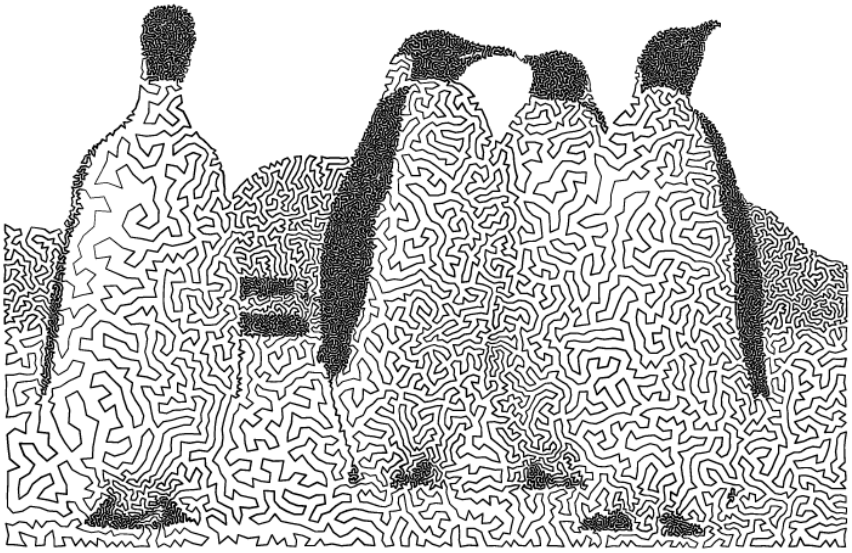
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΥΡΕΣΗ ΒΕΛΤΙΣΤΗΣ ΛΥΣΗΣ ΔΥΝΑΜΙΚΟΥ ΑΣΣΥΜΕΤΡΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΠΛΑΝΟΔΙΟΥ ΠΩΛΗΤΗ / OPTIMIZATION OF DYNAMIC ATSP



ΒΑΣΙΛΑΚΑΚΟΣ ΒΑΣΙΛΕΙΟΣ

Επιβλέπων:

Μαρινάκης Ιωάννης
Λέκτορας Πολυτεχνείου Κρήτης

Χανιά 2013

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία αναλύουμε το δυναμικό πρόγραμμα εύρεσης βέλτιστης διαδρομής F.O.R (Finding Optimal Route).

Θα ασχοληθούμε με ένα από τα πιο στοιχειώδη προβλήματα βελτιστοποίησης, το πρόβλημα του πλανόδιου πωλητή TSP(Travelling Salesman Problem), εισάγωντας το σε ένα δυναμικό περιβάλλον.

Σκοπός αυτής της διπλωματικής εργασίας είναι η εύρεση βέλτιστης διαδρομής σε δίκτυα παράδοσης, δηλαδή σε προκαθορισμένες διαδρομές, βρισκόμενες σε δυναμικό περιβάλλον.

Ο αλγόριθμος που χρησιμοποιούμε για την επίλυση του προβλήματος του πλανόδιου πωλητή ονομάζεται Concorde, (των David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook) και θεωρείται παγκοσμίως, ο πιο γρήγορος και αξιόπιστος λύτης προβλημάτων που ανήκουν στην κατηγορία TSP.

Θα δωθεί βαρύτητα στα αποτελέσματα που θα προκύψουν, συγκρίνοντας λύσεις του ίδιου προβλήματος δίκτυου παράδοσης, αρχικά προσεγγίζοντας το σε στατικό και έπειτα σε δυναμικό περιβάλλον.

Ο τρόπος με τον οποίο δημιουργούμε το δυναμικό περιβάλλον είναι η εισαγωγή δεδομένων κυκλοφοριακής συμφόρησης, στο αντίστοιχο πρόβλημα εύρεσης βέλτιστης διαδρομής. Τα δεδομένα κυκλοφοριακής συμφόρησης πραγματικού χρόνου είναι πειραματικά-ψευδή, δηλαδή παράγονται από μια τυχαία συνάρτηση που έχουμε ορίσει, ανανεώνοντας τον πίνακα κόστους των διαδρομών από το ένα σημείο παράδοσης στο άλλο. Αυτό έχει ως αποτέλεσμα κάθε φορά που επιλύουμε το αντίστοιχο πρόβλημα δίκτυου παράδοσης, να δίνεται η νέα βέλτιστη διαδρομή που ανταποκρίνεται στα δεδομένα του νέου (τυχαία παραγόμενου) δυναμικού περιβάλλοντος.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στον Τομέα Στοχαστική Βελτιστοποίηση και Εφαρμογές του Τμήματος Μηχανικών Παραγωγής και Διοίκησης του Πολυτεχνείου Κρήτης υπό την επίβλεψη του Dr. Ιωάννη Μαρινάκη. Οφείλω να ευχαριστήσω θερμά όλους όσους συνέβαλλαν στην ολοκλήρωση της διπλωματικής μου εργασίας.

Ιδιαίτερα ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Ιωάννη Μαρινάκη για την εμπιστοσύνη του και το ενδιαφέρον που έδειξε κατά την ανάθεση της εργασίας καθώς επίσης και για την συμπαράστασή του καθ'όλη την διάρκεια της.

Θα ήθελα να ευχαριστήσω τον φίλο μου Γεώργιο Πιερρή, υποψήφιο διδάκτορα του University of Wales, Newport στον τομέα Machine Learning and Robotics, για τις συμβουλές του και την αμέριστη συμπαράσταση του κατά την διαικπεραίωση της εργασίας μου, καθώς επίσης και την αγαπημένη μου Αιμιλία Θεοδωρακοπούλου για την πολύτιμη στήριξη της καθ'όλη την διάρκεια της προσπάθειας μου.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

1.1 ΕΙΣΑΓΩΓΗ

Στις μέρες μας οι αναπτυγμένες κοινωνίες έχουν διαμορφωθεί βασιζόμενες σε αλληλένδετες εφοδιαστικές αλυσίδες, οι οποίες μετασχηματίζουν ακατέργαστα υλικά και φυσικά αποθέματα σε τελικά προιόντα, όπου μέσω αυτών, παραδίδονται στους τελικούς καταναλωτές. Κάθε στοιχείο των εφοδιαστικών αλυσίδων είναι καθοριστικής σημασίας διότι αν δεν πληρούνται οι απαιτήσεις του σε σχέση με τα υπόλοιπα τότε το τελικό προιόν-υπηρεσία υποβαθμίζεται.

Ειδικότερα, ο χρόνος παράδοσης είναι ένα κρίσιμο στοιχείο. Η αποτελεσματική παράδοση προιόντων είναι υψίστης σημασίας για μία εταιρία αφού έτσι μπορεί να επιτευχθεί χαμηλό κόστος και οι πελάτες να είναι ικανοποιημένοι.

Τα σύγχρονα αστικά κέντρα σε ΗΠΑ, ΕΥΡΩΠΗ και ΑΣΙΑ είναι πυκνοκατοικημένα και σταθερά αυξάνεται ο πληθυσμός τους με γρήγορους ρυθμούς, με αποτέλεσμα οι κυβερνήσεις να δυσκολεύονται να ανταπεξέλθουν στις αυξανόμενες πληθυσμιακές ανάγκες. Η κυκλοφοριακή συμφόρηση σε κεντρικά οδικά δίκτυα είναι πλέον συχνή, λόγω φυσικών φαινομένων, απεργιών, ωρών αιχμής κ.τ.λ.

Ένα από τα στοιχειώδη προβλήματα που προσπαθεί να επιλύσει η επιστήμη της εφοδιαστικής αλυσίδας είναι το πρόβλημα του πλανώδιου πωλητη (TSP). Το 1800 ο Ιρλανδός μαθηματικός Sir William Rowan Hamilton και ο βρετανός μαθηματικός Thomas Kirkman ασχολήθηκαν με προβλήματα που σχετίζονται με το TSP. Το 1930 η γενική μορφή του TSP μελετήθηκε από μαθηματικούς στη Βιέννη και το Χάρβαρντ, και συγκεκριμένα από τον Karl Menger. Μέχρι και σήμερα το πρόβλημα του πλανώδιου πωλητή TSP παραμένει ένα από τα πιο δημοφιλή μαθηματικά προβλήματα και παρόλο που έχει προσελκύσει μεγάλο ποσοστό μελετητών δεν έχει βρεθεί ακόμα λύση για τη γενική περίπτωση.

Το περιβάλλον μας δεν είναι στατικό, κάνοντας ανεπαρκής τους στατικούς αλγόριθμους δρομολόγησης οχημάτων (Vehicle Routing Problem). Το ίδιο ισχύει και για την περίπτωση του προβλήματος του πλανώδιου πωλητή TSP. Η πρόκληση που αντιμετωπίζουμε πλέον στις μέρες μας, είναι να εισάγουμε τα προβλήματα της εφοδιαστικής αλυσίδας και πιο συγκεκριμένα το πρόβλημα του πλανώδιου πωλητή TSP σε ένα δυναμικό περιβάλλον, στο περιβάλλον όπου και ζούμε.

ΚΕΦΑΛΑΙΟ 2

2.1 Εφοδιαστική Αλυσίδα

Η εφοδιαστική αλυσίδα αποτελείται απ' όλα εκείνα τα μέρη που ικανοποιούν, άμμεσα ή έμμεσα τις απαιτήσεις ενός πελάτη. Τα στάδια της εφοδιαστικής αλυσίδας συμπεριλαμβάνουν Πελάτες, Εταιρίες διανομής, Κατασκευαστές και Προμηθευτές πρώτων υλών.

Μέσα σε κάθε οργανισμό (π.χ. Κατασκευαστές), η εφοδιαστική αλυσίδα συμπεριλαμβάνει όλες εκείνες τις λειτουργίες, που απαιτούνται για να εκπληρωθεί η ανάγκη-ζήτηση του πελάτη. Έτσι οι δραστηριότητες της εφοδιαστικής αλυσίδας μετατρέπουν φυσικούς πόρους, ακατέργαστα υλικά, σε ένα κατεργασμένο τελικό προιόν το οποίο παραδίδεται στον τελικό πελάτη.

2.2 Logistics

Logistics αποκαλείται η διαδικασία σχεδιασμού, εφαρμογής και ελέγχου αποτελεσματικότητας, αποδοτικής διακίνησης, αποθήκευσης πρώτων υλών, και καταγραφής αποθέματος, η οποία αξιοποιεί τα τελικά προιόντα και όλες τις πληροφορίες που σχετίζονται με αυτά, από το σημείο προέλευσης εώς το σημείο κατανάλωσης, με σκοπό την ικανοποίηση των αναγκών των πελατών.

Οι υπηρεσίες Logistics περιλαμβάνουν φυσικές δραστηριότητες, όπως μετακίνηση και αποθήκευση καθώς και μη φυσικές δραστηριότητες, όπως σχεδιασμός εφοδιαστικής αλυσίδας, επιλογή υπεργολάβων, διαπραγματεύσεις ναύλων κ.λ.π. Οι περισσότερες δραστηριότητες του logistics είναι διπλής κατεύθυνσης.

2.2.1 Ο ρόλος των μεταφορικών υπηρεσιών στην διαχείρηση logistics

Η διαδικασία των μεταφορικών υπηρεσιών καθορίζει την αποτελεσματική μεταφορά προιόντων. Ένα καλό δίκτυο μεταφορών στις δραστηριότητες logistics παρέχει αποτελεσματικότητα, μειώνει τα κόστη της επιχείρησης και προωθεί την ποιότητα της υπηρεσίας-προιόντος. Η πρόοδος σε τεχνικές και διαχείρηση βελτιώνει τον όγκο μεταφοράς, την ταχύτητα παράδοσης, την αποτελεσματικότερη χρήση εγκαταστάσεων και την εξοικονόμηση ενέργειας.

Οι μεταφορές είναι πρωταρχικές δραστηριότητες της εφοδιαστικής αλυσίδας που απορροφούν το μεγαλύτερο κόστος στην επιχείρηση. Έχει παρατηρηθεί ότι απορροφά μεταξύ του ενός τρίτου και των δύο τρίτων του συνολικού κόστους της εφοδιαστικής αλυσίδας.

Οι μεταφορές χωρίζονται σε δύο μέρη,

- σε εσωτερικές που περιλαμβάνουν την μεταφορά πρώτων υλών από τις πηγές προς τα εργοστάσια, τμημάτων των τελικών προιόντων ανάμεσα σε διάφορα εργοστάσια της εταιρείας και των τελικών προιόντων από τα εργοστάσια στις αποθήκες και στα σημεία πώλησης.

-σε εξωτερικές που περιλαμβάνουν την μεταφορά των τελικών προιόντων απο τις αποθήκες στους πελάτες άμεσα ή διαμέσου κέντρων διανομής.

Μερικά από τα σημαντικότερα προβλήματα σχεδιασμού μεταφορών είναι η δρομολόγηση των οχημάτων δηλαδή την επιλογή βέλτιστων διαδρομών λαμβάνοντας υπόψην τη δομή του δικτύου, τις αποστάσεις και την χωρητικότητα των διαδρομών.

Η σωστή διαχείρηση του συστήματος μεταφορών εχει σαν αποτέλεσμα το τελικό προιόν να βρίσκεται την σωστή στιγμή, στο σωστό μέρος, ικανοποιώντας τις ανάγκες και την ζήτηση των πελατών. Το γεγονός αυτό οφελεί όχι μόνο την ποιότητα εξυπηρέτησης αλλά αυξάνει και την ανταγωνιστικότητα της εταιρείας [2].

2.2.2 City Logistics / Εφοδιαστική εντός πόλης

Εφοδιαστική εντός πόλης αποκαλείται η διαδικασία βελτιστοποίησης της εφοδιαστικής και των υπηρεσιών μεταφοράς από ιδιωτικές εταιρίες με χρήση ανεπτυγμένων συστημάτων πληροφοριών σε αστικές περιοχές λαμβάνοντας υπόψην την κυκλοφορία σε οδικά δίκτυα, την κυκλοφοριακή συμφόρηση, την ασφάλεια και την εξοικονόμηση ενέργειας μέσα στα πλαίσια της αγοράς και της οικονομίας.

Είναι μια τεχνοτροπία που προσπαθεί να ενσωματώσει τους ήδη υπάρχων πόρους για να επιλύσει τις δυσκολίες που προκαλούνται από τον υπερπληθυσμό και την αυξανώμενη ιδιοκτησία οχημάτων σε αστικές περιοχές.

Σε πολλές πόλεις όπως στο Τόκυο, στην Μπανγκόκ, η ανταγωνιστικότητα των επιχειρήσεων μειώθηκε εξαιτίας της κυκλοφοριακής συμφόρησης και του χαμηλού επιπέδου μεταφορικών υπηρεσιών. Η εφοδιαστική εντός πόλεως παρέχει την ευκαιρία για ανάπτυξη και χρήση καινοτόμων λύσεων που βελτιώνουν το βιωτικό επίπεδο στις αστικές περιοχές.

Επιπλέον μειώνει τα κόστη μεταφοράς και τις αρνητικές επιδράσεις που υπόκεινται στο περιβάλλον [3]. Ένα από τα εργαλεία της εφοδιαστικής αλυσίδας είναι το Intelligent Transport System (ITS).

2.3 Intelligent Transport System (ITS)

Οι εφαρμογές του ITS στις υπηρεσίες μεταφορών είναι ευρέως διαδεδομένες. Οι πιο κοινές τεχνικές στην εφοδιαστική πόλης είναι το Global Positioning System (GPS), Geographic Information Systems (GIS) και ανεπτυγμένα πληροφοριακά συστήματα. Το GPS, παρέχει την θέση-στίγμα κάθε οχήματος βοηθώντας έτσι στην παρακολούθηση των οχημάτων και στην αποστολή αυτών. Το GIS, παρέχει τα βασικά γεωγραφικά δεδομένα κάθε περιοχής βοηθώντας τους διανομείς να οργανώνουν πιο γρήγορα και εύκολα τα δρομολόγια που θα ακολουθήσουν. Τα ανεπτυγμένα πληροφοριακά συστήματα, παρέχουν πραγματικού χρόνου (real-time) δεδομένα που οδηγούν στην αναπροσαρμογή των δρομολογίων. Ολα μαζί παρέχουν αποτελεσματικές και προσαρμοστικές μεταφορικές υπηρεσίες σε οποιεσδήποτε συνθήκες διότι μειώνονται τα επιπλέον δρομολόγια, με αποτέλεσμα να αυξάνεται η ποιότητα εξυπηρέτησης και να αυξάνεται ο όγκος τελικών προιόντων που μεταφέρεται.

ΚΕΦΑΛΑΙΟ 3

3.1 Προβλήματα Βελτιστοποίησης της Εφοδιαστικής Αλυσίδας

3.1.1. Προβλήματα Δρομολόγησης Οχημάτων / Vehicle Routing Problem

Τα Προβλήματα Δρομολόγησης Οχημάτων, αφορούν την διανομή προιόντων σε μια χρονική περίοδο, σε ένα σύνολο από πελάτες, από ένα σύνολο οχημάτων τα οποία έχουν σαν αφετηρία τους μια συγκεκριμένη αποθήκη. Το σύνολο οχημάτων χρησιμοποιείται από ένα συγκεκριμένο αριθμό οδηγών, όπου πραγματοποιούν τις διαδρομές τους, χρησιμοποιώντας ένα συγκεκριμένο οδικό δίκτυο. Γενικά, η επίλυση του προβλήματος δρομολόγησης οχημάτων έχει σαν αποτέλεσμα τον καθορισμό ενός συνόλου από διαδρομές. Κάθε μια από αυτές ξεκινά και καταλήγει σε μια αποθήκη, ικανοποιώντας τις απαιτήσεις των πελατών, χωρίς να παραβιάζεται κάποιος απο τους περιορισμούς και έχοντας ελαχιστοποιήσει το κόστος διανομής [4].

Ένα απο τα πιο βασικά και διασημότερο όλων των προβλημάτων δρομολόγησης είναι αυτό του πλανόδιου πωλητή - Traveling Salesman Problem (TSP).

3.1.2 Στατικό Πρόβλημα Δρομολόγησης Οχημάτων / The Static Vehicle Routing Problem

Σε ένα στατικό πρόβλημα δρομολόγησης οχημάτων όλες οι σχετικές πληροφορίες στο σχεδιασμό δρομολογίων θεωρούνται γνωστές απ'ο τον σχεδιαστή πριν ξεκινήσει το δρομολόγιο. Όλες οι σχετικές πληροφορίες των δρομολογίων δεν αλλάζουν/μεταβάλλονται απο την στιγμή που έχουν σχεδιασθεί τα δρομολόγια.

