

t = 22 Z = 1038
1 -> 2 -> 3 -> 1
4 -> 7 -> 6 -> 5 -> 4
8 -> 10 -> 9 -> 8
X10,9 = 0
Z = 1355
1 -> 3 -> 2 -> 1
4 -> 7 -> 4
5 -> 9 -> 6 -> 5
8 -> 10 -> 8
X7,6 = 0
X8,10 = 0
X10,8 = 0
t = 51 Z = 1311
1 -> 2 -> 3 -> 1
4 -> 7 -> 4
5 -> 8 -> 10 -> 9 -> 6 -> 5
t = 57 Z = 1324
1 -> 3 -> 2 -> 1
4 -> 7 -> 4
5 -> 10 -> 9 -> 8 -> 6 -> 5
t = 61 Z = 1364
1 -> 3 -> 2 -> 1
4 -> 7 -> 4
5 -> 9 -> 8 -> 10 -> 6 -> 5
X4,7 = 0
X4,7 = 0
X4,7 = 0
t = 52 Z = 1387
1 -> 2 -> 3 -> 1
4 -> 6 -> 5 -> 8 -> 10 -> 9
-> 7 -> 4
t = 58 Z = 1394
1 -> 3 -> 2 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6
-> 5 -> 4
t = 62 Z = 1413
1 -> 3 -> 2 -> 1
4 -> 9 -> 8 -> 10 -> 7 -> 6 -> 5
-> 4
X2,3 = 0
X1,3 = 0
X1,3 = 0
t = 53 Z = 1387
1 -> 3 -> 2 -> 1
4 -> 6 -> 5 -> 8 -> 10 -> 9
-> 7 -> 4
t = 59 Z = 1394
1 -> 2 -> 3 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6
-> 5 -> 4
t = 63 Z = 1413
1 -> 2 -> 3 -> 1
4 -> 9 -> 8 -> 10 -> 7 ->
-> 4
X1,3 = 0
X2,3 = 0
X2,3 = 0
t = 54 Z = 1414
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 6 -> 5
t = 60 Z = 1427
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X1,7 = 0
t = 55 Z = 1415
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 8 -> 10 -> 9 -> 6 -> 5
t = 64 Z = 1467, Απορρίπτεται.
1 -> 7 -> 4 -> 3 -> 2 -> 1
5 -> 9 -> 8 -> 10 -> 6 -> 5

