ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

ΕΥΡΕΣΗ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΜΕ ΧΡΗΣΗ ΑΣΤΙΚΗΣ ΣΥΓΚΟΙΝΩΝΙΑΣ ΓΙΑ ΤΗΝ ΠΟΛΗ ΤΟΥ ΒΟΛΟΥ

υπό

ΓΡΗΓΟΡΙΟΥ ΦΛΟΥΔΑ

Επιβλέπων Καθηγητής: Δρ. Σαχαρίδης Γεώργιος

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Σαχαρίδης Γεώργιος Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Κερμανίδης Αλέξιος Επίκουρος Καθηγητής, ΤμήμαΜηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Παντελής Δημήτριος Αναπληρωτής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα Δρ. Γεώργιο Σαχαρίδη, Επίκουρο Καθηγητή του Τμήματος Μηχανολόγων Μηχανικών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας, για την πολύτιμη επιστημονική και ηθική βοήθεια και καθοδήγησή του καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας. Ακόμη θα ήθελα να ευχαριστήσω τα υπόλοιπα μέλη της εξεταστικής επιτροπής κ. Κερμανίδη Αλέξιο και κ. Παντελή Δημήτριο για την προσεκτική ανάγνωση της εργασίας μου και για τις σημαντικές τους υποδείξεις. Ιδιαίτερη μνεία αξίζουν οι γονείς μου, οι οποίοι μου παρείχαν ηθική και οικονομική υποστήριξη όλα αυτά τα χρόνια. Ακόμη οφείλω ένα μεγάλο ευχαριστώ στην αδερφή μου και τους φίλους μου που όλα αυτά τα χρόνια με στηρίζουν πνευματικά και ηθικά. Αφιερώνω την παρούσα εργασία σε όλα τα παραπάνω άτομα που με βοήθησαν ο καθένας με τον δικό του τρόπο ώστε να την διεκπεραιώσω.

ΕΥΡΕΣΗ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΜΕ ΧΡΗΣΗ ΑΣΤΙΚΗΣ ΣΥΓΚΟΙΝΩΝΙΑΣ ΓΙΑ ΤΗΝ ΠΟΛΗ ΤΟΥ ΒΟΛΟΥ

ΦΛΟΥΔΑΣ ΓΡΗΓΟΡΙΟΣ

Πανεπιστήμιο Θεσσαλίας, Τμήμα Μηχανολόγων Μηχανικών, 2018

Επιβλέπων Καθηγητής:Δρ. Σαχαρίδης Γεώργιος, Επίκουρος Καθηγητής Επιχειρησιακής Έρευνας

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται το πρόβλημα δρομολόγησης με χρήση αστικής συγκοινωνίας τόσο με πραγματικούς όσο και με θεωρητικούς χρόνους άφιξης-αναχώρησης. Πιο συγκεκριμένα, προσεγγίζεται το πρόβλημα ενός χρήστη, ο οποίος επιθυμεί να μεταβεί από ένα σημείο της πόλης του Βόλου σε κάποιο άλλο χρησιμοποιώντας μόνο λεωφορεία του Αστικού Κτελ Βόλου με τον βέλτιστο τρόπο. Ακόμη, γίνεται μια σύντομη αναφορά στη Θεωρία των Γράφων, καθώς σε αυτήν βασίζεται η προσέγγιση του προβλήματος δρομολόγησης που μελετάται. Στη συνέχεια, παρουσιάζεται η μοντελοποίηση τόσο του προαναφερθέντος προβλήματος όσο και ενός προβλήματος που περιέχει επιπλέον έναν ευρετικό κανόνα, ο οποίος υπολογίζει ταχύτερα τη βέλτιστη ή κοντά σε αυτήν λύση. Τόσο το αρχικό πρόβλημα όσο και αυτό με τον ευρετικό κανόνα εφαρμόζονται και για θεωρητικά (Theoritical δεδομένα) και για πραγματικά δεδομένα (GTFS δεδομένα). Τα 2 μοντέλα, που προκύπτουν, υλοποιούνται σε γλώσσα προγραμματισμού C++ με ακέραιο προγραμματισμό. Στόχος του 1^{ου} προβλήματος (M1) είναι η εύρεση της βέλτιστης λύσης, ενώ του 2^{ου} (M2) η ταχύτερη εύρεση εφικτής λύσης που μπορεί να είναι είτε η βέλτιστη είτε να έχει μικρή απόκλιση από αυτήν. Επίσης, γίνεται αναφορά στο πρόγραμμα OpenTripPlanner (OTP) και πραγματοποιείται σύγκριση με τα μοντέλα ακέραιου προγραμματισμού (M1και M2) και για τα δύο είδη δεδομένων (Theoritical και GTFS) όσον αφορά τους χρόνους επίλυσής τους. Τέλος, εξάγονται τα συμπεράσματα που προκύπτουν από την παρούσα εργασία.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΑΚΡΩΝΥΜΙΩΝ

GTFS : General Transit Feed Specification

OTP : OpenTripPlanner

OSM : OpenStreetMap

O : Σημείοαναχώρησης

D : Σημείο άφιξης

ToA : Επιθυμητή ώρα άφιξης στον προορισμό

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγικά

Σε αυτό το κεφάλαιο παρουσιάζουμε πληροφορίες εισαγωγικού χαρακτήρα που δίνουν το κίνητρο και το υπόβαθρο αυτής της διπλωματικής εργασίας και περιγράφουμε συνοπτικά τις βασικές ενότητές της.