Ο Δρ. Psaraftis [5] κατηγοριοποιεί ένα πρόβλημα ως στατικό πρόβλημα δρομολόγησης οχημάτων όταν το αποτέλεσμα του είναι ένα σύνολο προκαθορισμένων δρομολογίων, μιας στάνταρ μοντελοποίησης, που δεν επαναβελτιστοποιούνται και προκύπτουν από δεδομένα που εισάγονται στο πρόβλημα, τα οποία δεν μεταβάλονται κατά τη διάρκεια του χρόνου.

3.1.3 Δυναμικό πρόβλημα δρομολόγησης οχημάτων

Σε ένα δυναμικό πρόβλημα δρομολόγησης οχημάτων, κάποιες σχετικές πληροφορίες στο σχεδιασμό δρομολογίων δεν θεωρούνται γνωστές από τον σχεδιαστή πριν ξεκινήσει το

δρομολόγιο. Κάποιες άλλες σχετικές πληροφορίες των δρομολογίων μεταβάλλονται απο την στιγμή που έχουν σχεδιασθεί τα δρομολόγια.

Ο Δρ. Psaraftis [5] κατηγοριοποιεί ένα πρόβλημα ως δυναμικό πρόβλημα δρομολόγησης οχημάτων όταν το αποτέλεσμα του δεν είναι κυρίως ένα σύνολο προκαθορισμένων δρομολογίων αλλά μια πολιτική, η οποία περιγράφει πως τα δρομολόγια εξελίσσονται συναρτήσει των δεδομένων που εισάγονται στο πρόβλημα, τα οποία είναι χρονικά μεταβαλλόμενα.

Προφανώς το δυναμικό πρόβλημα δρομολόγησης οχημάτων (DVRP) είναι πιο πολύπλοκο από το συμβατικό στατικό πρόβλημα δρομολόγησης οχημάτων (VRP). Το δυναμικό πρόβλημα δρομολόγησης οχημάτων (DVRP) χρησιμοποιεί αλγόριθμους στους οποίους εισάγονται χρονικά μεταβαλλόμενα δεδομένα έτσι ώστε να εξυπηρετούνται οι απαιτήσεις που παρουσιάζονται εκείνη τη στιγμή, εαν αυτό είναι δυνατόν να συμβεί. Καθώς το συμβατικό στατικό πρόβλημα δρομολόγησης οχημάτων (VRP) είναι NP βαθμού δυσκολίας, δεν είναι πάντα εφικτό να βρεθεί μια βέλτιστη λύση σε ρεαλιστικά προβλήματα σε λογικό υπολογιστικό χρόνο. Συμπερασματικά και το δυναμικό πρόβλημα δρομολόγησης οχημάτων (DVRP) είναι NP βαθμού δυσκολίας, αφού κάθε φορά που παρουσιάζεται και εισάγεται στο σύστημα μια καινούργια απαίτηση, προκύπτει ένα νέο στατικό πρόβλημα δρομολόγησης οχημάτων (VRP) για επίλυση. Οταν αναλύουμε ένα δυναμικό πρόβλημα δρομολόγησης (DVRP), συνήθως χρησιμοποιούμε τυχαία αναπαραγόμενα δεδομένα και όχι πραγματικά. Υπάρχουν δύο λόγοι για αυτό.

Πρώτον όταν χρησιμοποιούμε τυχαία αναπαραγόμενα δεδομένα μας δίνεται η δυνατότητα μιας εις βάθος ανάλυσης του προβλήματος αφού μπορούν να κατασκευαστούν με τέτοιο τρόπο ώστε να διευθύνονται θέματα και να εξετάζονται πτυχές του προβλήματος που επιλέγουμε εμείς.

Δεύτερον η πλειοψηφία των δυναμικών προβλημάτων δρομολόγησης (DVRP), στην πραγματικότητα δεν μπορούν να συλλάβουν όλα τα δεδομένα που χρειάζονται για εις βάθος ανάλυση, των συγκεκριμένων προβλημάτων δρομολόγησης. Για παράδειγμα, τα γεωγραφικά στίγματα των οχημάτων,την στιγμή που προκύπτουν νέες απαιτήσεις (ζήτηση), είναι συχνά δεδομένα που λείπουν στην πραγματικότητα κατά την μελέτη αυτών των προβλημάτων.

Γενικά, όσο πιο δυναμικό ένα σύστημα, τόσο πιο δύσκολα μπορούμε να παράγουμε μια γρήγορη αντίδραση. Γι’ αυτό στη δημιουργία ενός δυναμικού συστήματος παίζει μεγάλο

ρόλο η επιλογή των κατάλληλων αλγορίθμων και μοντέλων για την υποστήριξη αποφάσεων σε πολλά και διαφορετικά σενάρια.

3.1.4 Δυναμικό πρόβλημα δρομολόγησης οχημάτων στην Εφοδιαστική

Τα τελευταία χρόνια κατέσθει σημαντικό για πολλές εταιρείες διανομής ανά την υφήλιο να αξιοποιούν τις on line πληροφορίες, μέσω των τηλεπικοινωνιών και των υπολογιστών, με σκοπό να γίνουν πιο αποτελεσματικές. Ο χαμηλού κόστους σχεδιασμός δρομολόγησης οχημάτων παίζει σημαντικό ρόλο στην παγκόσμια αγορά, αν αναλογισθούμε ότι τα έξοδα διανομής σε κάθε χώρα ανέρχονται στο 10%-15% του ΑΕΠ της. Ακόμη και μια μικρή βελτίωση στον τομέα διανομής είναι εξαιρετικά σημαντική για τις βιομηχανίες.

Στην πραγματικότητα, ένα μεγάλο μέρος των προβλημάτων δρομολόγησης τα οποία μοντελοποιούνται ως στατικά, δεν εμπεριέχουν δυναμικά στοιχεία και δεν μπορούν να ανταπεξέρχονται σε μεγάλο βαθμο σε πραγματικές καταστάσεις. Για παράδειγμα οι χρόνοι εξυπηρέτησης και οι χρόνοι δρομολογίων στην πραγματικότητα αλλάζουν-μεταβάλλονται (φαινόμενο θορύβου). Το γεγονός αυτό υποδεικνύει τη μη τήρηση των προσχεδιασμένων δρομολογίων, λόγω των νέων προσωρινών συνθηκών που ανέκυψαν στο σύστημα. Έτσι αυτό που θεωρούνταν στο σύστημα μας ως η βέλτιστη λύση (βέλτιστο δρομολόγιο), τώρα εξελίσσεται σε υποβέλτιστη λύση του προβλήματος.

3.2 Travelling Salesman Problem

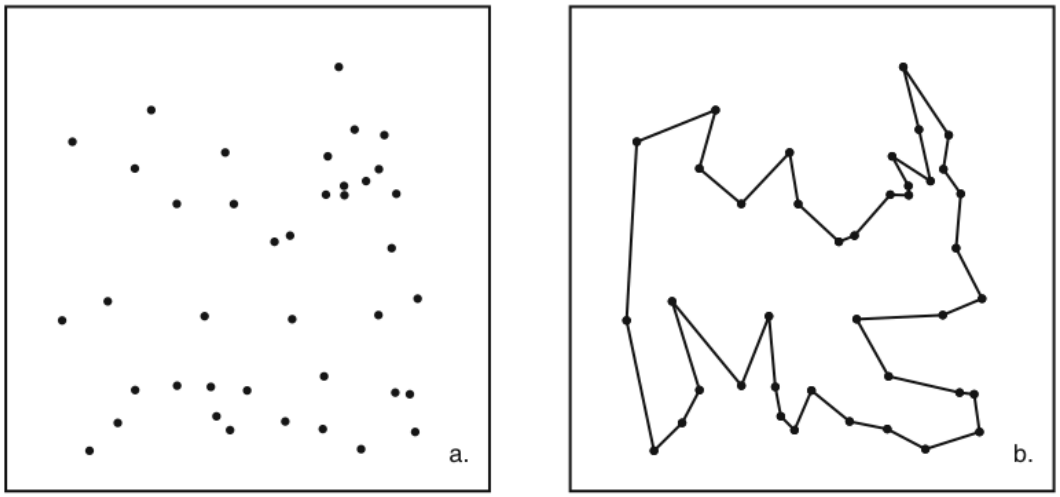
Το πρόβλημα του πλανόδιου πωλητή [6], αφορά την εύρεση μιας διαδρομής που πρέπει να ακολουθήσει ένας πωλητής, ο οποίος κινείται σε ένα δίκτυο πόλεων, με σκοπό να επισκεπτεί το σύνολο αυτών, ελαχιστοποιώντας το κόστος της διαδρομής που ακολουθεί.

Το δίκτυο των πόλεων αναπαρίσταται με ένα γράφο, οι κόμβοι του οποίου αναπαριστούν τις πόλεις, ενώ οι ακμές του το οδικό δίκτυο με το οποίο συνδέονται. Για την επίλυση του συγκεκριμένου προβλήματος, έχουν προταθεί διάφοροι αλγόριθμοι οι οποίοι ως ένα βαθμό επιλύουν ικανοποιητικά το πρόβλημα. Ωστόσο, η πολυπλοκότητα επίλυσης του προβλήματος, ιδιαίτερα όταν αυτό αφορά μεγάλους γράφους, είναι μη- πολυωνυμική καθώς η μόνη μέθοδος που εγγυάται τη λύση του προβλήματος συνίσταται στον εξαντλητικό υπολογισμό του κόστους, του συνόλου των πιθανών διαδρομών και στην επιλογή της

συντομότερης. Αυτό όμως καθίσταται εξαιρετικά πολύπλοκο για μεγάλους γράφους, διότι αυξάνεται πολύ ο χρόνος επεξεργασίας των δεδομένων. Συνεπώς, δεν υφίσταται μια συγκεκριμένη ενιαία μέθοδος που να λύνει οποιαδήποτε περίπτωση του προβλήματος του πλανόδιου πωλητή.

Οι διαδικασίες επίλυσης βασίζονται σε ευρετικούς κανόνες, οι οποίες όμως λύνουν προσεγγιστικά το πρόβλημα καθώς δε δίνουν τη βέλτιστη.

Μια ευρετική λύση του προβλήματος θα μπορούσε να διατυπωθεί ως ακολούθως: Έστω ένας κόμβος *ε*, ο οποίος επιλέγεται ως κόμβος εκκίνησης και ονομάζεται κόμβος *ν*. Εν συνεχεία, αναζητείται κατά προτεραιότητα η ακμή *(ν, u)* από τον κόμβο *ν* η οποία δεν έχει διαπεραστεί και έχει το ελάχιστο κόστος διάσχισης. Αφού, επισημανθεί ως διαπερασμένη, ο κόμβος *u* μετονομάζεται σε κόμβο *ν* και η διαδικασία επαναλαμβάνεται μέχρις ότου προσπελαθούν όλοι οι κόμβοι και ο αλγόριθμος επιστρέψει στο αρχικό σημείο εκκίνησης (σχήμα 3.1).



Σχήμα 3.1

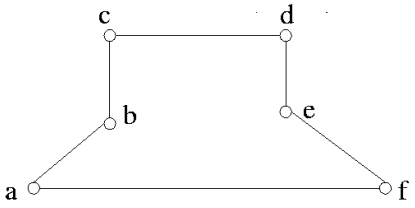
3.2.1 Κατηγορίες ευρετικών αλγορίθμων

Στις πιο πολλές περιπτώσεις οι αλγόριθμοι που χρησιμοποιούμε χαρακτηρίζονται ως **ευρετικοί** (heuristic), δηλαδή βρίσκουν λύσεις ανάμεσα στις πολλές εφικτές αλλά δεν υπάρχει εξ' αρχής εγγύηση ότι βρίσκουν μια βέλτιστη λύση. Άρα μπορούν να θεωρηθούν ως προσεγγιστικοί και μη ακριβείς αλγόριθμοι, συνήθως όμως βρίσκουν λύση "κοντά" στη βέλτιστη και τη βρίσκουν εύκολα και "γρήγορα".

- **Αλγόριθμοι απληστίας (greedy algorithms)**

Στην περίπτωση του TSP, ας αριθμήσουμε τις πόλεις από 1 μέχρι *n*, και ας είναι 1 η πόλη-βάση και *c(i,j)* το κόστος επίσκεψης από *i* σε *j*. Δεχόμαστε ότι είναι δυνατόν να συμβεί *c(i,j)* ≠ *c(j, i)*. Προφανώς είναι δυνατόν οι εφικτές λύσεις να είναι όσες και οι διατάξεις των *n-1* πόλεων (όλες πλην της βάσης), δηλαδή *(n-1)*!. Θα μπορούσε κάποιος να τις καθορίσει συστηματικά, να βρει για κάθε μια από αυτές το κόστος και να κρατήσει αυτή με το ελάχιστο. Αυτό απαιτεί τουλάχιστον *(n-1)*! βήματα.

Αν π.χ. είχαμε μόνο 21 πόλεις, βρίσκουμε *(n-1)! = 20!* βήματα. Αν κάθε βήμα απαιτούσε ένα χιλιοστό του δευτερολέπτου χρειάζονται περίπου 770 αιώνες συνεχούς υπολογισμού. Προφανώς η εξαντλητική εξέταση όλων των λύσεων πρέπει να απορριφθεί. Και επειδή δεν γνωρίζουμε άλλο σύντομο αλγόριθμο που να δίνει σίγουρα μια βέλτιστη λύση, χρησιμοποιούμε ένα ευρετικό αλγόριθμο. Σύμφωνα με αυτόν, όταν ο πωλητής βρίσκεται στην πόλη *i* επιλέγει ως επόμενη πόλη εκείνη την *j* για την οποία το κόστος *c(i,j)* είναι το μικρότερο μεταξύ όλων των *c(i,k),* όπου *k* είναι οι δείκτες των πόλεων που δεν έχει ακόμα επισκεφθεί ο πωλητής (σχήμα 3.2). Εννοείται ότι υπάρχει και κάποιος απλός κανόνας, π.χ. ελάχιστο *k*, για την περίπτωση "ισοπαλίας", όταν δηλαδή πάνω από μια πόλεις *k* δίνουν ελάχιστο κόστος. Πρόκειται για αλγόριθμο απληστίας που σε κάθε βήμα επιλέγει την πιο φθηνή επίσκεψη και αδιαφορεί αν αυτό μπορεί τελικά να οδηγήσει σε λάθος αποτέλεσμα.



Σχήμα 3.2

- **Προσεγγιστικοί αλγόριθμοι (approximation algorithms)**

Οι προσεγγιστικοί αλγόριθμοι προσπαθούν να λύσουν το πρόβλημα του TSP χρησιμοποιώντας επιπλέον πληροφορία.

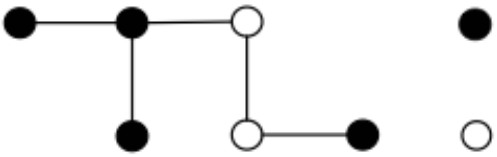
Η ευρετική μέθοδος που προτάθηκε από τον Christofides [7], ξεκινάει απο ένα ελάχιστο τανύον δένδρο (minimum spanning tree) και αφού το μετατρέψει σε κύκλο του Euler στην συνέχεια το μετατρέπει σε ένα κύκλο Hamilton.

Βήμα 1 Υπολογισμός ενός ελαχίστου τανύοντος δένδρου (σχήμα 3.3).

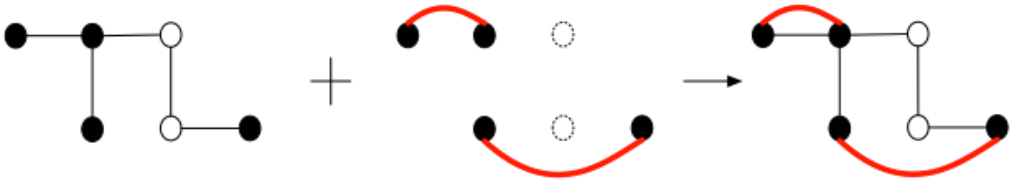
Βήμα 2 Υπολογισμός ενός τέλειου ταιριάσματος ελαχίστου βάρους για τους κόμβους μονού βαθμού του δένδρου και στη συνέχεια τους προσθέτουμε στο δένδρο έτσι ώστε να κάνουμε ενα Eulerian γράφο (σχήμα 3.4).

Βήμα 3 Υπολογίζουμε ένα κύκλο Euler για το συγκεκριμένο γράφο (σχήμα 3.5).

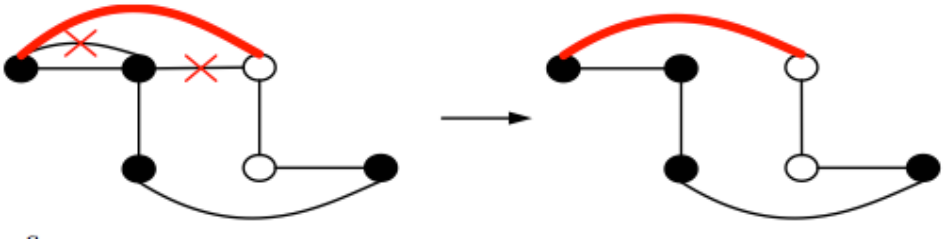
Βήμα 4 Μετατρέπουμε τον Eulerian γράφο σε Hamiltonian γράφο (σχήμα 3.5).