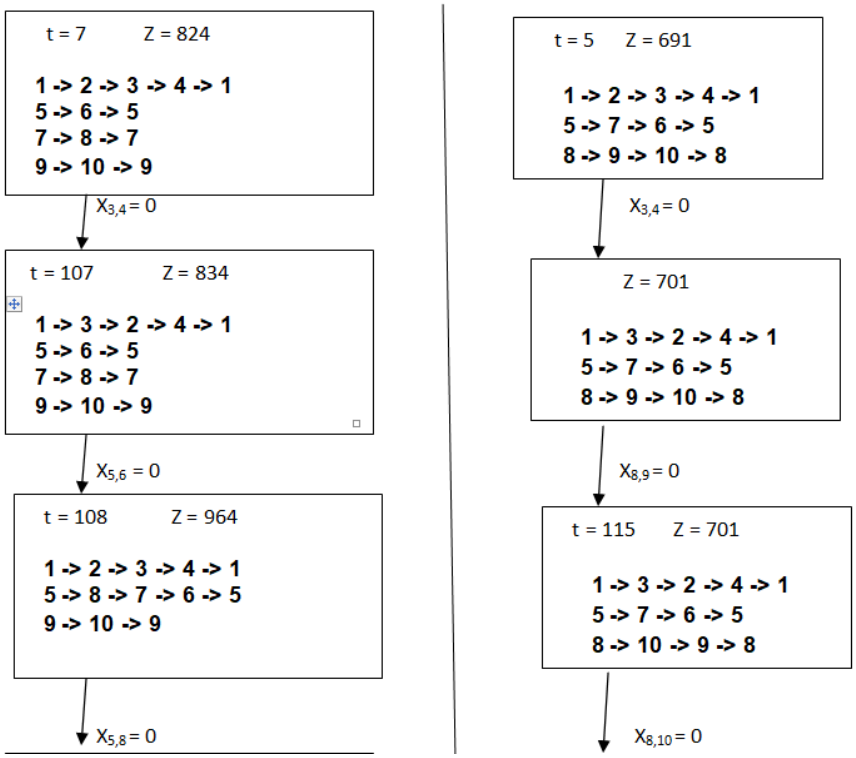

$X_{1,2} = 0$
t = 56 Z = 1472, Απορρίπτεται.
1 -> 6 -> 5 -> 8 -> 10 -> 9 ->7 -> 4 -> 3 -> 2 -> 1
t = 14 Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
$X_{1,2} = 0$
t = 65 Z = 985
1 -> 3 -> 4 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
$X_{3,4} = 0$
t = 66 Z = 1038
1 -> 3 -> 2 -> 1
4 -> 7 -> 6 -> 5 -> 4
8 -> 10 -> 9 -> 8
$X_{4,7} = 0$
t = 70 Z = 1498, Απορρίπτεται.
1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
t = 67 Z = 1140
1 -> 3 -> 2 -> 7 -> 6 -> 5 -> 4 -> 1
8 -> 10 -> 9 -> 8
$X_{2,3} = 0$
$X_{8,10} = 0$
t = 69 Z = 1496
1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 1
2 -> 3 -> 2
$X_{1,3} = 0$
t = 68 Z = 1394
1 -> 3 -> 2 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4