1.2 Κίνητρο και Υπόβαθρο

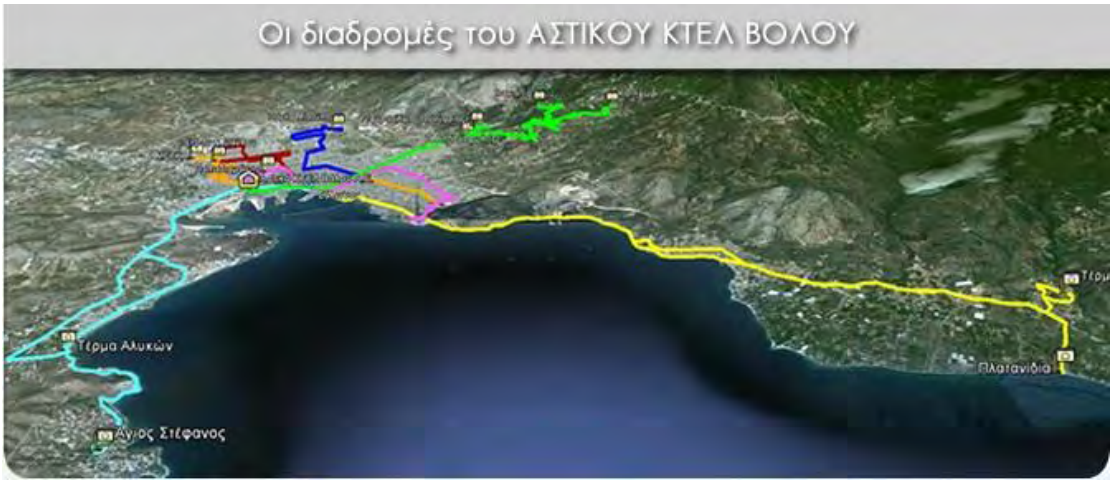
Είναι κοινώς αποδεκτό πως στη σύγχρονη κοινωνία ο σχεδιασμός διαδρομής αποκτά όλο και μεγαλύτερη σημασία. Τόσο η κινητικότητα στην κοινωνία όσο και η πολυπλοκότητα των δικτύων μεταφοράς ώθησε την επιστημονική κοινότητα να αναζητήσει πιο αποτελεσματικές μεθόδους στον σχεδιασμό της διαδρομής. Εξάλλου, δεν είναι τυχαίο πως πολλές εταιρείες έχουν αναπτύξει σύστημα πλοήγησης (GPS) για τα αυτοκίνητα, το οποίο δίνει τη δυνατότητα σε έναν ταξιδιώτη να ορίσει τον σχεδιασμό της διαδρομής. Επίσης, κάποιες αεροπορικές και σιδηροδρομικές εταιρείες προσφέρουν μέσω του Διαδικτύου κάποιο είδος εφαρμογών σχεδιασμού διαδρομής.

Ακόμη παρατηρείται πως οι κάτοικοι των μεγάλων πόλεων στρέφονται, με την πάροδο του χρόνου, όλο και περισσότερο προς τις αστικές συγκοινωνίες, ώστε να μειώσουν το φόρτο του οδικού δικτύου και να βελτιώσουν την ποιότητα ζωής τους. Η αναζήτηση του τρόπου μετάβασης από μία περιοχή σε κάποια άλλη αποτελεί πλέον ένα πολύπλοκο πρόβλημα που η επίλυσή του απαιτεί καλή γνώση του συγκοινωνιακού και οδικού δικτύου. Η ανάγκη αυτή, σε συνδυασμό με την εισαγωγή του Διαδικτύου στην καθημερινότητά μας, έχει δημιουργήσει έντονη ερευνητική δραστηριότητα γύρω από τον τομέα των αστικών συγκοινωνιών και της αποδοτικότερης χρήσης αυτών.