Σχήμα 3.3



Σχήμα 3.4

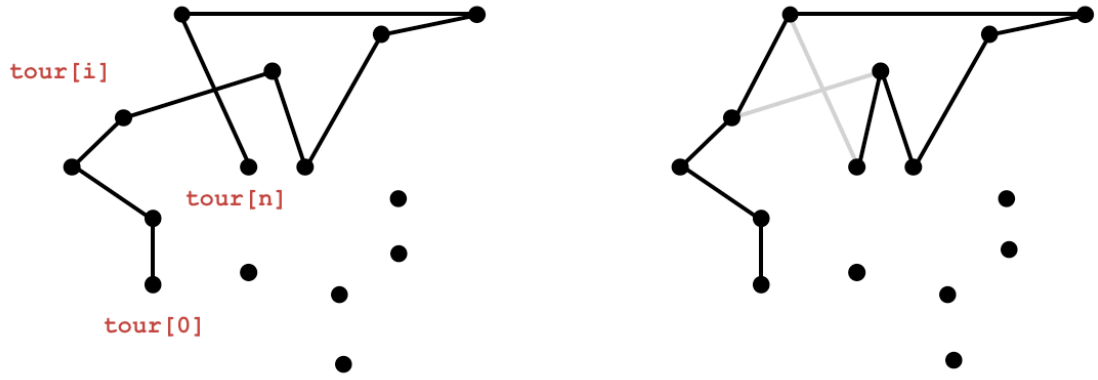


Σχήμα 3.5

- **Αλγόριθμοι τοπικής αναζήτησης (local search algorithms)**

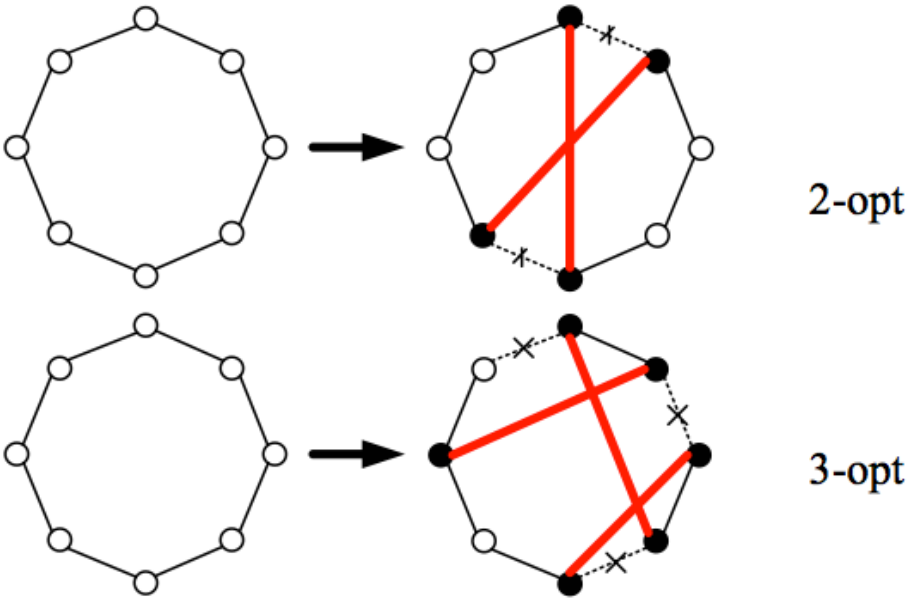
Οι αλγόριθμοι τοπικής αναζήτησης προσπαθούν από μια αρχική εφικτή λύση να βελτιώσουν τη λύση με κάποια μέθοδο αναζήτησης στη γειτονιά της λύσης.

Η μέθοδος 2 – opt [6] εξετάζει την αντικατάσταση ενός ζεύγους τόξων με δύο άλλα τόξα (σχήμα 3.4) έτσι ώστε να επιτευχθεί καλύτερο αποτέλεσμα. Η διαδικασία συνεχίζεται έως ότου το συνολικό κόστος του δρομολογίου να μη μειώνεται με τις αλλαγές των τόξων.



Σχήμα 3.4

Ομοίως η μέθοδος 3 – opt [6] εξετάζει την αντικατάσταση ενός ζεύγους τόξων με τρία άλλα τόξα (σχημα 3.5) έτσι ώστε να επιτευχθεί καλύτερο αποτέλεσμα. Η διαδικασία συνεχίζεται έως ότου το συνολικό κόστος του δρομολογίου να μη μειώνεται με τις αλλαγές των τόξων.



Σχημα 3.5

ΚΕΦΑΛΑΙΟ 4

4.1 Αλγόριθμος Concorde

Ο αλγόριθμος Concorde [8] είναι ένα υπολογιστικό πρόγραμμα, το οποίο επιλύει το συμμετρικό πρόβλημα του πλανώδιου πωλητή (TSP) και κάποια παρεμφερή προβλήματα που σχετίζονται με τη βελτιστοποίηση δικτύων. Ο κώδικας είναι γραμμένος στη γλώσσα προγραμματισμού ANSI C και ο αλγόριθμος Concorde είναι διαθέσιμος για ακαδημαϊκή έρευνα. Θεωρείται παγκοσμίως, ο πιο γρήγορος και αξιόπιστος λύτης προβλημάτων που ανήκουν στην κατηγορία TSP.

Ο λύτης Concorde TSP (Concorde's TSP solver) έχει χρησιμοποιηθεί για την επίτευξη των βέλτιστων λύσεων για την πλήρη σειρά περιπτώσεων της TSPLIB, η οποία δημιουργήθηκε απο τον Gerhard Reinelt και αποτελείται απο 110 περιπτώσεις προβλημάτων που ανήκουν στην κατηγορία προβλημάτων του TSP, εκ των οποίων το μεγαλύτερο εμπεριέχει 85900 πόλεις. Ακολουθούν σχηματικές επιλύσεις (σχήμα 4.1/σχήμα 4.2) δύο γνωστών προβλημάτων TSP που έχει επιλύσει ο αλγόριθμος Concorde :

Σχήμα 4.1: 15,112 cities in Germany

Σχήμα 4.2: 24,978 cities in Sweden

Η βιβλιοθήκη του αλγόριθμου Concorde περιλαμβάνει πάνω από 700 συναρτήσεις, επιτρέπωντας στους χρήστες να δημιουργήσουν εξειδικευμένες κωδικούς για προβλήματα που ανήκουν στην κατηγορία προβλημάτων του TSP. Όλες οι συναρτήσεις του αλγόριθμου Concorde είναι thread-safe για προγραμματισμό σε κοινόχρηστη μνήμη σε παράλληλα περιβάλλοντα. Η κύριος λύτης Concorde TSP (TSP solver) περιλαμβάνει κώδικα για τη λειτουργία του σε δίκτυα, τα οποία χρησιμοποιούν UNIX σταθμούς εργασίας.

Πλέον ο αλγόριθμος Concorde υποστηρίζει τον λύτη γραμμικού προγραμματισμού QSopt [9] (QSopt linear programming solver), με εκτελέσιμες εκδόσεις του αλγόριθμου Concorde με qsopt solver για τα λειτουργικά συστήματα Linux και Solaris. Ο κύριος σκοπός αυτού του προγράμματος είναι να παρέχει μια βιβλιοθήκη συναρτήσεων, η οποία μας δίνει την δυνατότητα να δημιουργήσουμε, να διαχειριστούμε και να επιλύσουμε προβλήματα γραμμικού προγραμματισμού (LP problems). Αυτός είναι ο λόγος που χρησιμοποιείται σε προβλήματα, όπως του πλανόδιου πωλητή (TSP) ή μεικτού-ακέραιου προγραμματσμού.

Επιπλέον ο λύτης γραμμικού προγραμματισμού QSopt (QSopt linear programming solver), είναι ένας ανεξάρτητος κώδικας, ο οποίος μπορεί να επιλύσει μεγάλης κλίμακας προβλήματα γραμμικού προγραμματισμού.

4.2 Μεθοδολογία αλγόριθμου Concorde

4.2.1 Ζώνες Ελέγχου

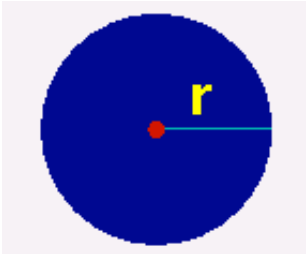
Πώς μπορούμε να ξέρουμε ότι το δρομολόγιο μας είναι καλό, ή καλύτερο από όλα τα υπόλοιπα δρομολόγια που δεν έχουν εξεταστεί; Μια απλή απάντηση είναι να εξετάσουμε όλα τα πιθανά δρομολόγια. Αλλά, όπως και σε πολλά άλλα παρόμοια προβλήματα, ο αριθμός των δρομολογίων-γύρων αυξάνεται τόσο γρήγορα, ώστε ακόμη και σε ένα μικρού μεγέθους πρόβλημα, για παράδειγμα ας πούμε 30 πόλεις, ο υπολογισμός τους είναι πάνω από τις δυνατότητες ενός σημερινού υπερ-υπολογιστή. Αυτό που χρειαζόμαστε είναι ένας πιο ουσιαστικός τρόπος ώστε να εξάγουμε συμπεράσματα για τα μήκη των διαδρομών-γύρων μέσα από ένα σύνολο σημείων.

Με άλλα λόγια, χρειαζόμαστε μεθόδους που μας επιτρέπουν να κάνουμε έγκυρες δηλώσεις της μορφής «ένα δρομολόγιο μέσα από αυτό το σύνολο των σημείων δεν μπορεί να έχει μήκος μικρότερο από 100», όπου το μήκος 100 μπορεί να αντικατασταθεί από ένα κατάλληλο μήκος B για το συγκεκριμένο παράδειγμα TSP που επιλύουμε. Ένας τέτοιος αριθμός B ονομάζεται κάτω φράγμα για το TSP, δεδομένου ότι δεσμεύει προς τα κάτω το μήκος των δρομολογίων μας [10].

Υπάρχουν μια σειρά από διαφορετικές στρατηγικές για την εύρεση καλών χαμηλότερων οριών για το TSP. Στον αλγόριθμο Concorde και σε σύγχρονους λύτες TSP η τεχνική οριοθέτησης που επιλέγεται είναι αυτή των G. Dantzig, R. Fulkerson, και S. Johnson [11] και βασίζεται στην επιλυσημότητα μαθηματικών μοντέλων τα οποία είναι γνωστά ως προβλημάτα γραμμικού προγραμματισμού. Θα παρουσιάσουμε την τεχνική, Dantzig-Fulkerson-Johnson, χρησιμοποιώντας μια γεωμετρική ερμηνεία που περιγράφεται από τον M. Juenger και WR Pulleyblank στη δεκαετία του 1980.

Ας υποθέσουμε το TSP πρόβλημα μας, το οποίο αποτελείται από ένα σύνολο σημείων στο επίπεδο και ότι το κόστος για να ταξιδέψουμε μεταξύ των δύο πόλεων είναι η Ευκλείδεια απόσταση μεταξύ των αντίστοιχων σημείων. Κατ 'αρχάς, σχεδιάζουμε ένα δίσκο ακτίνας r με κέντρο την πόλη 1 με τέτοιο τρόπο ώστε ο δίσκος να μην έρχεται σε επαφή με

κάποια από τις άλλες πόλεις. Οι Juenger και Pulleyblank ονομάζουν έναν τέτοιο δίσκο ως ζώνη ελέγχου (σχήμα 4.3).

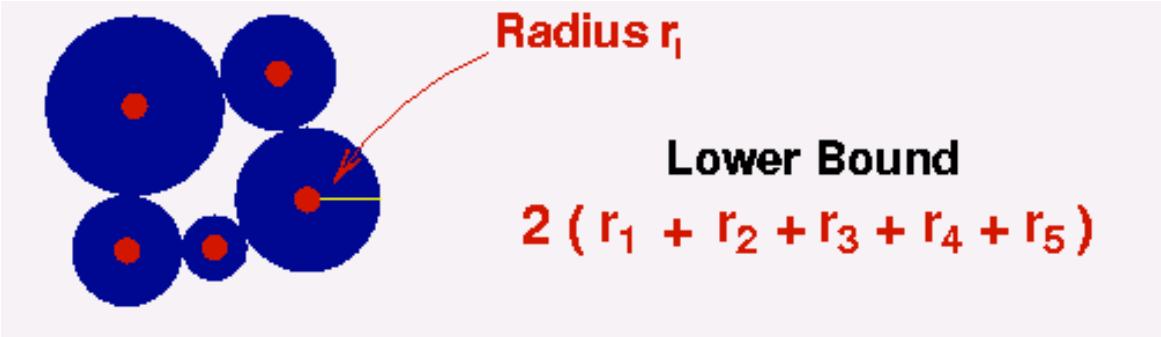


Σχήμα 4.3

Ο πωλητής, θα επισκεφθεί την πόλη 1 διανύοντας τουλάχιστον απόσταση r, για να φτάσει στην πόλη και τουλάχιστον απόσταση r για να φύγει από την (αφού η πόλη 1 απέχει τουλάχιστον απόσταση r από κάθε άλλη πόλη). Μπορούμε να συμπεράνουμε ότι κάθε περιοδεία έχει μήκος τουλάχιστον 2r, δηλαδή, μπορούμε να θέσουμε B = 2r και έχουμε ένα κάτω φράγμα για αυτό το TSP παράδειγμα.

Όταν κρίνουμε ένα κάτω όριο, όσο μεγαλύτερο είναι τόσο το καλύτερο. Δυστυχώς, για σχεδόν κάθε πρόβλημα TSP που μπορείτε να σκεφτείτε, το κάτω φράγμα 2r θα είναι μάλλον μικρό σε σύγκριση με το μήκος του κάθε δρομολογίου ανάμεσα στα σημεία. Γι' αυτό το λόγο δημιουργούμε ένα ξεχωριστό δίσκο για κάθε μία από τις πόλεις μας, εφ 'όσον οι δίσκοι δεν αλληλοεπικαλύπτονται. Με αυτό τον τρόπο θα έχουμε δύο φορές το άθροισμα των ακτίνων των δίσκων ως κατώτερο όριο για το TSP πρόβλημα μας.

Χρησιμοποιούμε ένα όνομα για την ακτίνα καθενός από τους δίσκους που δημιουργούμε, έτσι για κάθε πόλη i, συμβολίζουμε με ri την ακτίνα του δίσκου του. Για παράδειγμα, στην περίπτωση 5 πόλεων που ακολουθεί, καθορίζοντας τις τιμές των ακτίνων r1 r2 r3 r4 r5, προκύπτει το κατώτερο όριο για το συγκεκριμένο TSP πρόβλημα (σχήμα 4.4).



Σχήμα 4.4

Επειδή θέλουμε το κατώτερο όριο να είναι όσο το δυνατόν πιο μεγάλο, επιλέγουμε τις πέντε ακτίνες, με σκοπό την μεγιστοποίηση του διπλάσιου αθροίσματος αυτών, με την προϋπόθεση ότι οι δίσκοι δεν αλληλοεπικαλύπτονται. Η προϋπόθεση αυτή εκφράζεται ως

$$ri + rj <= dist(i,j)$$

όπου dist(i,j) συμβολίζει την απόσταση της πόλης i απο την πόλη j. Δηλαδή το άθροισμα των δύο ακτίνων μπορεί να είναι το πολύ ίσο με την απόσταση μεταξύ των πόλεων i και j.

Μοντελοποιώντας το πρόβλημα μας για την εύρεση κατώτερου ορίου απο τους δίσκους που έχουμε δημιουργήσει μπορεί να γραφθεί:

Maximize 2r1 + 2r2 + 2r3 + 2r4 + 2r5

Υ.Π

r1 + r2 <= dist(1,2)

r1 + r3 <= dist(1,3)

r1 + r4 <= dist(1,4)

r1 + r5 <= dist(1,5)

r2 + r3 <= dist(2,3)

r2 + r4 <= dist(2,4)

r2 + r5 <= dist(2,5)

r3 + r3 <= dist(3,4)

r3 + r3 <= dist(3,5)

r4 + r5 <= dist(4,5)

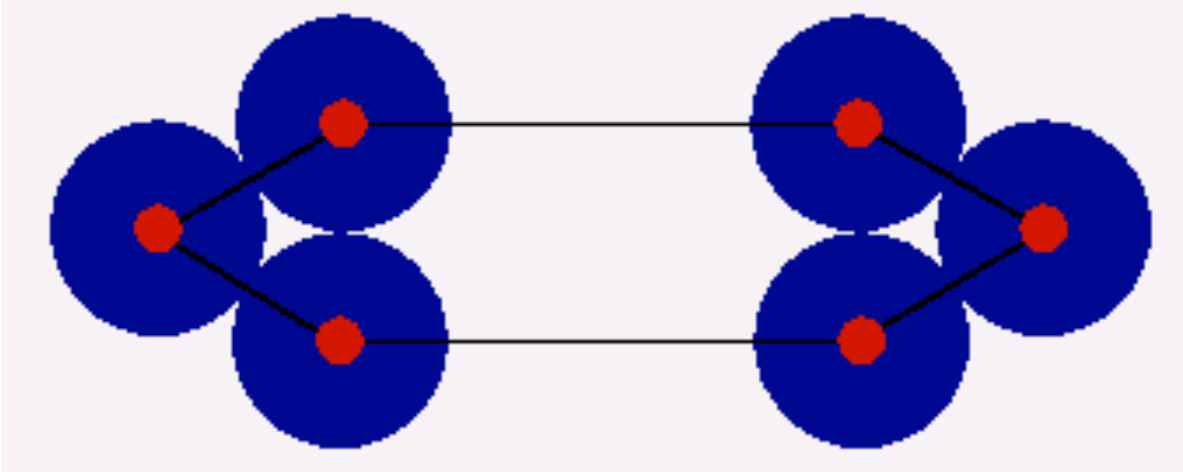
r1 >= 0, r2 >= 0, r3 >= 0, r4 >= 0, r5 >= 0

Αυτό είναι ένα παράδειγμα ενός προβλήματος γραμμικού προγραμματισμού (Linear Programming). Τα βασικά χαρακτηριστικά του οποίου είναι, η μεγιστοποίηση ενός σταθμισμένου αθροίσματος με κάποιες άγνωστες ποσότητες (στην περίπτωσή μας, οι ακτίνες των δίσκων), υπό περιορισμών, οι οποίοι εκφράζονται ως διάφορα σταθμισμένα ποσά με συγκεκριμένες τιμές (στην περίπτωσή μας, περιορισμοί μη-επικαλυπτόμενων δίσκων υπό την συνθήκη μη αρνητικής ακτίνας).

Ο Γραμμικός προγραμματισμός είναι ένα πολύ σημαντικό μαθηματικό μοντέλο, που έχει εκτεταμένες εφαρμογές. Ένα σημαντικό χαρακτηριστικό των μοντέλων Γραμμικού Προγραμματισμού (LP) είναι ότι ακόμη και για πολύ μεγάλο αριθμό άγνωστων τιμών, το πρόβλημα μπορεί να επιλυθεί αποτελεσματικά με μία ποικιλία νέων και παλαιών μεθόδων.

Στο πλαίσιο του προβλήματος του πλανώδιου πωλητή TSP, η πιο σημαντική από αυτές τις τεχνικές, είναι εκείνη που δημιουργήθηκε από τον George Dantzig στα 1940 και είναι ευρέως γνωστή ως μέθοδος Simplex. Μάλιστα το 2000, η μέθοδος simplex συμπεριλήφθηκε στη λίστα των 10 κορυφαίων Αλγόριθμων του αιώνα.