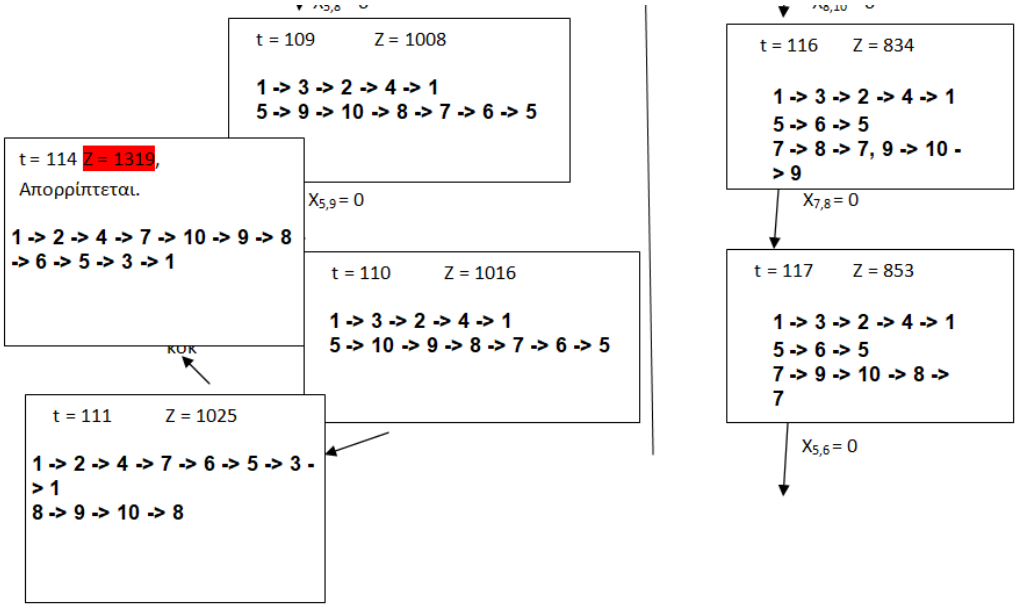

t = 14 Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X2,3 = 0
t = 71 Z = 975
1 -> 2 -> 1
3 -> 4 -> 3
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X1,2 = 0
t = 72 Z = 985
1 -> 3 -> 4 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X3,4 = 0
t = 73 Z = 1038
1 -> 3 -> 2 -> 1
4 -> 7 -> 6 -> 5 -> 4
8 -> 10 -> 9 -> 8
t = 76 Z = 1499, Απορρίπτεται.
1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 ->
4 -> 3 -> 2 -> 1
X8,10 = 0
X1,3 = 0
t = 74 Z = 1324
1 -> 3 -> 2 -> 1
4 -> 7 -> 4
5 -> 10 -> 9 -> 8 -> 6 -> 5
X4,7 = 0
t = 75 Z = 1395
1 -> 3 -> 2 -> 1
4 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 ->
4
X7,4 = 0
t = 77 Z = 1332
1 -> 3 -> 2 -> 1
4 -> 7 -> 10 -> -> 9 -> 8 -> 6 -> 5
-> 4
X3,2 = 0
t = 79 Z = 1446, Απορρίπτεται.
1 -> 2 -> 1
3 -> 4 -> 3
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X1,3 = 0
t = 78 Z = 1498, Απορρίπτεται.
1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 ->
4 -> 3 -> 2 -> 1