Στόχος ύπαρξης της δρομολόγησης αστικών μέσων είναι να δίνεται η δυνατότητα στον εκάστοτε χρήστη να μετακινηθεί από ένα σημείο μιας πόλης σε ένα άλλο με το βέλτιστο τρόπο γι'αυτόν. Ο βέλτιστος τρόπος δρομολόγησης ορίζεται κάθε φορά από το είδος του προβλήματος και συνήθως έχει να κάνει με την ελαχιστοποίηση συνολικού κόστους ή χρόνου μετακίνησης. Είναι σημαντικό να αναφερθεί πως ο χρήστης επιθυμεί η βέλτιστη λύση να υπολογίζεται όσο το δυνατόν ταχύτερα.

Στο πλαίσιο αυτό, η παρούσα εργασία παρουσιάζει τη μοντελοποίηση με ακέραιο προγραμματισμό ενός προβλήματος δρομολόγησης με αστική συγκοινωνία για την πόλη του

Βόλου. Στόχος είναι η εύρεση της βέλτιστης διαδρομής για έναν χρήστη που μετακινείται στην πόλη του Βόλου με Αστικό Κτελ με συγκεκριμένες ώρες άφιξης και αναχώρησης. Είναι σημαντικό να αναφερθεί πως σε αυτή την εργασία επιλύουμε το ίδιο πρόβλημα και με χρήση του προγράμματος OTP.



Εικόνα 1.1: Διαδρομές του Αστικού Κτελ Βόλου

1.3 Οργάνωση Διπλωματικής Εργασίας

Κεφάλαιο 1^0: Σε αυτό το κεφάλαιο γίνεται εισαγωγή περί δρομολόγησης με αστική συγκοινωνία και παρουσιάζεται η οργάνωση της διπλωματικής εργασίας.

Κεφάλαιο 2^0: Στο δεύτερο κεφάλαιο γίνεται αναφορά στη Θεωρία των Γράφων, δίνονται κάποιοι βασικοί ορισμοί, παρουσιάζονται οι τρόποι αναπαράστασής τους, ενώ γίνεται σύντομη αναφορά και στα δίκτυα.

Κεφάλαιο 3^0: Σε αυτό το κεφάλαιο γίνεται μία σύντομη αναφορά στον ακέραιο προγραμματισμό και στη δρομολόγηση. Περιγράφεται και μοντελοποιείται το πρόβλημα της βέλτιστης διαδρομής με χρήση αστικής συγκοινωνίας και γίνεται μία περαιτέρω βελτίωση αυτού με χρήση ευρετικού κανόνα. Επίσης, αναφέρονται κάποιες βασικές πληροφορίες για τα GTFS δεδομένα.

Κεφάλαιο 4^0: Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα από την υλοποίηση των μοντέλων, που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, πάνω σε 6 παραδείγματα και για τους δύο τύπους δεδομένων (θεωρητικά και GTFS) και ακόμη γίνεται συγκριτική ανάλυση των δύο μοντέλων με βάση τους υπολογιστικούς χρόνους.

Κεφάλαιο 5^0: Στο πέμπτο κεφάλαιο γίνεται αναφορά στο πρόγραμμα OTP. Δημιουργείται ένα γράφημα OTP για την πόλη του Βόλου και παρουσιάζονται τα αποτελέσματα από τα παραδείγματα που εφαρμόσαμε. Τέλος, πραγματοποιείται σύγκριση του προγράμματος OTP με αυτό της C++ (μοντέλο με ακέραιο προγραμματισμό).

Κεφάλαιο 6ο: Σε αυτό το τελευταίο κεφάλαιο αναφέρουμε τα συμπεράσματα της διπλωματικής εργασίας.

Παράρτημα: Στο παράρτημα παρατίθεται το μαθηματικό μοντέλο M1 για τα θεωρητικά δεδομένα και το μαθηματικό μοντέλο M1 για τα GTFS δεδομένα.