Προκύπτει όμως ένα σημαντικό θέμα σχετικά με την ποιότητα του κατώτερου ορίου από την ομάδα δίσκων που έχει επιλεχθεί. Το παράδειγμα των 5 πόλεων που δόθηκε παραπάνω δεν ανταποκρίνεται συνήθως στις απαιτήσεις του προβλήματός μας, διότι το κατώτερο όριο που παίρνουμε είναι σημαντικά μικρότερο της βέλτιστης διαδρομής.

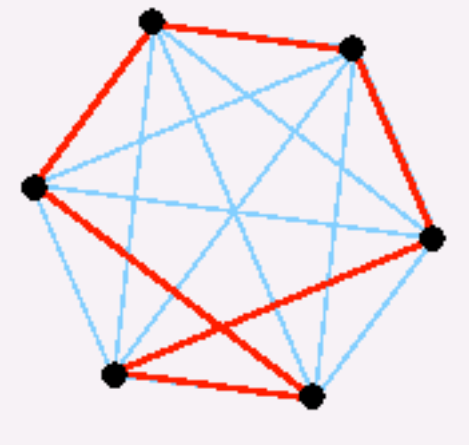


Σχήμα 4.5

Το πρόβλημα στο άνωθεν σχήμα (σχήμα 4.5) είναι ότι λόγω της συνθήκης μη αλληλοεπικάλυψης δίσκων, το κενό μεταξύ των δύο δίσκων δεν μπορεί να καλυφθεί. Πρέπει λοιπόν να εισάγουμε μια δομή η οποία θα εκμεταλευθεί το γεγονός ότι οποιοδήποτε διαδρομή θα διασχίσει το συγκεκριμένο κενό τουλάχιστον 2 φορές. Γεγονός που μας οδηγεί στους περιορισμούς εξάλειψης υποκύκλων (*subtour-elimination constraint*).

4.2.2 Εξάλειψη Υπόκυκλων

Ξεκινάμε με ένα παράδειγμα 6 πόλεων (σχήμα 4.6). Το δρομολόγιο μας αντιστοιχεί στην επιλογή 6 εκ των 15 πιθανών δρόμων μεταξύ των 6 σημείων.

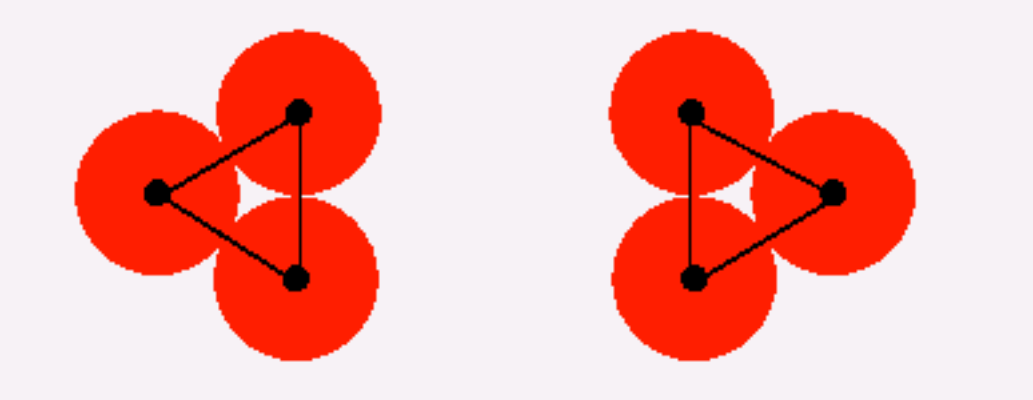


Σχήμα 4.6

Χαρακτηρίζουμε τις πόλεις 1,2,3,...,n , όπου x(i,j) για δύο πόλεις i και j, δηλώνει αν επιλέγουμε ή όχι την συγκεκριμένη διαδρομή στο δρομολόγιο μας.
Άρα x(i,j) = 1 όταν χρησιμοποιούμε την διαδρομή
Και x(i,j) = 0 όταν δεν χρησιμοποιούμε την διαδρομή
Το συνολικό μήκος διαδρομής θα είναι
Tour Length = dist(1,2)x(1,2) + dist(1,3)x(1,3) + ... + dist(n-1,n)x(n-1,n)
Έχουμε την αντικειμενική μας συνάρτηση πλέον (σταθμισμένο αθροισμα) και μπορούμε να διαμορφώσουμε το γραμμικό προβλημα το οποίο είναι:

Minimize dist(1,2)x(1,2) + dist(1,3)x(1,3) + ... + dist(4,5)x(4,5)
Subject To
x(1,2) + x(1,3) + x(1,4) + x(1,5) >= 2
x(1,2) + x(2,3) + x(2,4) + x(2,5) >= 2
x(1,3) + x(2,3) + x(3,4) + x(3,5) >= 2
x(1,4) + x(2,4) + x(3,4) + x(4,5) >= 2
x(1,5) + x(2,5) + x(3,5) + x(4,5) >= 2
x(1,2) >= 0, x(1,3) >= 0, ... , x(4,5) >= 0

Παρατηρώντας το πρόβλημα μας επανερχόμαστε στην περίπτωση που αναφέραμε άνωθεν. Έχουμε δύο ομάδες πόλεων όπου οι μη επικαλυπτόμενοι δίσκοι δεν καλύπτουν το ενδιάμεσο κενό (σχήμα 4.7).



Σχήμα 4.7

Αρχικά βελτιώνουμε το κατώτερο όριο χαλαρώνοντας τους περιορισμούς μας όπου προκύπτει το νέο γραμμικό πρόβλημα.

Minimize dist(1,2)x(1,2) + dist(1,3)x(1,3) + ... + dist(4,5)x(4,5)

Subject To

x(1,2) + x(1,3) + x(1,4) + x(1,5) = 2
x(1,2) + x(2,3) + x(2,4) + x(2,5) = 2
x(1,3) + x(2,3) + x(3,4) + x(3,5) = 2
x(1,4) + x(2,4) + x(3,4) + x(4,5) = 2
x(1,5) + x(2,5) + x(3,5) + x(4,5) = 2
x(1,2) >= 0, x(1,3) >= 0, ... , x(4,5) >= 0
x(1,2) <= 1, x(1,3) <= 1, ... , x(4,5) <= 1

Παρατηρούμε ότι κάθε δρομολόγιο θα περιλαμβάνει τουλάχιστον δύο δρόμους που θα ενώνουν τις δύο ομάδες πόλεων. Κατά συνέπεια μπορούμε να προσθέσουμε έναν ακόμη περιορισμό που αναφέρει ότι το άθροισμα των αντίστοιχων μεταβλητών x(i,j), πρέπει να είναι τουλάχιστον δύο.

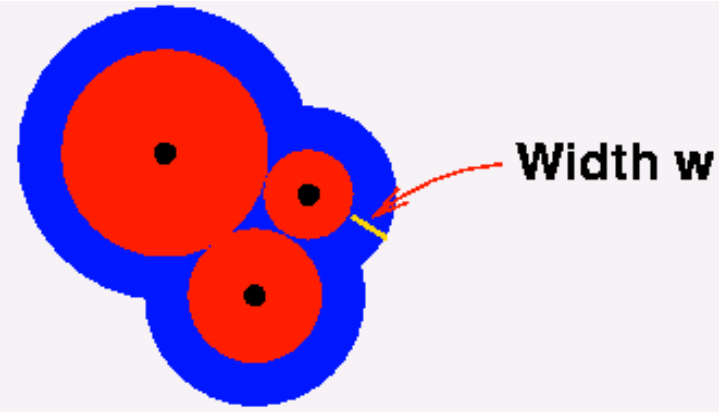
Γενικότερα, ας ονομάσουμε S μια οποιαδήποτε ομάδα πόλεων αποτελούμενη απο τουλάχιστον 2 και το πολύ n-1 πόλεις, (δηλαδή, S δεν μπορεί να είναι το σύνολο των πόλεων στο TSP) και με T την ομάδα που αποτελείται απ΄ τις υπόλοιπες πόλεις. Για να απαγορεύσουμε την κυκλική διαδρομή/υπόκυκλο (subtour) μέσω των πόλεων της ομάδας S, προσθέτουμε την

συνθήκη/περιορισμό, ότι το άθροισμα των μεταβλητών που αντιστοιχούν στους δρόμους από την ομάδα S στην ομάδα T, πρέπει να είναι τουλάχιστον 2.

$$\Sigma(x(i,j) : \text{exactly 1 of i and j is in S)) >= 2}$$

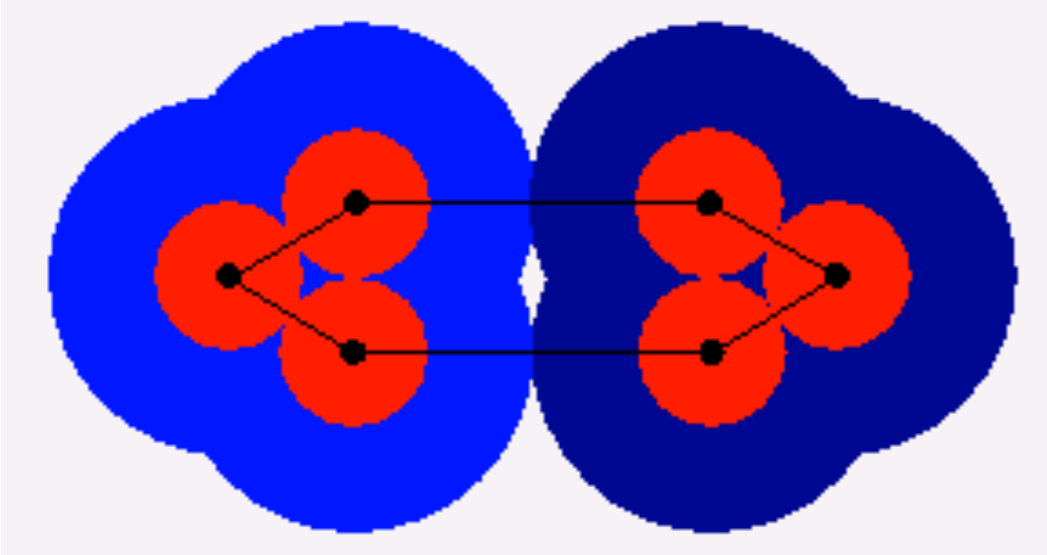
Αυτός ο περιορισμός αποσκοπεί στην εξάλειψη των κυκλικών διαδρομών (υπόκυκλων) για την ομάδα πόλεων S. Το κατώτερο όριο που προκύπτει από το νέο μας γραμμικό πρόβλημα είναι πολύ καλύτερο από το αρχικό μας [12].

Τώρα πρέπει να χρησιμοποιήσουμε κάποια μέθοδο για να ξεπεράσουμε το πρόβλημα του κενού που υπάρχει ανάμεσα στις δύο ομάδες πόλεων. Αυτή η μέθοδος είναι των Juenger και Pulleyblank, η οποία συμπληρώνει την μέθοδο των ζώνων ελέγχου. Η ιδέα είναι η εξής. Δωθέντος της ομάδας S, σχεδιάζουμε μια γεωγραφική περιοχή η οποία χωρίζει την ομάδα πόλεων S από τις πόλεις που δεν ανήκουν στην ομάδα S. Δεδομένου ότι κάθε δρομολόγιο πρέπει να επισκεφθεί την ομάδα πόλεων S, προσθέτουμε κατά δύο φορές το πλάτος της γεωγραφικής περιοχής που σχεδιάσαμε στο κατώτερο όριο.



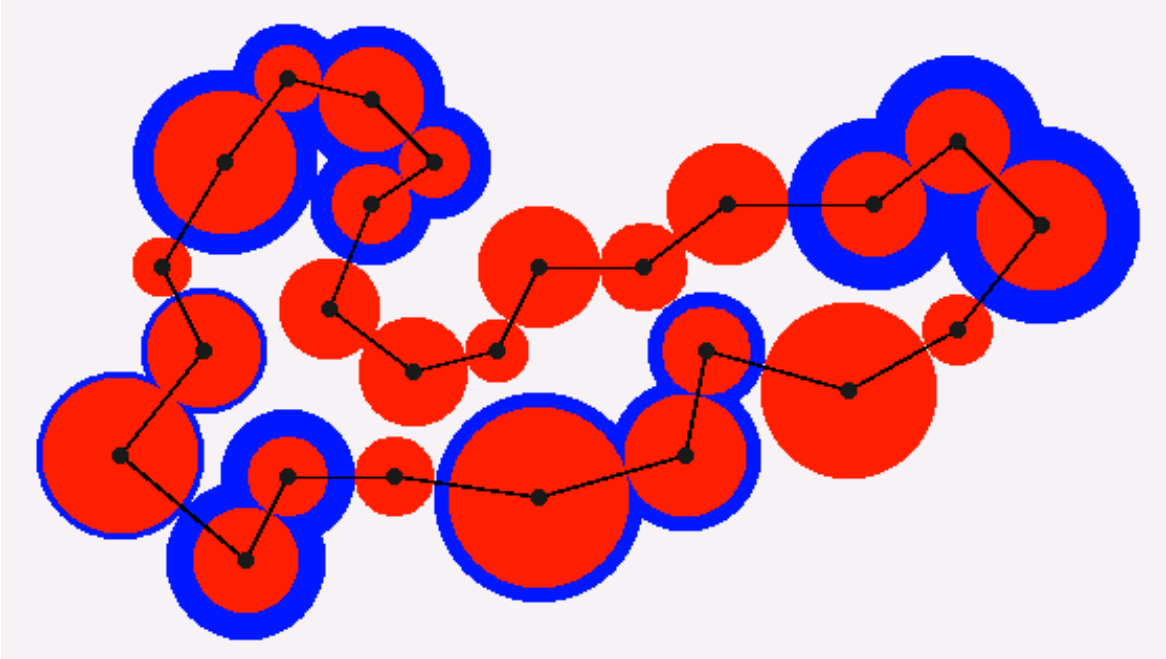
Σχήμα 4.8

Αυτές οι γεωγραφικές περιοχές (σχήμα 4.8) καλούνται τάφροι (moats) και μας επιτρέπουν να καλύψουμε τα κενά που υπάρχουν ανάμεσα στις ζώνες ελέγχου. Ακολουθεί η εφαρμογή της μεθόδου (σχήμα 4.9) στο παράδειγμα των δύο ομάδων πόλεων που δείξαμε σχηματικά άνωθεν, όπου παρουσιάζονται δύο τάφροι, αν και στο συγκεκριμένο παράδειγμα μία τάφρος αρκεί.



Σχήμα 4.9

Παρακάτω δίνεται ένα παράδειγμα του TSP στο οποίο δίνεται σχηματικά (σχήμα 4.10) το βέλτιστο δρομολόγιο που προκύπτει μέσω των συγκεκριμένων μεθόδων που αναφέραμε.



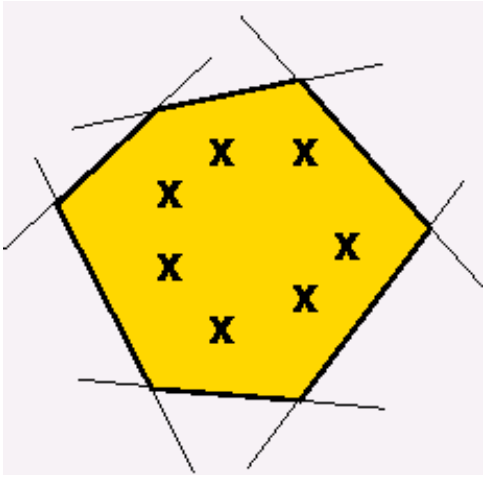
Σχήμα 4.10

4.2.3 Cutting Planes

Η πρόσθεση περιορισμών λόγω της μεθοδολογίας εξάλειψης υπόκυκλων στο πρόβλημα TSP, καθιστά την λύση του από ένα λύτη γραμμικού προγραμματισμού (LP solver) αρκετά δύσκολη, διότι αυξάνεται σε μεγάλο βαθμό ο αριθμός περιορισμών. Για να

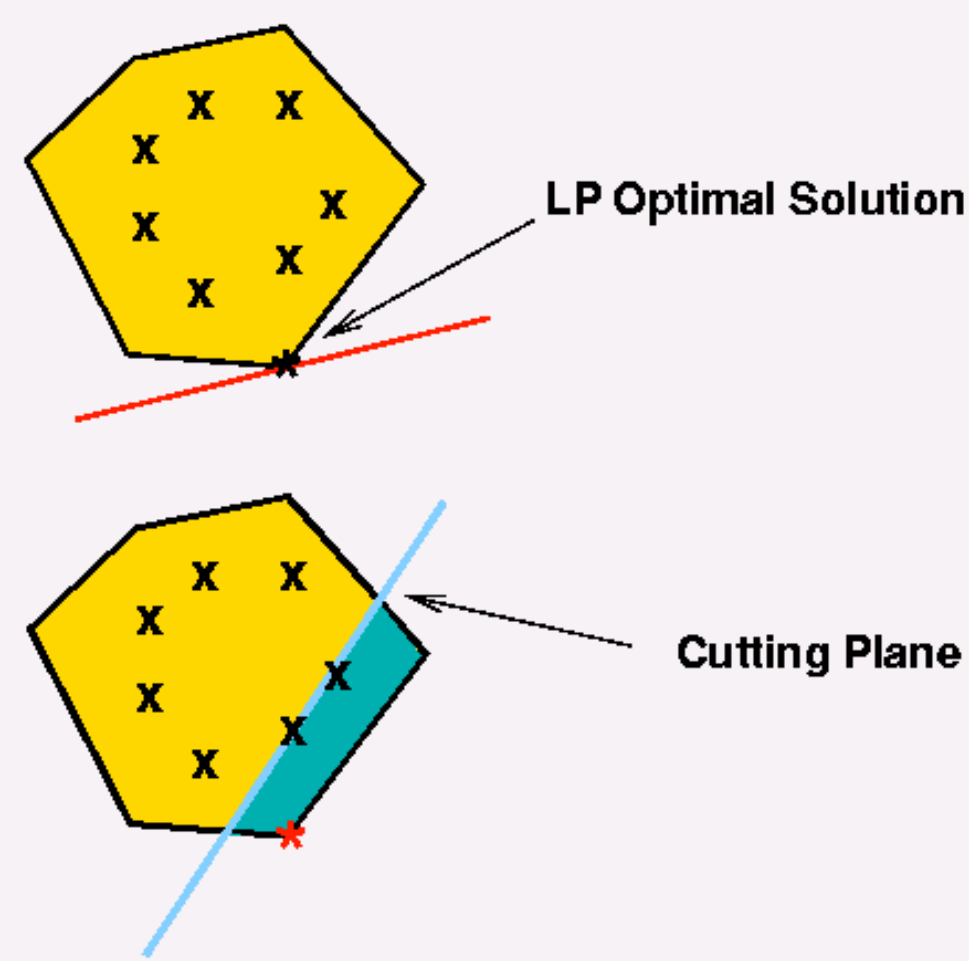
αντιμετωπιστεί αυτό το ζήτημα χρησιμοποιούμε την μεθοδολογία Cutting Planes των G. Dantzig, R. Fulkerson, και S. Johnson [11].