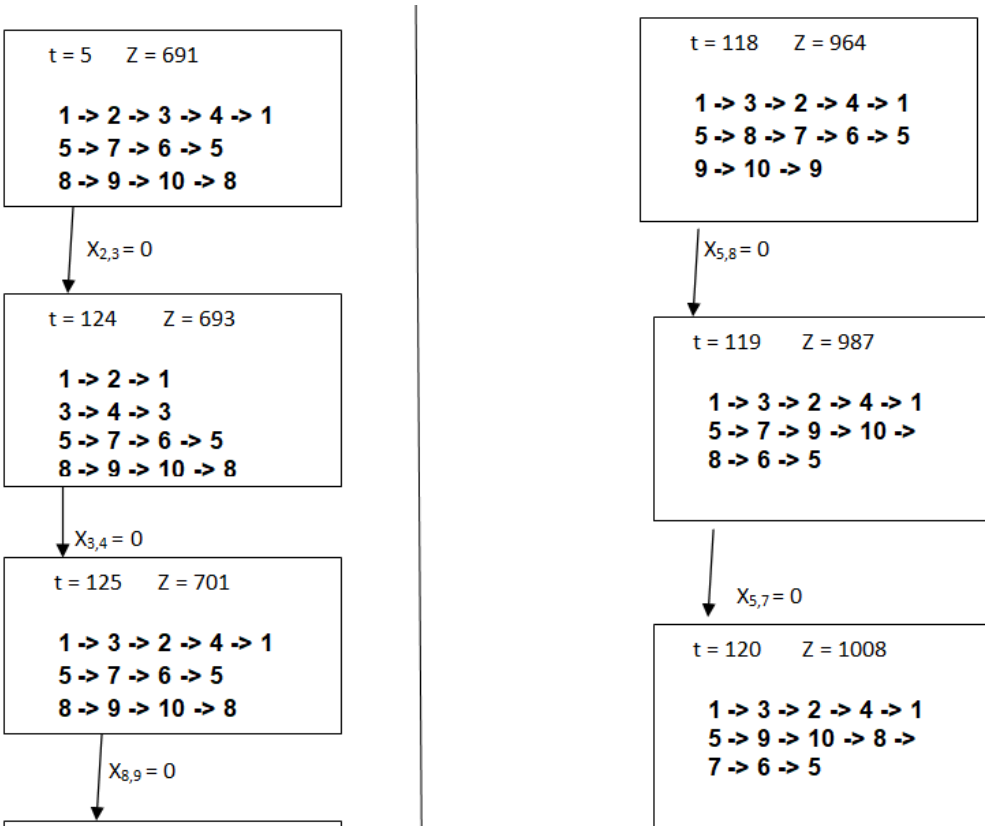

t = 14 Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X2,3 = 0
t = 80 Z = 975
1 -> 2 -> 1
3 -> 4 -> 3
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X3,4 = 0
t = 82 Z = 983
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X1,2 = 0
t = 81 Z = 983
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X1,3 = 0
t = 83 Z = 983
1 -> 2 -> 4 -> 3 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X2,4 = 0
t = 14 Z = 973
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
X1,2 = 0
t = 84 Z = 1142
1 -> 2 -> 7 -> 6 -> 5 -> 4 -> 3 -> 1
8 -> 10 -> 9 -> 8
X8,10 = 0
t = 86 Z = 983
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
t = 85 Z = 1428, Απορρίπτεται.
1 -> 2 -> 7 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 6 -> 5
X3,2 = 0
X1,3 = 0
t = 87 Z = 1470, Απορρίπτεται.
1 -> 6 -> 5 -> 8 -> 10 -> 9 -> 7 -> 4 -> 1
2 -> 3 -> 2
t = 88 Z = 985
1 -> 3 -> 4 -> 2 -> 1
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5