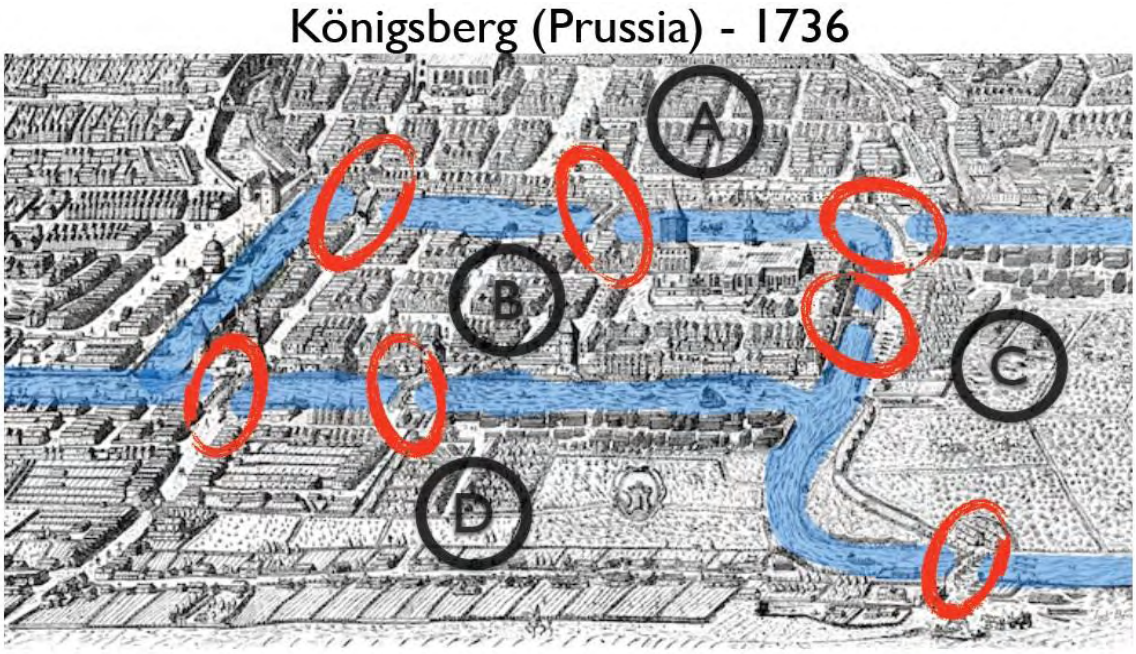
ΚΕΦΑΛΑΙΟ 2: ΘΕΩΡΙΑ ΓΡΑΦΩΝ

2.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει αναφορά στη Θεωρία των Γράφων, καθότι η προσέγγιση του προβλήματος δρομολόγησης, που μελετάμε, βασίζεται σε αυτήν. Θα παρουσιαστούν οι βασικές έννοιες και τα βασικά χαρακτηριστικά ενός γράφου, ενώ θα γίνει και σύντομη αναφορά στα δίκτυα.

2.2 Ιστορική Αναδρομή

Ο όρος γράφος παρουσιάστηκε για πρώτη φορά σε μία δημοσίευση το 1878, ενώ το πρώτο έντυπο εγχειρίδιο για Θεωρία Γράφων εκδόθηκε το 1936 από τον Denis Konig. Ωστόσο, η αρχή για την ανάπτυξη της Θεωρίας των Γράφων έγινε όταν ο Leonard Euler μελέτησε για τις Επτά Γέφυρες του Konigsberg το 1736. Η πόλη του Konigsberg βρισκόταν στην Πρωσία και περιελάμβανε δύο μεγάλα νησιά (B και C), τα οποία συνδέονταν μεταξύ τους αλλά και με την ηπειρωτική χώρα (A και D) με επτά γέφυρες. Το πρόβλημα αυτό μελετούσε κατά πόσο είναι δυνατόν να υπάρξει μία διαδρομή σύμφωνα με την οποία θα διασχίζεται κάθε γέφυρα ακριβώς μία φορά και θα επιστρέφουμε πάλι στο σημείο εκκίνησης.