Υποθέτωντας ότι όλοι οι περιορισμοί του γραμμικού μας προβλήματος είναι της μορφής <= (γραμμικές ανισότητες), τότε η λύση του δίνεται, από την διασταύρωση ημιχώρων που προκύπτουν από τους περιορισμούς (σχήμα 4.11). Η ομάδα λύσεων S είναι μια επίπεδη επιφάνεια η οποία σχηματίζει ένα πολύγωνο όπως στο σχήμα που ακολουθεί.



Σχήμα 4.11

Χαρακτηριστικό της μεθόδου Simplex είναι ότι η βέλτιστη λύση x* (optimal solution) που προκύπτει βρίσκεται σε ένα από τα ακραία σημεία του πολυγώνου η οποία ορίζεται από έναν συγκεκριμένο γραμμικό περιορισμό. Είναι πιθανό όμως να υπάρχει ένας γραμμικός περιορισμός ο οποίος ικανοποιείται απ' όλα τα σημεία x που ανήκουν στην ομάδα λύσεων S αλλά όχι από την βέλτιστη λύση x* (σχήμα 4.12). Τότε αυτός ο γραμμικός περιορισμός καλείται Cutting Plane (cut), ο οποίος με την μορφή γραμμικής ανισότητας προστίθεται στο πρόβλημα μας, αποκλείωντας την προηγούμενη βέλτιστη λύση x*(εφόσον δεν ανήκει στην ομάδα λύσεων S) και εκ νέου μέσω της μεθόδου Simplex βρίσκεται νέα βέλτιστη λύση x* (η οποία πλέον ανήκει στην ομάδα λύσεων S) [13].



σχήμα 4.12

4.3 Πρόγραμμα F.O.R (Finding Optimal Route)

Σκοπός του προγράμματος που δημιουργήσαμε είναι η εύρεση της βέλτιστης διαδρομής στο πρόβλημα του πλανόδιου πωλητή (TSP), είτε αυτό είναι στατικό είτε είναι εως κάποιο βαθμό δυναμικό. Συγκεκριμένα πρόκειται για ένα διαδραστικό πρόγραμμα του οποίου ο κώδικας είναι γραμμένος στη γλώσσα προγραμματισμού PYTHON. Το πρόγραμμα F.O.R δίνει την δυνατότητα στον χρήστη, να επιλύσει ολες τις περιπτώσεις του προβλήματος του πλανώδιου πωλητή (TSP) απο την βιβλιοθήκη TSPlib. Επιπλέον μέσω συγκεκριμένων ερωτήσεων, δίνει την δυνατότητα στον χρήστη να αλλάξει την ροή του προγράμματος σύμφωνα με τις δικές του ανάγκες και να δημιουργήσει σε ένα βαθμό ένα νέο πρόβλημα πλανόδιου πωλητή, το οποίο διαμορφώνεται υπό τις συνθήκες που επιλέγει ο ίδιος.

4.3.1 Πρόγραμμα Main.py

Πρόκειται για το κυρίως πρόγραμμα το οποίο καλεί συγκεκριμένα υποπρογράμματα ανάλογα με τις ανάγκες του χρήστη. Ο χρήστης στην αρχή έχει την δυνατότητα να επιλέξει την συγκεκριμένη τοποθεσία/φάκελο που θα αποθηκευθεί εν τέλει το πρόβλημα που θα δημιουργήσει, καθώς και τα αποτελέσματα που θα προκύψουν.

Στην συνέχεια ο χρήστης υποβάλλεται στην ερώτηση αν θέλει να δημιουργήσει έναν νέο γράφο. Εάν όχι τότε μπορεί να επιλέξει έναν από τους προηγούμενους που έχει ήδη δημιουργήσει δίνοντας την τοποθεσία/φάκελο όπου είναι αποθηκευμένος ο συγκεκριμένος γράφος και να επιλέξει τον νέο φάκελο όπου θα αποθηκευθεί εκ νέου το συγκεκριμένο παράδειγμα το οποίο θα τρέξει.

Εάν ο χρήστης επιλέξει να δημιουργήσει νέο γράφο τότε υποβάλλεται στην επόμενη ερώτηση, η οποία είναι εάν θέλει να προσθέσει στο πρόβλημα του τυχαία δεδομένα κυκλοφοριακής συμφόρησης. Το επόμενο δεδομένο το οποίο καλείται να εισάγει ο χρήστης είναι ο αριθμός πόλεων που θέλει να έχει στο πρόβλημα του έτσι ώστε να δημιουργηθεί ο πίνακας κόστους της ευκλίδειας απόστασης. Επόμενο βήμα του χρήστη είναι να χαρακτηρίσει την συνδεσιμότητα των πόλεων επιλέγωντας ένα κριτήριο βάρους απο το 0 έως το 1 ([0.0 , 1.0]).

Ύστερα από αυτά τα βήματα το πρόβλημα του χρήστη έχει δημιουργηθεί, επιλύεται μέσω του αλγόριθμου **Concorde** και γίνεται σχηματική παρουσίαση όλων των βημάτων επίλυσης του συγκεκριμένου προβλήματος πλανόδιου πωλητή (TSP).

Ουσιαστικά το κυρίως πρόγραμμα **main.py** είναι υπεύθυνο για την δημιουργία φακέλων για κάθε ξεχωριστό παράδειγμα που θέλει να εκτελέσει ο χρήστης, την σωστή εκτέλεση όλων των υποπρογραμμάτων και το πέρασμα των σωστών ορισμάτων σε κάθε φάκελο. Τα 4 υποπρογράμματα τα οποία το κυρίως πρόγραμμα **main.py** χρησιμοποιεί είναι τα **createLocations.py, createTrafficData.py, prepareData.py** και **plotLocations.m.**

4.3.2 Υποπρόγραμμα createLocations.py

Το υποπρόγραμμα **createLocations.py** έχει ως σκοπό την δημιουργία δύο αρχείων, ένα για τις γεωμετρικές συντεταγμένες (x , y) και ένα για το κόστος της απόστασης σε km από τη μία στην άλλη. Αρχικά ρωτάμε τον χρήστη για τον αριθμό πόλεων (προορισμών) που θέλει να έχει ο γράφος καθώς και την πιθανότητα συνδεσιμότητας με κάθε πόλη (προορισμό) – π.χ αν ρίχνουμε ένα κέρμα κάθε φορά που σκεφτόμαστε αν μια πόλη συνδέεται με την άλλη τότε η πιθανότητα είναι 50% ή 0.5 (κριτήριο βάρους χρήστη).

Το πρώτο αρχείο ονομάζεται locations.dat όπου αποτελείται από μια αριθμημένη λίστα από το 0 έως n-1, όσες δηλαδή είναι και οι πόλεις μας και έχει την μορφή π.χ.

These are the locations of the cities
#city x y
0 12.2 0.61
1 23.94 19.47
2 15.84 19.97
3 12.71 19.79
4 22.46 17.36

Οι γεωγραφικές συντεταγμένες δίνονται απο την uniform random distribution συνάρτηση και παίρνουν τιμές απο 0 έως 30 km. Συνεπώς ο γράφος μπορεί να καλύψει μια επιφάνεια 900 τετραγωνικών χιλιομέτρων.

Το δεύτερο αρχείο ονομάζεται euclideanDistances.dat όπου αντιστοιχεί στον πίνακα συνδεσιμότητας τών πόλεων. Κάθε στοιχείο (i , j) του πίνακα αντιστοιχεί στο κόστος της ευκλείδιας απόστασης από την μία πόλη στην άλλη. Ο πίνακας αρχικοποιείται με τον αριθμό 100000 και με βάση την πιθανότητα/κριτήριο βάρους του χρήστη, δημιουργούμε τις συνδέσεις της κάθε πόλης. Κάθε φορά που υπάρχει συνδεσιμότητα υπολογίζουμε την ευκλείδια απόσταση sqrt((xj-xi)^2 + (yj-yi)^2) των συγκεκριμένων πόλεων. Παρατηρούμε ότι υπολογίζουμε την ευκλείδια απόσταση από την πόλη i προς την πόλη j, αλλά και την ευκλείδια απόσταση από την πόλη j προς την πόλη i. Προφανώς, η μεταξύ τους ευκλείδια απόσταση δεν μεταβάλλεται. Αλλά τα συγκεκριμένα δεδομένα θα μεταβληθούν λόγω των (τυχαίων) δεδομένων κυκλοφοριακής συμφόρησης.

Συνεπώς το ένα ρεύμα κατεύθυνσης θα έχει διαφορετικό κόστος από το άλλο. Δεδομένο που ανταποκρίνεται στην πραγματικότητα π.χ στις ώρες κυκλοφοριακής αιχμής σε αστικά κέντρα, όπου το ένα ρεύμα κυκλοφορίας είναι μπλοκαρισμένο ενώ το άλλο σχεδόν άδειο. Με αυτό τον τρόπο καταφέρνουμε να ενισχύσουμε και την δυναμικότητα που έχει το πρόγραμμά μας.
Ο πίνακας συνδεσιμότητας που προκύπτει έχει την μορφή :

This the cost matrix using only euclidean distance
10000.0 22.22 19.7 19.19 19.64
22.22 10000.0 8.12 11.23 2.58

19.7 8.12 10000.0 3.14 7.12
19.19 11.23 3.14 10000.0 10.05
19.64 2.58 7.12 10.05 10000.0

4.3.3 Υποπρόγραμμα createTrafficData.py

Το υποπρόγραμμα **createTrafficData.py** καλείται από το πρόγραμμα main.py με το όρισμα none, στην περίπτωση που ο χρήστης μας δεν θέλει να εισάγει στο πρόβλημα TSP δεδομένα κυκλοφοριακής συμφόρησης. Όμως αυτό που καλείται κάθε φορά να κάνει είναι να ανοίξει και να τροποποιήσει τον πίνακα του αρχείου euclideanDistances.dat, διαιρώντας όλες τις ευκλείδιες αποστάσεις με μια μέση ταχύτητα V, την οποία έχουμε θέσει εξ' αρχής στο πρόγραμμα μας ίση με V=40 km/h. Επίσης διαιρούμε όλες τις ευκλείδιες αποστάσεις με τον αριθμό 60 έτσι ώστε ο χρόνος που προκύπτει για κάθε στοιχείο του πίνακα να είναι σε λεπτά.

Αφού έχουμε μετατρέψει το κόστος των διαδρομών σε χρόνο τότε προσθέτουμε ένα τυχαίο χρόνο, ο οποίος δίνεται από μια συνάρτηση με gaussian κατανομή, με μέση τιμή το 40% του χρόνου που χρειαζόμαστε να διανύσουμε την συγκεκριμένη απόσταση (όπως υπολογίστηκε παραπάνω) και standard deviation 5. Όλα αυτά τα δεδομένα που έχουμε ορίσει μπορούν εύκολα να τροποποιηθούν μέσα στον κώδικα με σκοπό να μελετήσουμε στο μέλλον και άλλες συνθήκες. Τέλος, γράφουμε όλα τα δεδομένα, έτσι ώστε να είναι έτοιμα για την μετατροπή τους από το επόμενο υποπρόγραμμα, το **prepareData.py.**

Ακολουθεί τελικός πίνακας δεδομένων κυκλοφοριακής συμφόρησης, ο οποίος έχει υποστεί τις μετατροπές που προαναφέραμε.

These are the traffic data.
10000.0 33.33 29.55 28.79 29.46
33.33 10000.0 12.18 16.85 3.87
29.55 12.18 10000.0 4.71 10.68
28.79 16.85 4.71 10000.0 15.08
29.46 3.87 10.68 15.08 10000.0

4.3.4 Υποπρόγραμμα prepareData.py

Το υποπρόγραμμα prepareData.py είναι υπεύθυνο για την τροποποίηση του πίνακα δεδομένων κυκλοφοριακής συμφόρησης σε τελική μορφή αρχείου τύπου TSPLib. Στη

συνέχεια εισάγουμε το προηγούμενο αρχείο σαν αρχείο εισόδου στον αλγόριθμο Concorde. Το τελικό αρχείο που προκύπτει έχει την εξής μορφη :

παράδειγμα αρχείου της TSPlib (br17.atsp.gz)

```
NAME: br17
TYPE: ATSP
COMMENT: 17 city problem (Repetto)
DIMENSION: 17
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
EDGE_WEIGHT_SECTION
 9999   3   5  48  48   8   8   5   5   3   3   0   3   5   8   8
  5
  3 9999   3  48  48   8   8   5   5   0   0   3   0   3   8   8
  5
  5   3 9999  72  72  48  48  24  24   3   3   5   3   0  48  48
 24
 48  48  74 9999   0   6   6  12  12  48  48  48  48  74   6   6
 12
 48  48  74   0 9999   6   6  12  12  48  48  48  48  74   6   6
 12
  8   8  50   6   6 9999   0   8   8   8   8   8   8  50   0   0
  8
  8   8  50   6   6   0 9999   8   8   8   8   8   8  50   0   0
  8
  5   5  26  12  12   8   8 9999   0   5   5   5   5  26   8   8
  0
  5   5  26  12  12   8   8   0 9999   5   5   5   5  26   8   8
  0
  3   0   3  48  48   8   8   5   5 9999   0   3   0   3   8   8
  5
  3   0   3  48  48   8   8   5   5   0 9999   3   0   3   8   8
  5
```

```
  0   3   5  48  48   8   8   5   5   3   3 9999   3   5   8   8
  5
  3   0   3  48  48   8   8   5   5   0   0   3 9999   3   8   8
  5
  5   3   0  72  72  48  48  24  24   3   3   5   3 9999  48  48
 24
  8   8  50   6   6   0   0   8   8   8   8   8   8  50 9999   0
  8
  8   8  50   6   6   0   0   8   8   8   8   8   8  50   0 9999
  8
  5   5  26  12  12   8   8   0   0   5   5   5   5  26   8   8
 9999
EOF
```

4.3.5 Υποπρόγραμμα plotLocations.m

Το υποπρόγραμμα **plotLocations.m** καλείται από το πρόγραμμα main.py με στόχο να παρουσιάσει με φιλικό τρόπο στο χρήστη ένα γράφημα της λύσης που έχει προταθεί από τον Concorde ως βέλτιστη. Χρησιμοποιώντας ως αρχείο εισόδου το **locations.dat,** πρώτα παρουσιάζουμε με κύκλο τις τοποθεσίες των πόλεων πάνω στο δισδιάστατο χάρτη καθώς επίσης και τον αύξοντα αριθμό της πόλης. Ο αρχικός κόμβος και το πρώτο σκέλος της διαδρομής παρουσιάζονται με πράσινο χρώμα. Τα υπόλοιπα σκέλη εμφανίζονται σταδιακά με κόκκινο χρώμα, ενώ παράλληλα σε κάθε βήμα αποθηκεύεται ένα στιγμιότυπο του γράφου σε αρχείο φωτογραφίας τύπου image00xxxx.png. Ο χρήστης με αυτό το τρόπο μπορεί στο μέλλον να αναλύσει τα αποτελέσματα ή ακόμα και να δημιουργήσει ένα video χρησιμοποιώντας αυτά τα στιγμιότυπα. Αυτή η επιλογή είναι διαθέσιμη για χρήστες Linux ή Mac που έχουν προεγκατεστημένο το πρόγραμμα mencoder, αρκεί να τρέξουν το script ./convertToVideoLinux.sh ή ./convertToVideoMac.sh.