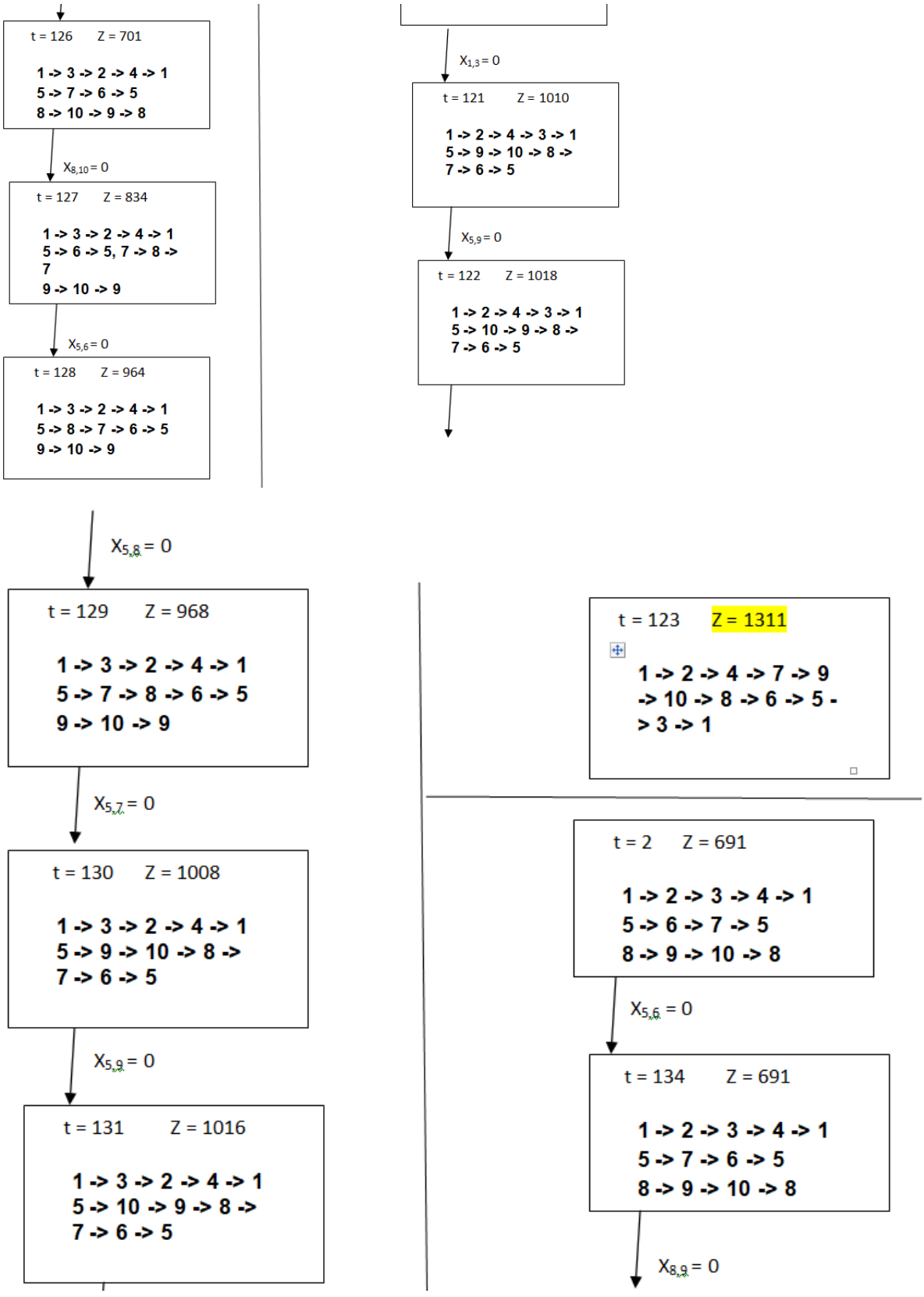

$X_{3,4} = 0$
$X_{1,3} = 0$
t = 89 Z = 995
1 -> 3 -> 1 $X_{1,3} = 0$ -> 1482, t = 90, Απορρίπτεται.
2 -> 4 -> 2
5 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5
t = 91 Z = 1473, Απορρίπτεται.
1 -> 5 -> 3 -> 4 -> 2 -> 1
6 -> 8 -> 10 -> 9 -> 7 -> 6
t = 12 Z = 954
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
$X_{1,2} = 0$
t = 92 Z = 964
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
$X_{5,8} = 0$
t = 93 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5
$X_{2,4} = 0$
t = 94 Z = 1010
1 -> 3 -> 4 -> 2 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5
$X_{5,9} = 0$
t = 95 Z = 1018
1 -> 3 -> 4 -> 2 -> 1
5 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5
t = 10 Z = 843
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 5
7 -> 9 -> 10 -> 8 -> 7
$X_{3,4} = 0$
t = 99 Z = 853
1 -> 3 -> 2 -> 4 -> 1
5 -> 6 -> 5
7 -> 9 -> 10 -> 8 -> 7
$X_{5,6} = 0$
t = 100 Z = 964
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
$X_{5,8} = 0$
t = 101 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5
$X_{5,9} = 0$
t = 102 Z = 1016
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5
$X_{1,3} = 0$