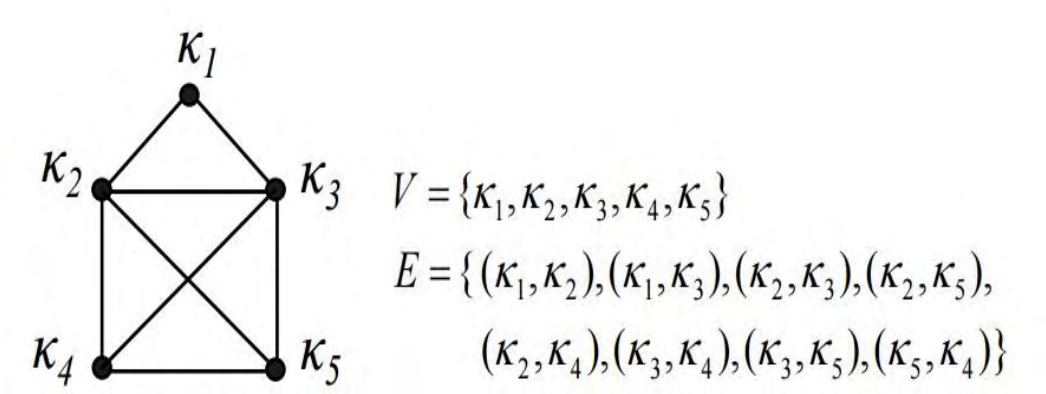
Εικόνα2.1: Επτά Γέφυρες του Konigsberg

Η δημοσίευση του Euler, όπως κι εκείνη που γράφτηκε από τον Γάλλο χημικό Alexandre-Theophile Vandermonde στο μαθηματικό πρόβλημα του Ίππου στη σκακιέρα που κατευθύνεται με την ανάλυση θέσης που εισήγαγε ο Gottfried Wilhelm Leibniz, αποτέλεσαν τη βάση για την ανάπτυξη της Θεωρίας των Γράφων. Ο τύπος του Euler, σχετικά με τον αριθμό των ακμών, των κορυφών και των εδρών ενός κυρτού πολυέδρου μελετήθηκε από τον Augustin-Louis Cauchy και τον Simon Antoine Jean L'Huillier. Το 1857 ο Arthur Cayley μελέτησε μια ιδιαίτερη κατηγορία γράφων, τα δέντρα. Η μελέτη των δέντρων είχε πολλές εφαρμογές στη θεωρητική χημεία. Οι τεχνικές που αναπτύχθηκαν είχαν να κάνουν κυρίως με την απαρίθμηση γράφων που παρουσίαζαν κάποιες ιδιαίτερες ιδιότητες. Η απαριθμητική θεωρία γράφων ήταν ένα από τα συμπεράσματα του Cayley και δημοσιεύτηκε από τον George Polya μεταξύ των ετών 1935 και 1937, ενώ η γενίκευση των συμπερασμάτων εκδόθηκε από τον Nicolaas Govert de Bruijn το 1959. Ο Cayley συνέδεσε τα συμπεράσματά του για τα δέντρα με τις σύγχρονες μελέτες για τη χημική σύνθεση. Η σύνθεση των μαθηματικών και των χημικών εννοιών είναι το αρχικό τμήμα της στερεότυπης ορολογίας της Θεωρίας των Γράφων.

2.3 Βασικές Έννοιες Θεωρίας των Γράφων

Ο γράφος G είναι ένα διατεταγμένο ζεύγος G=(V,E) όπου $V=\{\kappa_1,\kappa_2,\ldots,\kappa_n\}$ το σύνολο των κορυφών, το οποίο είναι μη κενό πεπερασμένο σύνολο, και $E=\{\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_m\}$ το σύνολο των ακμών, που είναι σύνολο από διμελή σύνολα κορυφών (είναι μη διατεταγμένο ζεύγος κόμβων).

Η ακμή ε_1 είναι $\varepsilon_1=(\kappa_1,\kappa_2)$ αν και μόνο αν η ε_1 είναι προσπίπτουσα στους κόμβους κ_1,κ_2.



Εικόνα 2.2: Παράδειγμα Γράφου

Οι γράφοι μπορούν να χωριστούν σε τρεις κατηγορίες:

- **Κατευθυνόμενος (directed graph):** είναι ο γράφος, του οποίου κάθε μια από τις ακμές του είναι προσανατολισμένη προς μία κατεύθυνση.
- **Μη κατευθυνόμενος (undirected graph):** είναι ο γράφος, του οποίου οι ακμές δεν είναι προσανατολισμένες.
- **Μεικτός (mixed graph):** είναι ο γράφος, του οποίου κάποιοι κλάδοι έχουν διεύθυνση και κάποιοι όχι.

Αν (κ_1,κ_2) είναι ακμή τότε λέμε ότι οι κορυφές κ_1 και κ_2 είναι **γειτονικές** (adjacent) ή ότι **γειτνιάζουν**.

Ο **βαθμός** ενός κόμβου σε έναν μη κατευθυνόμενο γράφο είναι ο αριθμός των κλάδων των οποίων μία κορυφή είναι αυτός ο κόμβος, ενώ σε έναν προσανατολισμένο γράφο ορίζεται ο βαθμός για τον αριθμό των κλάδων που καταλήγουν σε αυτόν τον κόμβο (indegree) και ο βαθμός για τον αριθμό των κλάδων που απομακρύνονται απο αυτόν τον κόμβο (outdegree).

Ο αριθμός των κορυφών του γράφου καλείται **τάξη (order)** του γράφου, ενώ ο αριθμός των ακμών του γράφου καλείται **μέγεθος (size)** του γράφου.

Δύο ακμές προσπίπτουσες στο ίδιο ζεύγος κόμβων ονομάζονται **παράλληλες**. Μία ακμή προσπίπτουσα στον ίδιο κόμβο ονομάζεται **βρόγχος (loop)**. Ένας γράφος χωρίς παράλληλες ακμές και βρόγχους ονομάζεται **απλός**.

Μονοπάτι ή **διαδρομή (path)** ενός γράφου μήκους n, είναι μία ακολουθία κόμβων κ_o, κ_1,..., κ_n, όπου για κάθε i, $0 \leq i < n$, $(\kappa_i, \kappa_i + 1)$ είναι ακμή του γράφου. **Μήκος** ενός μονοπατιού είναι ο αριθμός ακμών που περιέχει.

Μία διαδρομή ενός γράφου ονομάζεται **απλή (simple)** αν όλες οι κορυφές της είναι διαφορετικές μεταξύ τους, εκτός από την πρώτη και την τελευταία οι οποίες μπορούν να είναι οι ίδιες.