ΚΕΦΑΛΑΙΟ 5

5.1 Επίλυση Προβλήματος της TSPlib

Αρχικά θα επιλύσουμε ένα αρχείο της TSPlib με ονομασία **dj38** , το οποίο αφορά την εύρεση βέλτιστης διαδρομής σε ένα δίκτυο 38 πόλεων στο Djibouti. Δίνεται το αρχείο δεδομένων TSPlib (http://www.tsp.gatech.edu/world/dj38.tsp) :

NAME: dj38
COMMENT : 38 locations in Djibouti
COMMENT : Derived from National Imagery and Mapping Agency data
COMMENT : This file is a corrected version of dj89, where duplications
COMMENT: requesting data sets without duplications.
TYPE: TSP
DIMENSION: 38
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 11003.611100 42102.500000
2 11108.611100 42373.888900
3 11133.333300 42885.833300
4 11155.833300 42712.500000
5 11183.333300 42933.333300
6 11297.500000 42853.333300
7 11310.277800 42929.444400
8 11416.666700 42983.333300
9 11423.888900 43000.277800
10 11438.333300 42057.222200
11 11461.111100 43252.777800
12 11485.555600 43187.222200
13 11503.055600 42855.277800
14 11511.388900 42106.388900
15 11522.222200 42841.944400
16 11569.444400 43136.666700
17 11583.333300 43150.000000
18 11595.000000 43148.055600

19 11600.000000 43150.000000
20 11690.555600 42686.666700
21 11715.833300 41836.111100
22 11751.111100 42814.444400
23 11770.277800 42651.944400
24 11785.277800 42884.444400
25 11822.777800 42673.611100
26 11846.944400 42660.555600
27 11963.055600 43290.555600
28 11973.055600 43026.111100
29 12058.333300 42195.555600
30 12149.444400 42477.500000
31 12286.944400 43355.555600
32 12300.000000 42433.333300
33 12355.833300 43156.388900
34 12363.333300 43189.166700
35 12372.777800 42711.388900
36 12386.666700 43334.722200
37 12421.666700 42895.555600
38 12645.000000 42973.333300

Τα αποτελέσματα που προέκυψαν μέσω της επίλυσης απο τον αλγόριθμο Concorde είναι τα εξής :

Running concorde on the Data/dj38.tsp file
Host: unknown-00-26-bb-19-22-8e-2.home Current process id: 11282
Using random seed 1362828444
Problem Name: dj38
38 locations in Djibouti
Derived from National Imagery and Mapping Agency data
Problem Type: TSP
Number of Nodes: 38
Rounded Euclidean Norm (CC_EUCLIDEAN)
Set initial upperbound to 6656 (from tour)
  LP Value 1: 6357.000000 (0.00 seconds)

LP Value 2: 6656.000000 (0.00 seconds)

New lower bound: 6656.000000

Final lower bound 6656.000000, upper bound 6656.000000

Exact lower bound: 6656.000000

DIFF: 0.000000

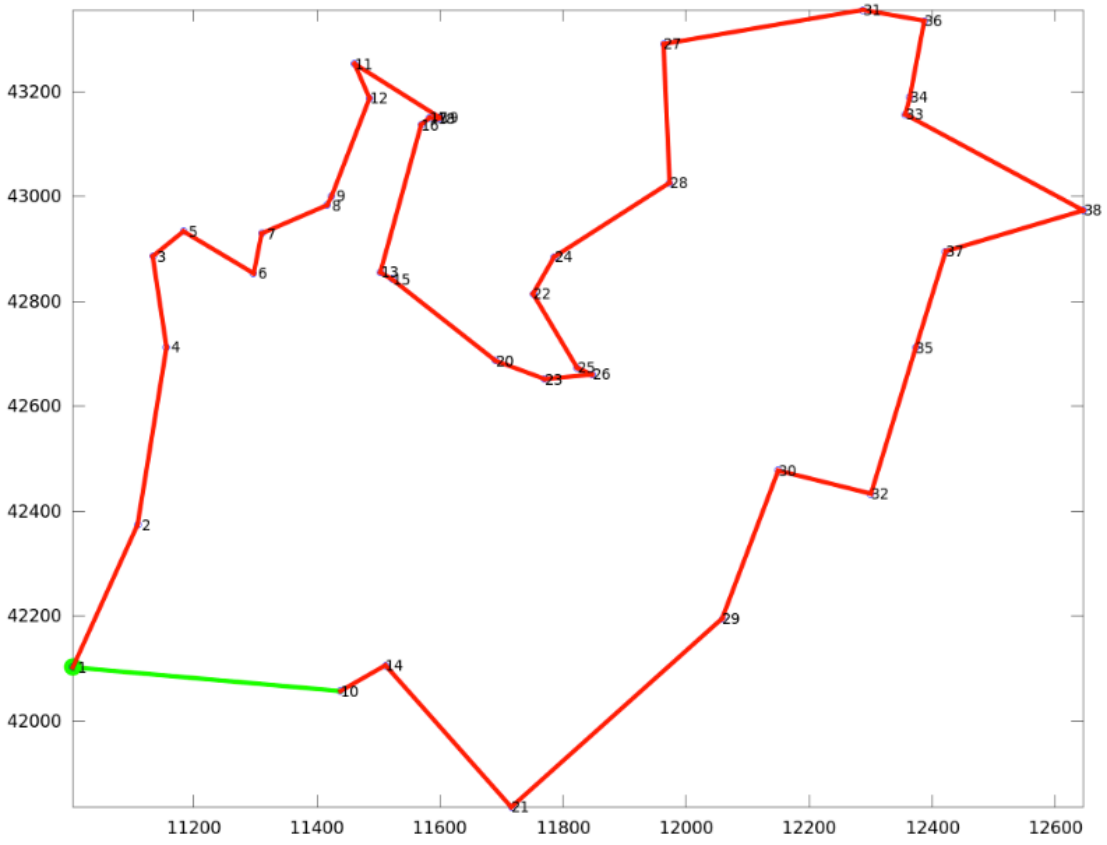
Final LP has 43 rows, 172 columns, 427 nonzeros

Optimal Solution: 6656.00

Number of bbnodes: 1

Total Running Time: 0.04 (seconds).

Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP σε 0.04 δευτερόλεπτα. Το κόστος της βέλτιστης διαδρομής είναι 6656 (δηλαδή το μήκος της). Ακολουθεί η σχηματική επίλυση του προβλήματος dj38 της TSPlib (σχήμα 5.1), όπου το πράσινο σημείο είναι το σημείο εκκίνησης και η πράσινη διαδρομή δηλώνει ότι είναι η πρώτη που ακολουθείται στο δρομολόγιο μας :



Σχήμα 5.1

5.2 Δημιουργία και Επίλυση στατικού προβλήματος TSP απο χρήστη

Το συγκεκριμένο πρόβλημα πλανόδιου πωλητή TSP, δημιουργήθηκε και διαμορφώθηκε από τον χρήστη, επιλέγωντας τα δεδομένα που εισήγαγε στο πρόγραμμα/σύστημα **F.O.R**.

Αρχικά επιλέχθηκε ο προορισμός αποθήκευσης του αρχείου με ονομασία tp200. Στο επόμενο βήμα ο χρήστης αποφασίζει την δημιουργία ενός νέου γράφου. Έπειτα ο χρήστης αρνήθηκε να εισάγει δεδομένα κυκλοφοριακής συμφόρησης, εφόσον αποφασίζει να επιλύσει ένα στατικό πρόβλημα TSP. Η επόμενη απόφαση του είναι ο αριθμός των πόλεων από τον οποίο αποτελείται ο νέος γράφος και η επιλογή του κριτήριου βάρους συνδεσιμότητας μεταξύ των πόλεων. Αποφασίζει ότι ο αριθμός των πόλεων στο σύστημα μας θα είναι 200 και το κριτήριο βάρους συνδεσιμότητας παίρνει την τιμή 0.4.

Τα αποτελέσματα που προέκυψαν μέσω της επίλυσης απο τον αλγόριθμο Concorde είναι τα εξής :

```
concordeMac/concorde -f Data/tp200/final.tsp
Host: unknown-00-26-bb-19-22-8e-2.home  Current process id: 16996
Using random seed 1362844204
Problem Name: Data/tp200/final
Problem Type: TSP
myCity
Number of Nodes: 200
Explicit Lengths (CC_MATRIXNORM)
Set initial upperbound to 609 (from tour)
  LP Value  1: 598.500000  (0.12 seconds)
  LP Value  2: 606.500000  (0.25 seconds)
  LP Value  3: 608.750000  (0.34 seconds)
  LP Value  4: 608.750000  (0.37 seconds)
  LP Value  5: 609.000000  (0.45 seconds)
  LP Value  6: 609.000000  (0.47 seconds)
New lower bound: 609.000000
Final lower bound 609.000000, upper bound 609.000000
Exact lower bound: 609.000000
DIFF: 0.000000
```

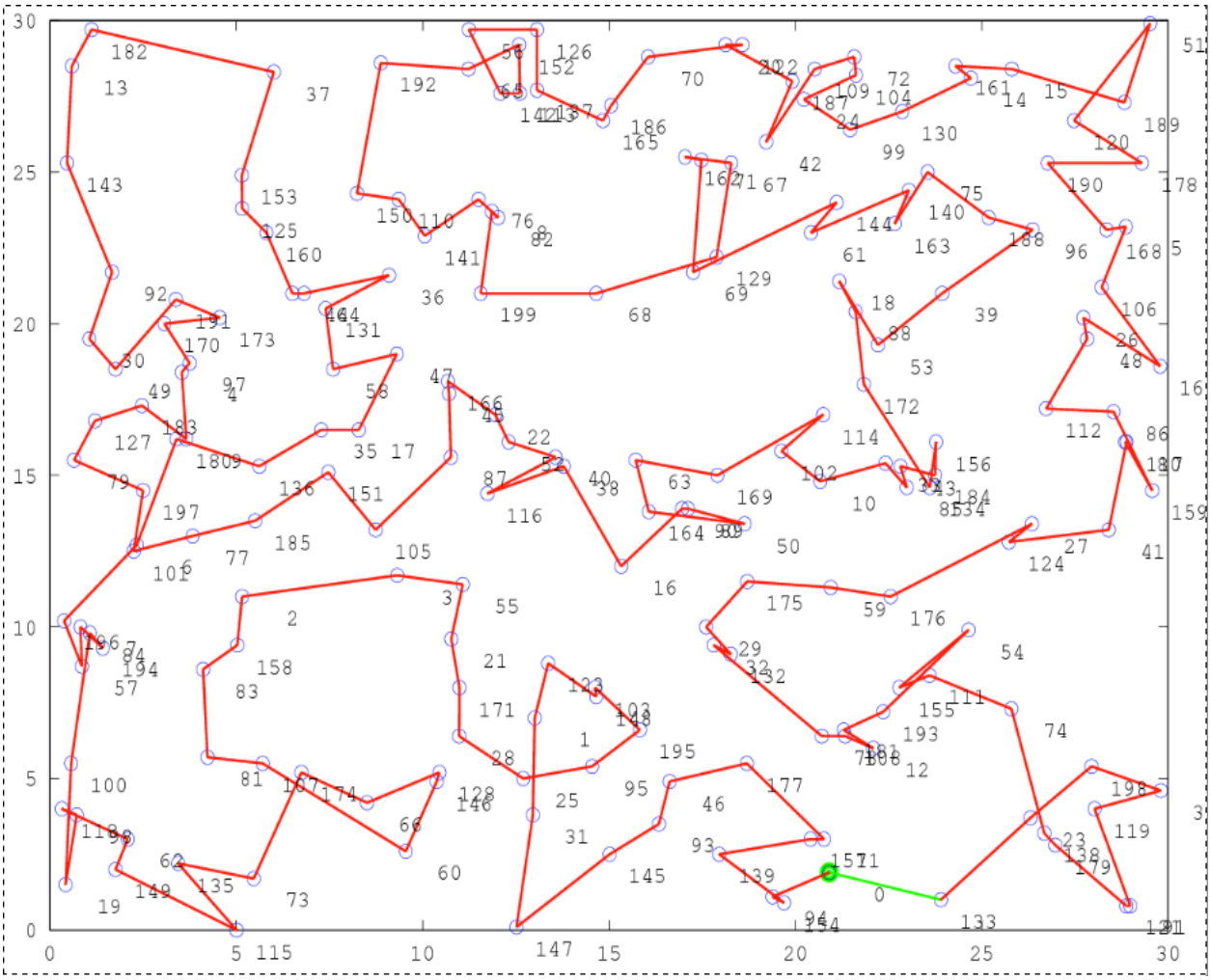
Final LP has 271 rows, 435 columns, 1591 nonzeros

Optimal Solution: 609.00

Number of bbnodes: 1

Total Running Time: 0.71 (seconds)

Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP με ονομασία tp200 σε 0.71 δευτερόλεπτα. Το κόστος της βέλτιστης διαδρομής είναι 609 (σε χρόνο/λεπτά/min).
Ακολουθεί η σχηματική επίλυση του προβλήματος tp200 (σχήμα 5.2), όπου το πράσινο σημείο είναι το σημείο εκκίνησης 0 και η πράσινη διαδρομή δηλώνει οτι είναι η πρώτη που ακολουθείται στο δρομολόγιο μας :



Σχήμα 5.2

5.3 Μετατροπή στατικού προβλήματος TSP σε δυναμικό πρόβλημα TSP

5.3.1 Δημιουργία και Επίλυση του στατικού προβλήματος TSP απο χρήστη

Το συγκεκριμένο πρόβλημα πλανόδιου πωλητή TSP, δημιουργήθηκε και διαμορφώθηκε από τον χρήστη, επιλέγωντας τα δεδομένα που εισήγαγε στο πρόγραμμα/σύστημα **F.O.R**.

Αρχικά επιλέχθηκε ο προορισμός αποθήκευσης του αρχείου με ονομασία tp50. Στο επόμενο βήμα ο χρήστης αποφασίζει την δημιουργία ενός νέου γράφου. Έπειτα ο χρήστης αρνήθηκε να εισάγει δεδομένα κυκλοφοριακής συμφόρησης, εφόσον αποφασίζει να επιλύσει ένα στατικό πρόβλημα TSP. Η επόμενη απόφαση του είναι ο αριθμός των πόλεων από τον οποίο αποτελείται ο νέος γράφος και η επιλογή του κριτήριου βάρους συνδεσιμότητας μεταξύ των πόλεων. Αποφασίζει οτι ο αριθμός των πόλεων στο σύστημά μας θα είναι 50 και το κριτήριο βάρους συνδεσιμότητας παίρνει την τιμή 0.8.

Τα αποτελέσματα που προέκυψαν μέσω της επίλυσης από τον αλγόριθμο Concorde είναι τα εξής :

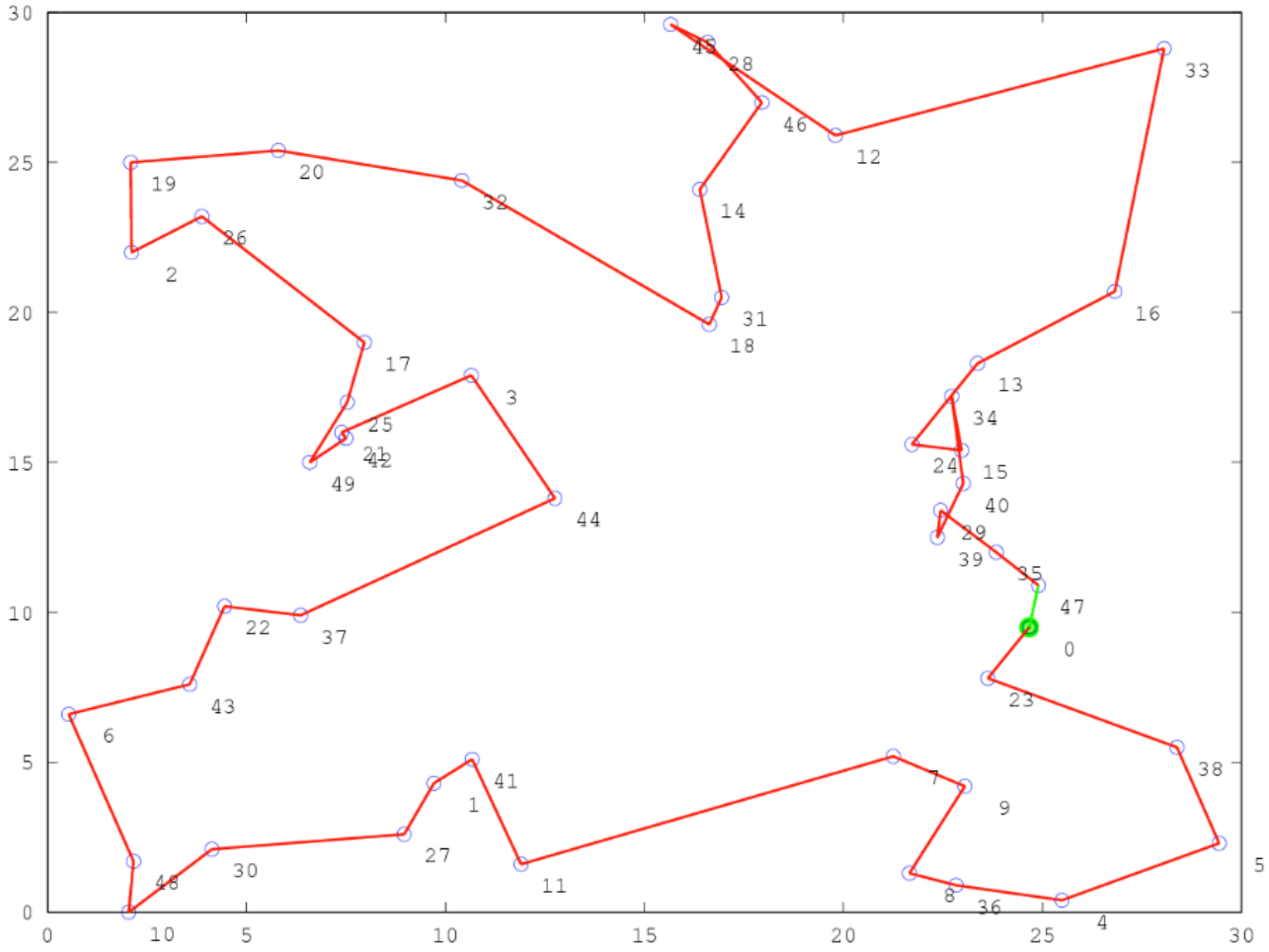
```
concordeMac/concorde -f Data/tp50/final.tsp
Host: unknown-00-26-bb-19-22-8e-2.home  Current process id: 21236
Using random seed 1362849907
Problem Name: Data/tp50/final
Problem Type: TSP
myCity
Number of Nodes: 50
Explicit Lengths (CC_MATRIXNORM)
Set initial upperbound to 248 (from tour)
  LP Value  1: 226.500000  (0.00 seconds)
  LP Value  2: 241.500000  (0.02 seconds)
  LP Value  3: 247.500000  (0.02 seconds)
  LP Value  4: 248.000000  (0.02 seconds)
New lower bound: 248.000000
Final lower bound 248.000000, upper bound 248.000000
Exact lower bound: 248.000000
DIFF: 0.000000
Final LP has 70 rows, 183 columns, 620 nonzeros
```

Optimal Solution: 248.00

Number of bbnodes: 1

Total Running Time: 0.05 (seconds)

Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP με ονομασία tp50 σε 0.05 δευτερόλεπτα. Το κόστος της βέλτιστης διαδρομής είναι 248 (σε χρόνο/λεπτά/min).
Ακολουθεί η σχηματική επίλυση του προβλήματος tp50 (σχήμα 5.3), όπου το πράσινο σημείο είναι το σημείο εκκίνησης 0 και η πράσινη διαδρομή δηλώνει ότι είναι η πρώτη που ακολουθείται στο δρομολόγιο μας :



σχήμα 5.3

5.3.2 Δημιουργία και Επίλυση του δυναμικού προβλήματος tp50_1 χρησιμοποιώντας δεδομένα κυκλοφοριακής συμφόρησης

Το συγκεκριμένο στατικό πρόβλημα πλανόδιου πωλητή TSP με ονομασία tp50, δημιουργήθηκε και διαμορφώθηκε από τον χρήστη, επιλέγωντας τα δεδομένα που εισήγαγε στο πρόγραμμα/σύστημα **F.O.R**.

Θα χρησιμοποιήσουμε το γράφο του στατικού προβλήματος tp50 για να μπορέσει να υπάρξει σύγκριση ως προς το νέο δυναμικό πρόβλημα. Ο χρήστης αυτή την φορά αποφασίζει να εισάγει στο σύστημα δεδομένα κυκλοφοριακής συμφόρησης, εφόσον αποφασίζει να μετατρέψει το στατικό πρόβλημα TSP tp50 στο δυναμικό πρόβλημα TSP με ονομασία tp50_1.