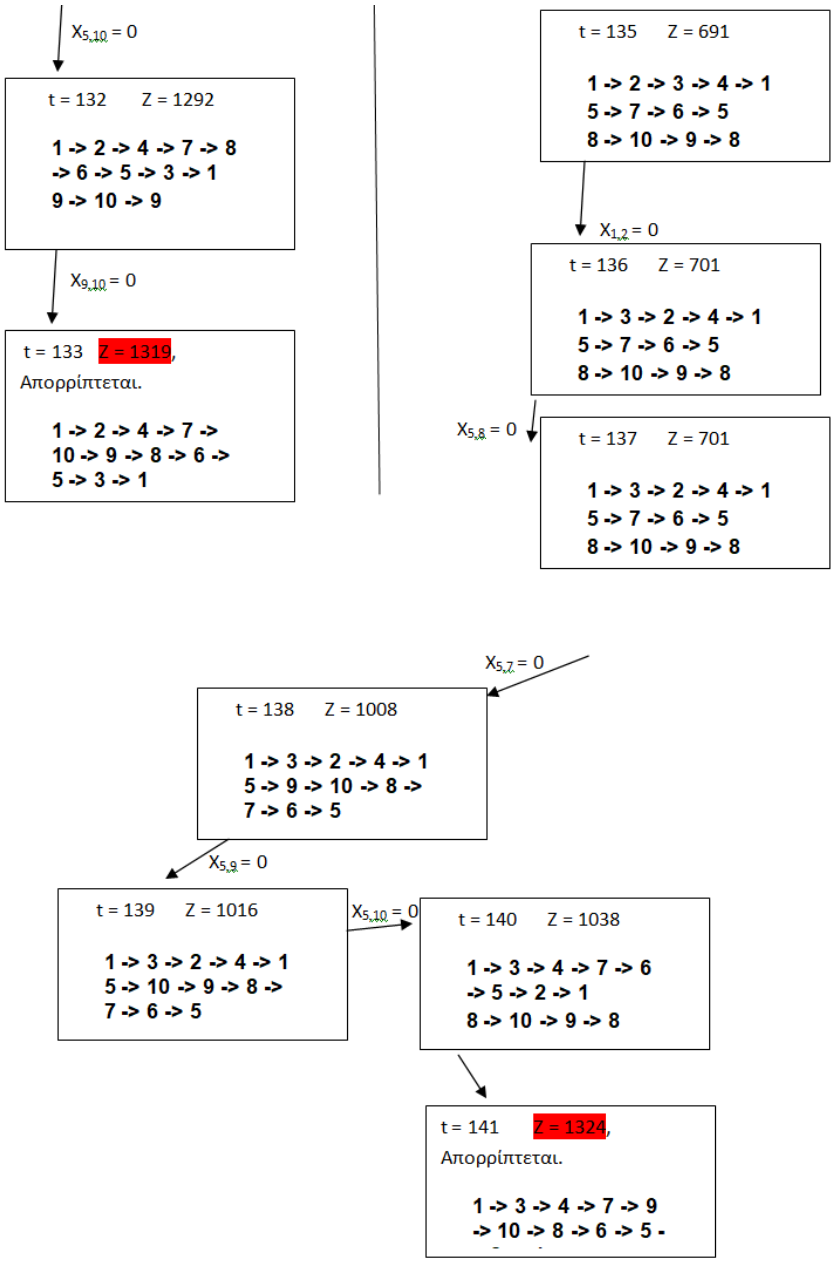

$X_{5,10} = 0$
t = 96 Z = 1038
1 -> 3 -> 4 -> 7 -> 6 -> 5 -> 2 -> 1
8 -> 9 -> 10 -> 8
$X_{8,9} = 0$
t = 97 Z = 1038
1 -> 3 -> 4 -> 7 -> 6 -> 5 -> 2 -> 1
8 -> 10 -> 9 -> 8
$X_{8,10} = 0$
t = 98 Z = 1324
1 -> 3 -> 4 -> 7 -> 9 -> 10 -> 8 -> 6 -> 5 -> 2 -> 1
t = 103 Z = 1018
1 -> 2 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5
$X_{5,10} = 0$
t = 104 Z = 1025
1 -> 2 -> 4 -> 7 -> 6 -> 5 -> 3 -> 1
8 -> 9 -> 10 -> 8
$X_{8,9} = 0$
t = 105 Z = 1025
1 -> 2 -> 4 -> 7 -> 6 -> 5 -> 3 -> 1
8 -> 10 -> 9 -> 8
$X_{8,10} = 0$
t = 106 Z = 1311
1 -> 2 -> 4 -> 7 -> 9 -> 10 -> 8 -> 6 -> 5 -> 3 -> 1


t = 7 Z = 824
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 5
7 -> 8 -> 7
9 -> 10 -> 9
$X_{3,4} = 0$
t = 107 Z = 834
1 -> 3 -> 2 -> 4 -> 1
5 -> 6 -> 5
7 -> 8 -> 7
9 -> 10 -> 9
$X_{5,6} = 0$
t = 108 Z = 964
1 -> 2 -> 3 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
$X_{5,8} = 0$
t = 5 Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
$X_{3,4} = 0$
Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
$X_{8,9} = 0$
t = 115 Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 10 -> 9 -> 8
$X_{8,10} = 0$