Ένας γράφος ονομάζεται **συνδεδεμένος** αν υπάρχει μονοπάτι μεταξύ δύο οποιονδήποτε κόμβων του και **επίπεδος** αν μπορεί να σχεδιαστεί σε ένα επίπεδο χωρίς να τέμνονται μεταξύ τους καμία από τις ακμές του.

Η **απόσταση** δύο κορυφών είναι το μήκος της συντομότερης διαδρομής που οδηγεί από την μία κορυφή στην άλλη.

Ένας μη κατευθυνόμενος γράφος λέγεται **συνεκτικός (connected)** αν για κάθε ζευγάρι κορυφών υπάρχει διαδρομή που τις συνδέει. Ένας κατευθυνόμενος γράφος που ικανοποιεί την ίδια ιδιότητα ονομάζεται **ισχυρά συνεκτικός (strongly connected)**. Αν ο μη-κατευθυνόμενος γράφος στον οποίο αντιστοιχεί είναι συνεκτικός, τότε ο γράφος ονομάζεται **ελαφρά συνεκτικός (weakly connected)**.

Κύκλος (cycle) ονομάζεται ένα μονοπάτι του οποίου η αρχή και το τέλος είναι ο ίδιος κόμβος.

Ένας γράφος που δεν περιέχει κύκλους ονομάζεται **άκυκλος (acyclic).**

Ένας γράφος λέγεται **αραιός (sparse)** αν ο αριθμός των ακμών του είναι της τάξης O(n), όπου n είναι ο αριθμός κορυφών του, ενώ σε αντίθετη περίπτωση λέγεται **πυκνός (dense)**.

Συχνά συσχετίζουμε κάθε ακμή ενός γράφου με κάποιο **βάρος (weight)**. Τότε ο γράφος ονομάζεται **γράφος με βάρη (weighted graph)**.

Ένα **δέντρο** είναι ένας συνεκτικός γράφος χωρίς κύκλους. Έτσι, ένα δέντρο σε ένα γράφο με n κόμβους περιέχει ακριβώς n - 1 κλάδους. Επίσης, κάθε ζεύγος κόμβων ενός δέντρου συνδέονται μέσω ενός μοναδικού μονοπατιού. Ένα **δέντρο κάλυψης (spanning tree)** ενός γράφου G είναι ένα δέντρο που περιέχει όλους τους κόμβους του G.

Διαδρομή Euler (Euler path) καλείται η διαδρομή, η οποία περνάει από όλες τις ακμές του γράφου ακριβώς μία φορά.

Διαδρομή Hamilton (Hamilton path) καλείται η διαδρομή, η οποία περνάει από όλους τους κόμβους του γράφου ακριβώς μία φορά.

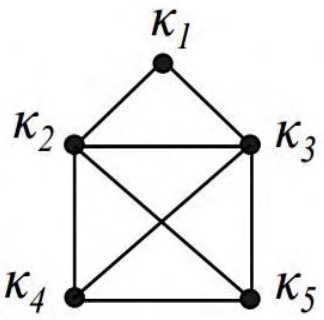
Υπογράφος (subgraph) B ενός γράφου A καλείται ο γράφος του οποίου όλες οι ακμές και οι κόμβοι περιέχονται στον A.

2.3.1 Αναπαράσταση Γράφων

Ο γράφος G=(V,E) μπορεί να αναπαρασταθεί από έναν πίνακα πρόσπτωσης (incidence matrix). Ο πίνακας έχει κ_nxκ_m μέγεθος, όπου κ_n οι κορυφές και κ_m οι ακμές. Οι γραμμές του πίνακα αριθμούνται με βάση τις κορυφές και οι στήλες με βάση τις ακμές. Αν η κορυφή κ_i είναι ένα από τα άκρα της ακμής κ_j, το στοιχείο α(i,j) του πίνακα πρόσπτωσης είναι ίσο με τη μονάδα, ενώ σε αντίθετη περίπτωση λαμβάνει την τιμή μηδέν.

Επίσης, ο γράφος G=(V,E) μπορεί εναλλακτικά να αναπαρασταθεί από έναν πίνακα γειτνίασης (adjacency matrix). Ο πίνακας έχει $\kappa_n x \kappa_n$ μέγεθος. Κάθε μη μηδενικό στοιχείο του πίνακα φανερώνει την ύπαρξη μίας ακμής μεταξύ των κορυφών της αντίστοιχης γραμμής και στήλης. Σε περίπτωση που η ακμή χαρακτηρίζεται από κάποια παράμετρο, π.χ βάρος, το αντίστοιχο στοιχείο του πίνακα θα είναι μία εγγραφή που θα περιέχει την τιμή της παραμέτρου αυτής.