Τα αποτελέσματα που προέκυψαν μέσω της επίλυσης απο τον αλγόριθμο Concorde είναι τα εξής :

```
concordeMac/concorde -f Data/tp50_1/final.tsp
Host: unknown-00-26-bb-19-22-8e-2.home  Current process id: 22029
Using random seed 1362850749
Problem Name: Data/tp50_1/final
Problem Type: TSP
myCity
Number of Nodes: 50
Explicit Lengths (CC_MATRIXNORM)
Set initial upperbound to 260 (from tour)
Error in dual solution - 2
Chkmat found error in matching
Fractional matching routine failed
Warning: restarting running timer Miscellaneous
No warmstart, stumbling on anyway
  LP Value  1: 244.000000  (0.00 seconds)
  LP Value  2: 255.500000  (0.00 seconds)
  LP Value  3: 260.000000  (0.01 seconds)
New lower bound: 260.000000
Final lower bound 260.000000, upper bound 260.000000
Exact lower bound: 260.000000
```

DIFF: 0.000000

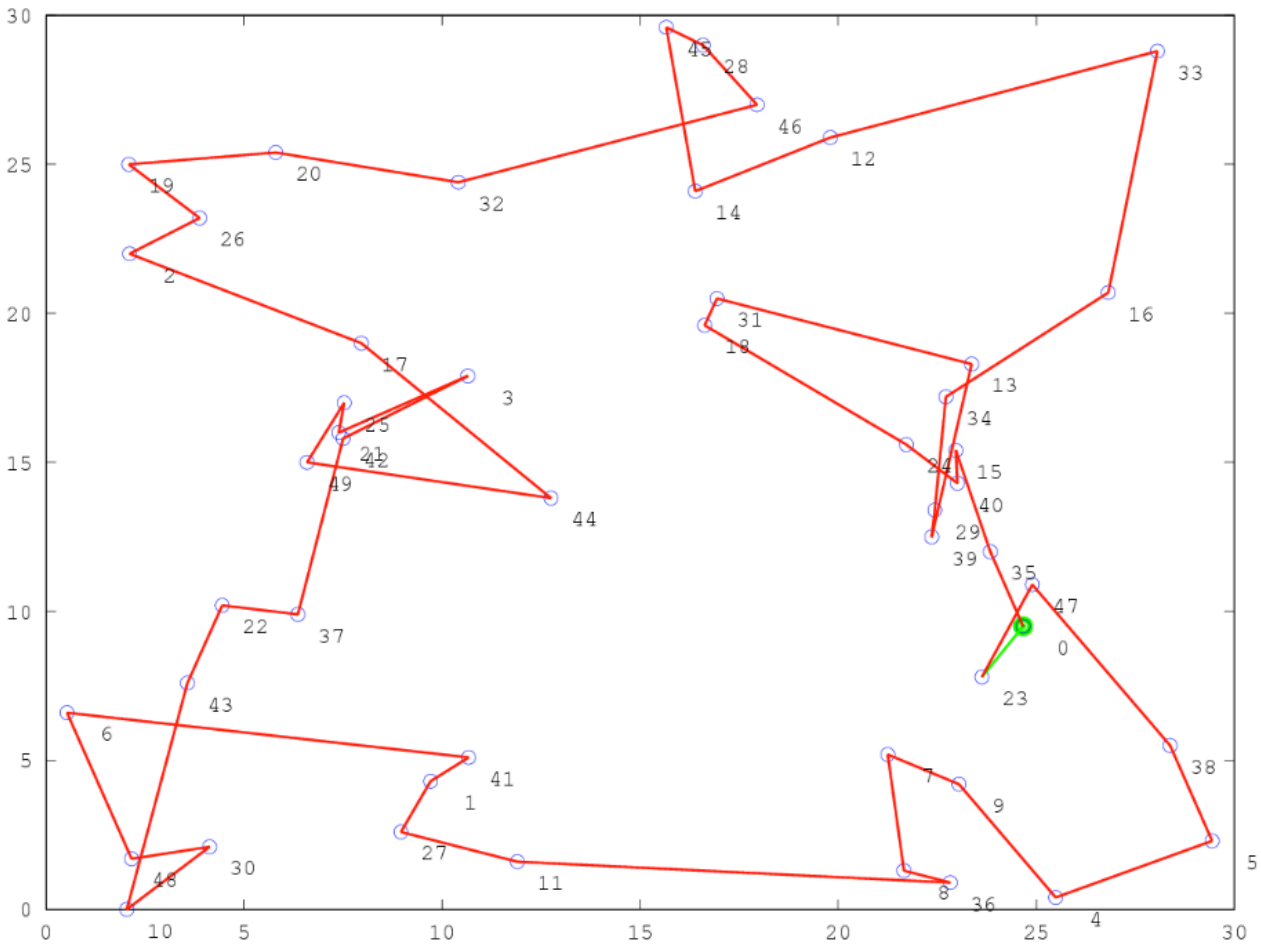
Final LP has 61 rows, 105 columns, 379 nonzeros

Optimal Solution: 260.00

Number of bbnodes: 1

Total Running Time: 0.04 (seconds)

Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP με ονομασία tp50 σε 0.04 δευτερόλεπτα. Το κόστος της βέλτιστης διαδρομής είναι 260 (σε χρόνο/λεπτά/min). Ακολουθεί η σχηματική επίλυση του προβλήματος tp50_1 (σχήμα 5.4) όπου, το πράσινο σημείο είναι το σημείο εκκίνησης 0 και η πράσινη διαδρομή δηλώνει ότι είναι η πρώτη που ακολουθείται στο δρομολόγιο μας :



Σχήμα 5.4

5.3.3 Επίλυση του δυναμικού προβλήματος tsp50_2 με νέα δεδομένα κυκλοφοριακής συμφόρησης.

Θα χρησιμοποιήσουμε το γράφο του δυναμικού προβλήματος tp50_1 για να μπορέσει να υπάρξει σύγκριση ως προς το νέο δυναμικό πρόβλημα tp50_2. Ο χρήστης αυτή την φορά αποφασίζει να εισάγει στο σύστημα νέα δεδομένα κυκλοφοριακής συμφόρησης τα οποία είναι διαφορετικά από τα δεδομένα κυκλοφοριακής συμφόρησης του δυναμικού προβλήματος tp50_1. Αυτό συμβαίνει λόγω της τυχαίας τροποποίησης του πίνακα κόστους-συνδεσιμότητας τών πόλεων.

Τα αποτελέσματα που προέκυψαν μέσω της επίλυσης από τον αλγόριθμο Concorde είναι τα εξής :

concordeMac/concorde -f Data/tp50_2/final.tsp
Host: unknown-00-26-bb-19-22-8e-2.home Current process id: 22737
Using random seed 1362851259
Problem Name: Data/tp50_2/final
Problem Type: TSP
myCity
Number of Nodes: 50
Explicit Lengths (CC_MATRIXNORM)
Set initial upperbound to 273 (from tour)
Error in dual solution - 2
Chkmat found error in matching
Fractional matching routine failed
Warning: restarting running timer Miscellaneous
No warmstart, stumbling on anyway
  LP Value 1: 244.000000 (0.00 seconds)
  LP Value 2: 262.000000 (0.01 seconds)
  LP Value 3: 272.500000 (0.01 seconds)
  LP Value 4: 273.000000 (0.02 seconds)
New lower bound: 273.000000
Final lower bound 273.000000, upper bound 273.000000
Exact lower bound: 273.000000
DIFF: 0.000000

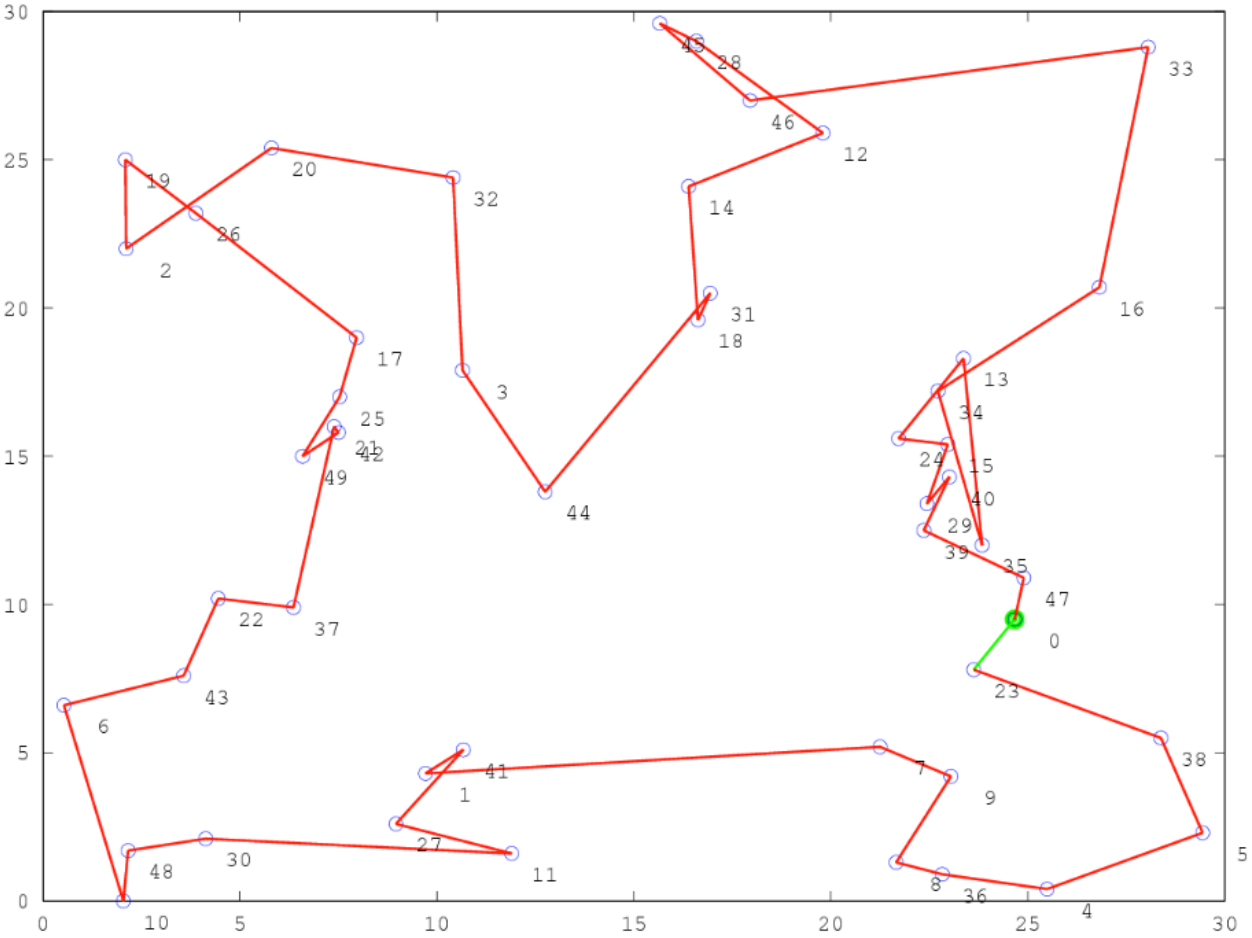
Final LP has 71 rows, 148 columns, 557 nonzeros

Optimal Solution: 273.00

Number of bbnodes: 1

Total Running Time: 0.05 (seconds)

Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP με ονομασία tp50_2 σε 0.05 δευτερόλεπτα. Το κόστος της βέλτιστης διαδρομής είναι 273 (σε χρόνο/λεπτά/min). Ακολουθεί η σχηματική επίλυση του προβλήματος tp50_2 (σχήμα 5.5) όπου, το πράσινο σημείο είναι το σημείο εκκίνησης 0 και η πράσινη διαδρομή δηλώνει ότι είναι η πρώτη που ακολουθείται στο δρομολόγιο μας :



Σχήμα 5.5

ΚΕΦΑΛΑΙΟ 6

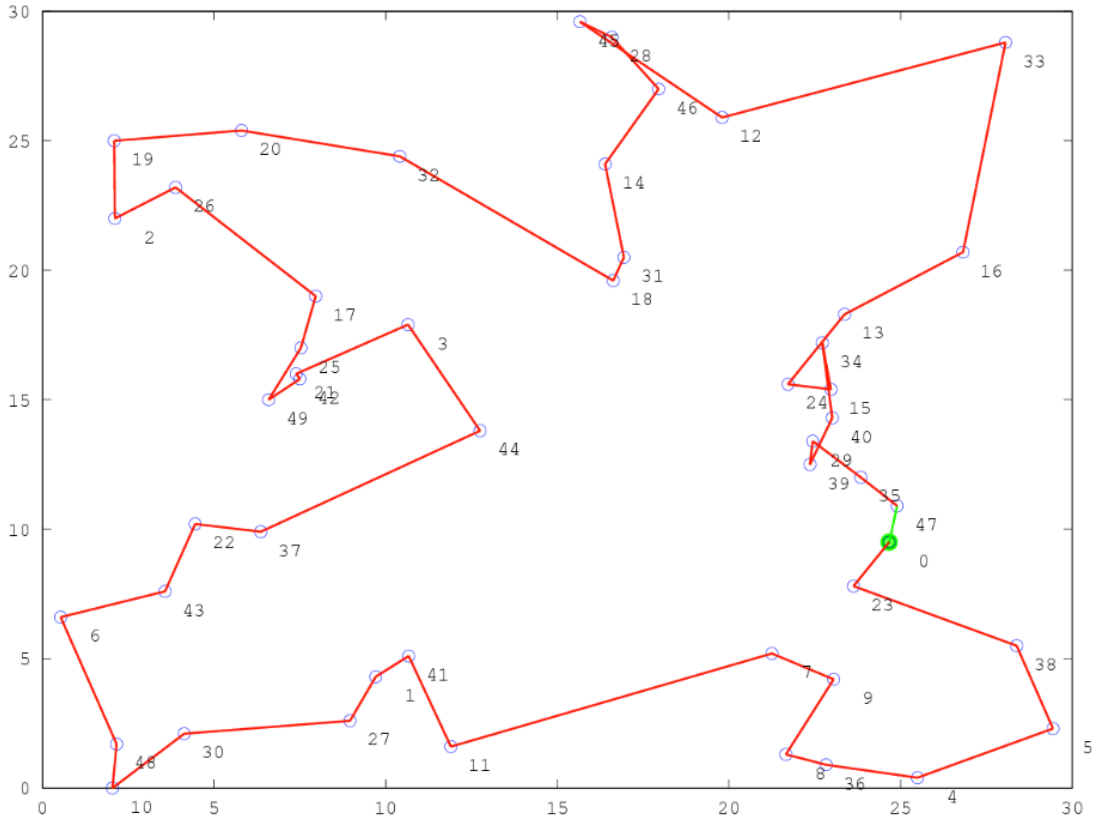
6.1 Σύγκριση αποτελεσμάτων μεταξύ στατικού και δυναμικού προβλήματος TSP

Μέσω του προγράμματος **F.O.R** δημιουργήσαμε το στατικό πρόβλημα TSP με ονομασία tp50, το οποίο αποτελείται απο 50 πόλεις/προορισμούς. Η επίλυση του από τον αλγόριθμο Concorde μας έδωσε το τελικό κατώτερο όριο το οποίο είναι ίσο με 248 καθώς και το κόστος της βέλτιστης διαδρομής, ίσο με 248 (σε χρόνο/λεπτα/min).

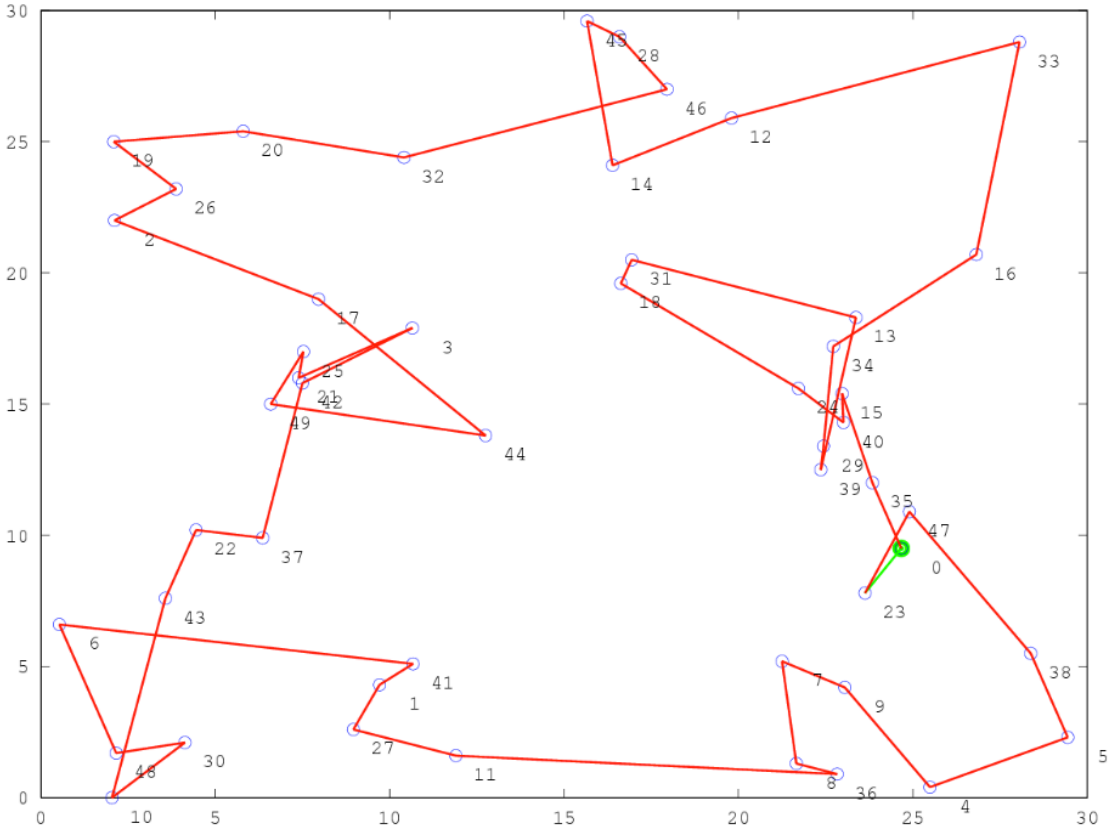
Έπειτα μετατρέψαμε το ίδιο στατικό πρόβλημε με ονομασία tp50 στο δυναμικό πρόβλημα tp50_1, διατηρώντας τον ίδιο γράφο στο σύστημα μας, αλλα προσθέσαμε δεδομένα κυκλοφοριακής συμφόρησης. Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP με ονομασία tp50_1 δίνωντας μας το τελικό κατώτερο όριο ίσο με 260 και το κόστος της βέλτιστης διαδρομής ίσο με 260 (σε χρόνο/λεπτά/min).