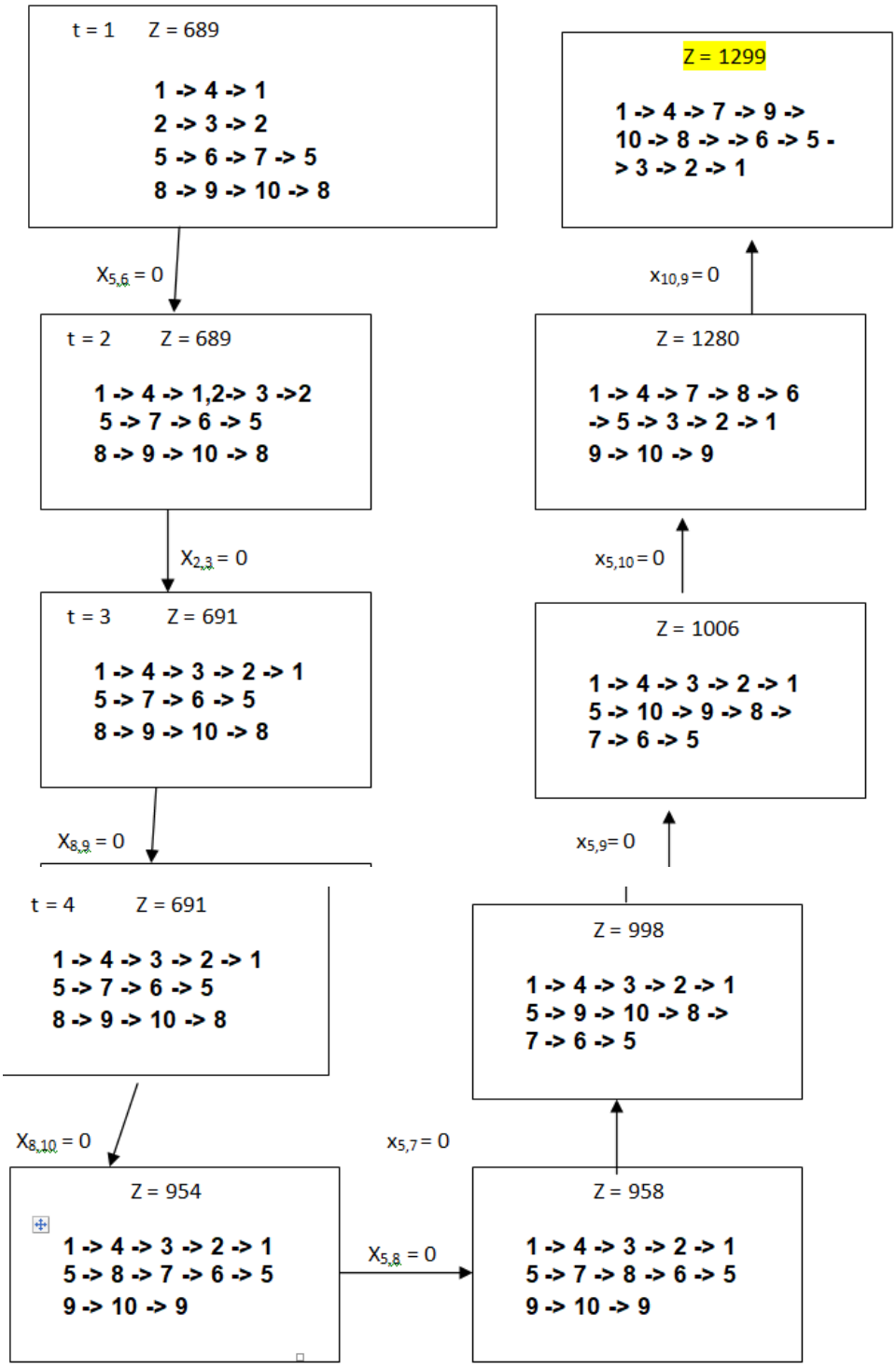

t = 109 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5
t = 114 Z = 1319,
Απορρίπτεται.
1 -> 2 -> 4 -> 7 -> 10 -> 9 -> 8 -> 6 -> 5 -> 3 -> 1
X5,9 = 0
t = 110 Z = 1016
1 -> 3 -> 2 -> 4 -> 1
5 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5
t = 111 Z = 1025
1 -> 2 -> 4 -> 7 -> 6 -> 5 -> 3 -> 1
8 -> 9 -> 10 -> 8
t = 116 Z = 834
1 -> 3 -> 2 -> 4 -> 1
5 -> 6 -> 5
7 -> 8 -> 7, 9 -> 10 -> 9
X7,8 = 0
t = 117 Z = 853
1 -> 3 -> 2 -> 4 -> 1
5 -> 6 -> 5
7 -> 9 -> 10 -> 8 -> 7
X5,6 = 0


t = 5 Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X2,3 = 0
t = 124 Z = 693
1 -> 2 -> 1
3 -> 4 -> 3
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X3,4 = 0
t = 125 Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X8,9 = 0
t = 118 Z = 964
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
X5,8 = 0
t = 119 Z = 987
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 9 -> 10 -> 8 -> 6 -> 5
X5,7 = 0
t = 120 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 -> 7 -> 6 -> 5