Ο πίνακας γειτνίασης ενός μη κατευθυνόμενου γραφήματος είναι συμμετρικός, ενώ ενός κατευθυνόμενου δεν είναι.

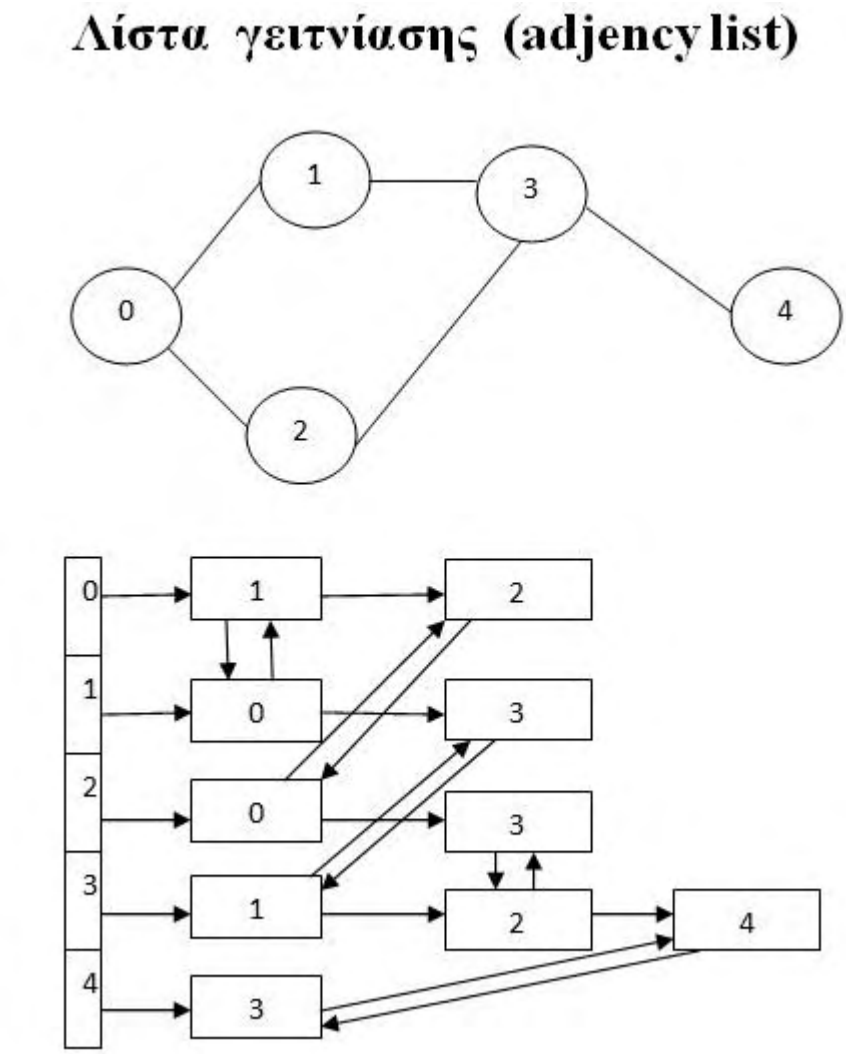


	K_1	K_2	K_3	K_4	K_5
K_1	0	1	1	0	0
K_2	1	0	1	1	1
K_3	1	1	0	1	1
K_4	0	1	1	0	1
K_5	0	1	1	1	0

Εικόνα 2.3: Παράδειγμα Γράφου με Πίνακα Γειτνίασης

2.3.2 Λίστες Γειτνίασης

Η αναπαράσταση σε μία λίστα που θα στηρίζεται στην διασύνδεση των κόμβων αποδεικνύεται ο καλύτερος τρόπος για την αποθήκευση ενός γράφου. Έτσι, σε έναν γράφο με κ_n κόμβους, κάθε κόμβος συσχετίζεται με μία συνδεδεμένη λίστα γειτνίασης, η οποία περιέχει όλα τα ονόματα των άλλων κόμβων με τους οποίους συνδέεται ο συγκεκριμένος κόμβος. Οι λίστες γειτνίασης για όλους τους κόμβους μπορούν να ξεκινούν από έναν μονοδιάστατο πίνακα μήκους κ_n ή από μία άλλη συνδεδεμένη λίστα.