Παρατηρούμε ότι ενώ διατηρήσαμε τον ίδιο γράφο δηλαδή τους ίδιους προορισμούς/πόλεις, τα αποτελέσματα που προέκυψαν είναι διαφορετικά. Η διαφορά των αποτελεσμάτων είναι λογική και αναμενώμενη αφού τα κόστη των διαδρομών από την πόλη i στην πόλη j τροποποιήθηκαν. Τα κόστη των διαδρομών στο στατικό πρόβλημα tp50 εκφράζουν την ευκλείδια απόσταση μεταξύ των πόλεων. Τα κόστη όμως των διαδρομών του δυναμικού προβλήματος tp50_1 εμπεριέχουν και την κυκλοφοριακή συμφόρηση στα οδικό δίκτυο. Συνεπώς προέκυψε ότι το κόστος της βέλτιστης διαδρομής στο δυναμικό προβλημα tp50_1 είναι υψηλότερο από το κόστος της βέλτιστης διαδρομής στο στατικό προβλημα tp50. Η διαφορά στο κόστος της βέλτιστης διαδρομής μεταξύ των δύο προβλημάτων TSP, προκύπτει επίσης από το γεγονός ότι η βέλτιστη διαδρομή που ακολουθείται στο στατικό πρόβλημα tp50, είναι διαφορετική από αυτή που ακολουθείται στο δυναμικό πρόβλημα tp50_1.

Ακολουθεί η γραφική επίλυση των δύο προβλημάτων (σχήμα 6.1/σχήμα 6.2), όπου παρατηρούμε τις διαφορετικές βέλτιστες διαδρομές που επιλέχθηκαν:



Σχήμα 6.1: γραφική επίλυση του tp50



Σχήμα 6.2: γραφική επίλυση του tp50_1

6.2 Σύγκριση αποτελεσμάτων δυναμικού προβλήματος TSP με διαφορετικά δεδομένα κυκλοφοριακής συμφόρησης

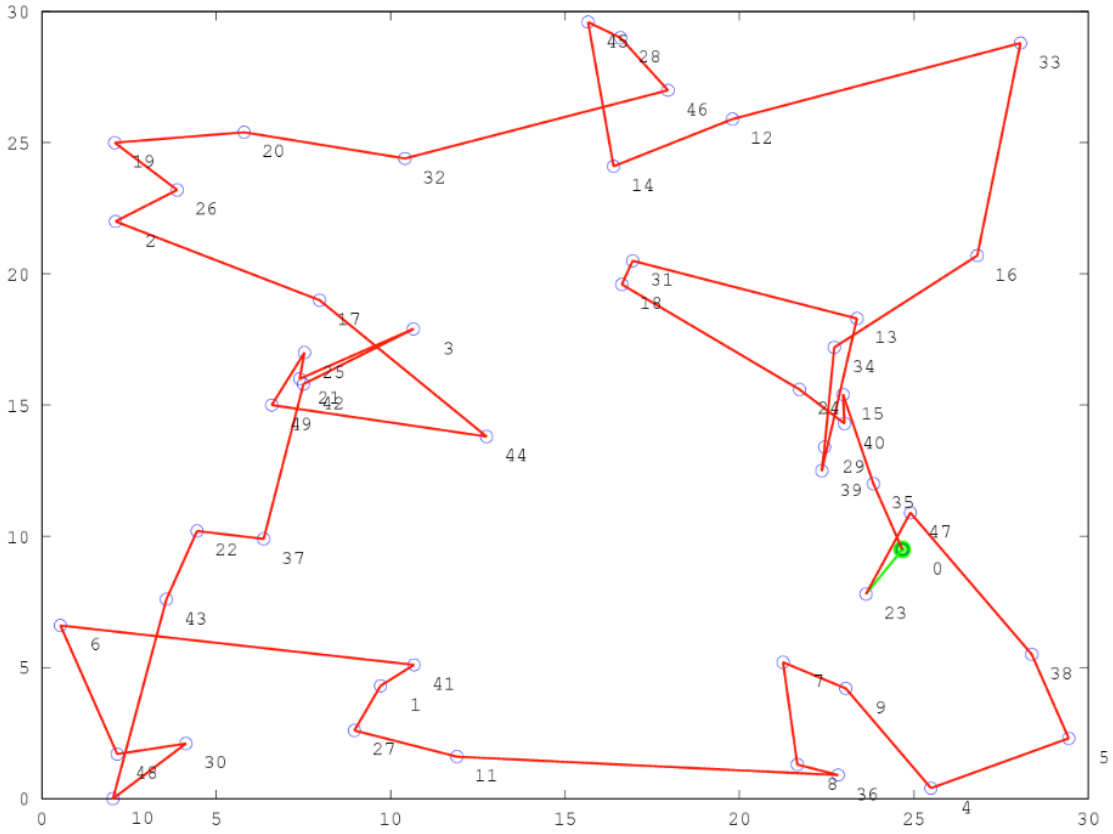
Μέσω του προγράμματος **F.O.R** δημιουργήσαμε το δυναμικό πρόβλημα TSP με ονομασία tp50_1 το οποίο αποτελείται απο 50 πόλεις/προορισμούς, διατηρώντας τον γράφο του στατικού προβλήματος TSP με ονομασία tp50. Η επίλυση του από τον αλγόριθμο Concorde μας έδωσε το τελικό κατώτερο όριο το οποίο είναι ίσο με 260 καθώς και το κόστος της βέλτιστης διαδρομής, ίσο με 260 (σε χρόνο/λεπτα/min).

Έπειτα δημιουργήσαμε το νέο δυναμικό πρόβλημα TSP με ονομασία tp50_2, διατηρώντας τον ίδιο γράφο στο σύστημα μας, εισάγωντας στο σύστημα μας νέα δεδομένα κυκλοφοριακής συμφόρησης. Ο αλγόριθμος Concorde επίλυσε το συγκεκριμένο πρόβλημα TSP δίνωντας μας το τελικό κατώτερο όριο ίσο με 273 και το κόστος της βέλτιστης διαδρομής ίσο με 273 (σε χρόνο/λεπτά/min).

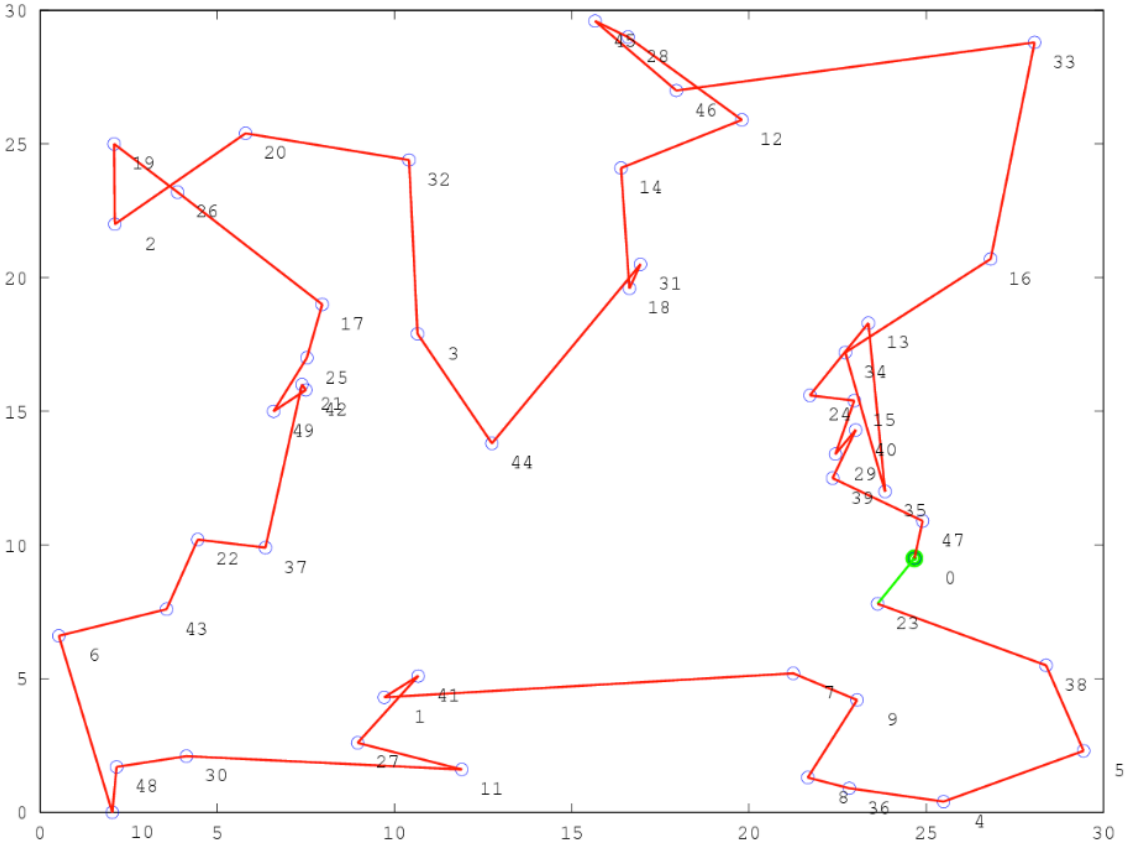
Παρατηρούμε ότι ενώ διατηρήσαμε τον ίδιο γράφο δηλαδή τους ίδιους προορισμούς/πόλεις, τα αποτελέσματα που προέκυψαν είναι διαφορετικά. Η διαφορά στα αποτελέσματα μεταξύ των δυναμικών προβλημάτων tp50_1 και tp50_2 οφείλεται στην διαφορετικότητα των δεδομένων κυκλοφοριακής συμφόρησης που εισήχθησαν στο κάθε σύστημα. Τα δεδομένα κυκλοφοριακής συμφόρησης που εισάγονται κάθε φορά στο σύστημα μας είναι τυχαία και προκύπτουν απο συγκεκριμένη συνάρτηση που έχουμε ορίσει στον κώδικα του προγράμματος **F.O.R**.

Η διαφορά στο κόστος της βέλτιστης διαδρομής μεταξύ των δύο δυναμικών προβλημάτων TSP, προκύπτει επίσης από το γεγονός ότι η βέλτιστη διαδρομή που ακολουθείται στο δυναμικό πρόβλημα tp50_1, είναι διαφορετική από τη βέλτιστη διαδρομή που ακολουθείται στο δυναμικό πρόβλημα tp50_1.

Ακολουθεί η γραφική επίλυση των δύο προβλημάτων (σχήμα 6.3/σχήμα 6.4), όπου παρατηρούμε τις διαφορετικές βέλτιστες διαδρομές που επιλέχθηκαν:



σχήμα 6.3: γραφική επίλυση του tp50_1



σχήμα 6.4: γραφική επίλυση του tp50_2

ΚΕΦΑΛΑΙΟ 7

7.1 Συμπέραμα

Το πρόβλημα του πλανόδιου πωλητή TSP είναι ένα από τα πιο δημοφιλή μαθηματικά προβλήματα καθώς δεν έχει βρεθεί ακόμα λύση για τη γενική περίπτωση του προβλήματος. Ανήκει στην κατηγορία προβλημάτων μη πολυωνυμικού χρόνου (NP-hard "non-deterministic polynomial"), με αποτέλεσμα ο χρόνος επίλυσης του αλγορίθμου να αυξάνεται εκθετικά σε σχέση με τον αριθμό πόλεων. Παρ' όλα αυτά ο αλγόριθμος Concorde βρίσκει την βέλτιστη λύση προσεγγιστικά και σε εύλογο χρονικό διάστημα. Να σημειωθεί ότι ο αλγόριθμος Concorde έχει δώσει το καλύτερο κατώτερο όριο, έως και σήμερα, στο παγκόσμιο πρόβλημα TSP το οποίο αποτελείται απο 1,904,711 πόλεις.

Η ιδιαιτερότητα όμως του πρόβληματος του πλανόδιου πωλητή TSP, έγκειται στο γεγονός ότι ο αριθμός πόλεων είναι ένα γνωστό αμετάβλητο δεδομένο του προβλήματος. Δηλαδή εισάγουμε τον αριθμό πόλεων στο σύστημα μας απο την αρχή και καθ' όλη την διάρκεια επίλυσης του προβλήματος του πλανόδιου πωλητή TSP έως και την εύρεση της βέλτιστης διαδρομής αυτός ο αριθμός παραμένει σταθερός. Αυτή η ιδιαιτερότητα είναι ένα σημαντικό εμπόδιο στην προσπάθεια ένταξης του πρόβληματος του πλανόδιου πωλητή TSP σε ένα δυναμικό περιβάλλον.

Συνεπώς η εύρεση βέλτιστης διαδρομής στο πρόβλημα του πλανόδιου πωλητή TSP μέσα σε ένα δυναμικό περιβάλλον κάποιες φορές καθίσταται ανέφικτη. Αναφέρουμε το χαρακτηριστικό παράδειγμα της φυσικής καταστροφής όπου ο μοναδικός δρόμος που συνδέει την πόλη i με το υπόλοιπο σύστημα κλείνει, αποκλείωντας την έτσι από τις υπόλοιπες. Άλλο χαρακτηριστικό παράδειγμα είναι η εμφάνιση ζήτησης σε μια πόλη, όπου δεν υπήρχε από την αρχή ανάμεσα στις πόλεις που εισάγαμε στο πρόβλημα μας, καθιστώντας έτσι το δρομολόγιο προς την νέα μας πόλη ανέφικτο. Η τροποποίηση στον αριθμό πόλεων έχει σαν συνέπεια την γέννεση νέων στατικών προβλημάτων και την σύγκριση νέων βέλτιστων διαδρομών αυξάνοντας έτσι την πολυπλοκότητα του προβλήματος.

Η προσθαφαίρεση πόλεων στο πρόβλημα του πλανόδιου πωλητή TSP είναι ένας πολύ σημαντικός περιορισμός στην εύρεση βέλτιστης διαδρομής σε δυναμικό περιβάλλον. Αυτός ο περιορισμός αποτελεί και μια από τις προκλήσεις που αντιμετωπίσαμε στην συγκεκριμένη διπλωματική εργασία (όπως και πολλοί ερευνητές στην προσπάθεια επίλυσης του

συγκεκριμένου συνδιαστικού προβλήματος βελτιστοποίησης), καθιστώντας μέχρι στιγμής την εύρεση βέλτιστης διαδρομής σε ένα ρεαλιστικό περιβάλλον, μη ρεαλιστική.

7.2 Μελλοντική Εργασία

Συγκεκριμένα όσον αφορά την ανάπτυξη του προγράμματος F.O.R, υπάρχουν κάποια στοιχεία τα οποία μπορούν να εισαχθούν και να βελτιώσουν την δυναμικότητα του προγράμματος όπως :

- Η δυνατότητα προσθαφαίρεσης πόλεων/προορισμών στο σύστημα μας από τον ίδιο τον χρήστη, σε πραγματικό χρόνο.
- Η επιλογή αρχικού αλλά και τελικού προορισμού/πόλης, δηλαδή η εισαγωγή συγκεκριμένων κανόνων στο πρόβλημα μας από τον χρήστη, σε πραγματικό χρόνο.
- Η δυνατότητα σύνδεσης του προγράμματος F.O.R με το διαδίκτυο, με σκοπό την αξιοποίηση την εισαγωγή και την ανανέωση των δεδομένων κυκλοφοριακής συμφόρησης (πραγματικού χρόνου) στο σύστημα μας.
- Ο υπολογισμός ρύπων, π.χ διοξειδίου του άνθρακα CO2, βασιζόμενοι στην εκτίμηση της απόστασης του βέλτιστου δρομολογίου που θα ακολουθηθεί.
- Η δυνατότητα σύνδεσης του προγράμματος F.O.R με το σύστημα G.P.S, με σκοπό την αξιοποίηση του γεωγραφικού στίγματος του οχήματος.
- Η δημιουργία εύχρηστου γραφικού περιβάλλοντος καθιστώντας την διαδικασία εισαγωγής δεδομένων απλή για τον κάθε χρήστη.

Η βελτίωση του προγράμματος F.O.R στο βαθμό που προαναφέραμε καθιστά την μελλοντική εφαρμογή του εφικτή όχι μόνο από οργανισμούς, δημόσιες υπηρεσίες και εταιρείες που ανήκουν στον τομέα διανομής αλλά και από κάθε οδηγό.

Π.χ ένας άνθρωπος που ζει σε μεγάλο αστικό κέντρο, εκτελεί κάθε μέρα το δρομολόγιο από το σπίτι του στον τόπο εργασίας του. Επιβιβάζεται στο όχημα του και προτού ξεκινήσει ενεργοποιεί το σύστημα F.O.R. Το σύστημα F.O.R αυτόματα ενημερώνεται για το παρών γεωγραφικό στίγμα του οχήματος από το σύστημα G.P.S και ζητά από τον οδηγό τον τελικό προορισμό. Έπειτα ενημερώνει μέσω του συστήματος G.P.S τον χρήστη για την βέλτιστη διαδρομή που θα ακολουθήσει. Μέχρι ο χρήστης να φτάσει στο τελικό προορισμό η βέλτιστη διαδρομή ανανεώνεται συνεχώς διότι αξιοποιούνται τα δεδομένα κυκλοφοριακής συμφόρησης πραγματικού χρόνου. Έτσι δίνεται η δυνατότητα

στον οδηγό να αποφύγει, σε πραγματικό χρόνο, την κυκλοφοριακή συμφόρηση που προκύπτει και να φτάσει όσο πιο γρήγορα, οικονομικά και ανώδυνα στο προορισμό εργασίας του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://www.logisticsworld.com/logistics.htm

[2] Transport Logistics Alan C. McKinnon, Kenneth John Button, Peter Nijkamp

[3] City Logistics -- Network modelling and Intelligent Transport Systems 2001 By Eiichi Taniguchi, Russell G. Thompson, Tadashi Yamada and Ron van Duin

[4] Marshall L. Fischer -- Network Routing

[5] Harilaos N. Psaraftis -- Vehicle Routing: Methods and Studies, chapter Dynamic Vehicle Routing Problems

[6] Ιωάννης Μαρινάκης , Αθανάσιος Μυγδαλάς -- Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας

[7] Nicos Christofides -- Worst-case analysis of a new heuristic for the travelling salesman problem

[8] http://www.tsp.gatech.edu/concorde/index.html

[9] http://www2.isye.gatech.edu/~wcook/qsopt/

[10] http://www.tsp.gatech.edu/methods/opt/zone.htm

[11] G. B. Dantzig, R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling salesman problem", Operations Research 2 (1954)

[12] http://www.tsp.gatech.edu/methods/opt/subtour.htm

[13] http://www.tsp.gatech.edu/methods/opt/cutting.htm