t = 126 Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 10 -> 9 -> 8
X8,10 = 0
t = 127 Z = 834
1 -> 3 -> 2 -> 4 -> 1
5 -> 6 -> 5, 7 -> 8 ->
7
9 -> 10 -> 9
X5,6 = 0
t = 128 Z = 964
1 -> 3 -> 2 -> 4 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
X1,3 = 0
t = 121 Z = 1010
1 -> 2 -> 4 -> 3 -> 1
5 -> 9 -> 10 -> 8 ->
7 -> 6 -> 5
X5,9 = 0
t = 122 Z = 1018
1 -> 2 -> 4 -> 3 -> 1
5 -> 10 -> 9 -> 8 ->
7 -> 6 -> 5
X5,8 = 0
t = 129 Z = 968
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 8 -> 6 -> 5
9 -> 10 -> 9
X5,7 = 0
t = 130 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 ->
7 -> 6 -> 5
X5,9 = 0
t = 131 Z = 1016
1 -> 3 -> 2 -> 4 -> 1
5 -> 10 -> 9 -> 8 ->
7 -> 6 -> 5
t = 123 Z = 1311
1 -> 2 -> 4 -> 7 -> 9
-> 10 -> 8 -> 6 -> 5 -
> 3 -> 1
t = 2 Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 6 -> 7 -> 5
8 -> 9 -> 10 -> 8
X5,6 = 0
t = 134 Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X8,9 = 0


X5,10 = 0
t = 132 Z = 1292
1 -> 2 -> 4 -> 7 -> 8
-> 6 -> 5 -> 3 -> 1
9 -> 10 -> 9
X9,10 = 0
t = 133 Z = 1319,
Απορρίπτεται.
1 -> 2 -> 4 -> 7 ->
10 -> 9 -> 8 -> 6 ->
5 -> 3 -> 1
t = 135 Z = 691
1 -> 2 -> 3 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 10 -> 9 -> 8
X1,2 = 0
t = 136 Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 10 -> 9 -> 8
X5,8 = 0
t = 137 Z = 701
1 -> 3 -> 2 -> 4 -> 1
5 -> 7 -> 6 -> 5
8 -> 10 -> 9 -> 8

X5,7 = 0
t = 138 Z = 1008
1 -> 3 -> 2 -> 4 -> 1
5 -> 9 -> 10 -> 8 ->
7 -> 6 -> 5
X5,9 = 0
t = 139 Z = 1016
1 -> 3 -> 2 -> 4 -> 1
5 -> 10 -> 9 -> 8 ->
7 -> 6 -> 5
X5,10 = 0
t = 140 Z = 1038
1 -> 3 -> 4 -> 7 -> 6
-> 5 -> 2 -> 1
8 -> 10 -> 9 -> 8
t = 141 Z = 1324,
Απορρίπτεται.
1 -> 3 -> 4 -> 7 -> 9
-> 10 -> 8 -> 6 -> 5 -


t = 1 Z = 689
1 -> 4 -> 1
2 -> 3 -> 2
5 -> 6 -> 7 -> 5
8 -> 9 -> 10 -> 8
X5,6 = 0
t = 2 Z = 689
1 -> 4 -> 1,2-> 3 ->2
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X2,3 = 0
t = 3 Z = 691
1 -> 4 -> 3 -> 2 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X8,9 = 0
t = 4 Z = 691
1 -> 4 -> 3 -> 2 -> 1
5 -> 7 -> 6 -> 5
8 -> 9 -> 10 -> 8
X8,10 = 0
Z = 954
1 -> 4 -> 3 -> 2 -> 1
5 -> 8 -> 7 -> 6 -> 5
9 -> 10 -> 9
X5,8 = 0
Z = 958
1 -> 4 -> 3 -> 2 -> 1
5 -> 7 -> 8 -> 6 -> 5
9 -> 10 -> 9
x5,7 = 0
Z = 998
1 -> 4 -> 3 -> 2 -> 1
5 -> 9 -> 10 -> 8 ->
7 -> 6 -> 5
x5,9= 0
Z = 1006
1 -> 4 -> 3 -> 2 -> 1
5 -> 10 -> 9 -> 8 ->
7 -> 6 -> 5
x5,10 = 0
Z = 1280
1 -> 4 -> 7 -> 8 -> 6
-> 5 -> 3 -> 2 -> 1
9 -> 10 -> 9
x10,9 = 0
Z = 1299
1 -> 4 -> 7 -> 9 ->
10 -> 8 -> -> 6 -> 5 -
> 3 -> 2 -> 1