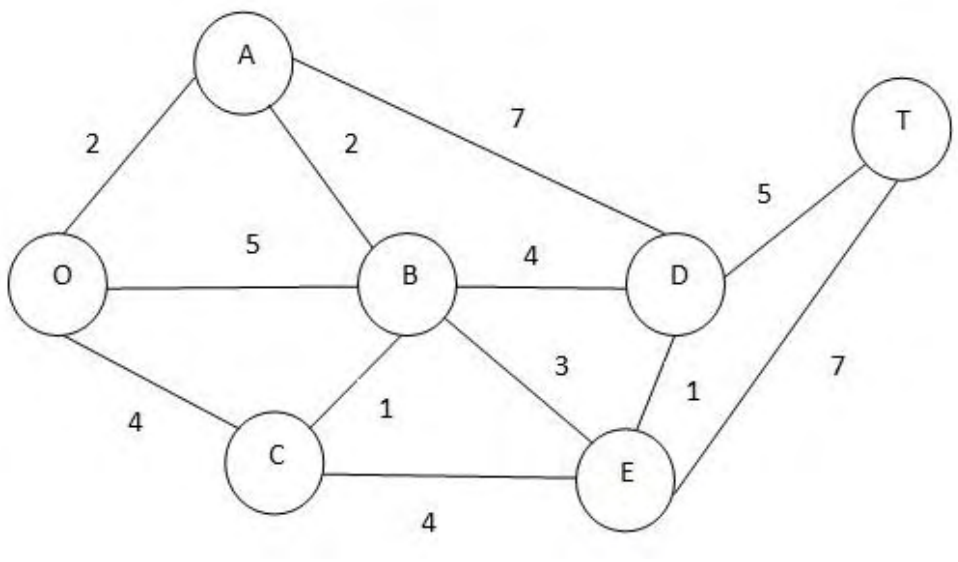
Εικόνα 2.4: Παράδειγμα Αναπαράστασης Γράφου σε Μορφή Λίστας

2.4 Δίκτυα

Στη σύχγρονη κοινωνία παρατηρούμε πως η παρουσία των δικτύων είναι ιδιαίτερα εμφανής. Δίκτυα επικοινωνίας καθιστούν εφικτή την επικοινωνία μεταξύ απομακρυσμένων περιοχών. Η μεταφορά ενέργειας για οικιακή και βιομηχανική χρήση γίνεται μέσω κατάλληλων δικτύων. Ακόμη, υπάρχουν τα οδικά δίκτυα που παρέχουν τη δυνατότητα κάλυψης μεγάλων αποστάσεων σε μικρό χρονικό διάστημα. Σε αυτά τα δίκτυα, σκοπός είναι η μεταφορά ενός χρήστη μεταξύ δύο σημείων με τον βέλτιστο τρόπο. Η θεωρία των δικτύων είναι μία επιστημονική περιοχή που βρίσκεται στο μεταίχμιο μεταξύ διαφόρων ερευνητικών αντικειμένων όπως οι επιστήμες του μηχανικού, της διοίκησης και της επιχειρησιακής έρευνας. Η εύρεση της θεωρίας των δικτύων οφείλεται στην δουλειά του Kirchhof καθώς και

άλλων ερευνητών, οι οποίοι πρώτοι μελέτησαν συστηματικά ηλεκτρικά κυκλώματα. Αυτή η δουλειά αποτέλεσε την αφετηρία για τη μετέπειτα πορεία και εξέλιξη της θεωρίας των δικτύων.

Δίκτυο είναι ένα διάγραμμα, το οποίο αναπαριστά τη σχεδίαση κάποιου συστήματος μεταφοράς-επικοινωνίας. Σχεδιάζεται ως ένα σύνολο κύκλων (κόμβοι), οι οποίοι συνδέονται μεταξύ τους με γραμμές (ακμές). Μεταξύ των κόμβων και διαμέσου των ακμών ρέουν διάφορα υλικά. Κάθε κόμβος συμβολίζεται με έναν αριθμό (ή γράμμα ή λέξη). Οι αριθμοί αυτοί χρησιμοποιούνται και για τον συμβολισμό των ακμών.



Εικόνα 2.5: Παράδειγμα Δικτύου

Σε ένα πρόβλημα δικτύου ορίζονται κάποια χαρακτηριστικά για κάθε κόμβο και κλάδο. Συνήθως υπάρχει κάποιο μέγεθος για κάθε κλάδο του δικτύου, το οποίο μπορεί να παριστάνει απόσταση, χρόνο, χωρητικότητα, ροή κτλ. Το μήκος ενός μονοπατιού μεταξύ δύο σημείων ενός δικτύου ισούται προφανώς με το άθροισμα των αποστάσεων όλων των κλάδων στο συγκεκριμένο μονοπάτι. Ο συμβολισμός d(i,j) χρησιμοποιείται συνήθως για να υποδηλώσει την ελάχιστη απόσταση μεταξύ των κόμβων i και j.

Είναι σημαντικό να αναφερθεί πως η έννοια του δικτύου είναι άμεσα συνδεδεμένη με αυτήν του γράφου, καθώς η βασική διαδικασία απλοποίησης των δικτύων είναι η μετατροπή τους σε γραφήματα. Μία από τις πιο σημαντικές ιδιότητες ενός δικτύου αποτελεί η συνδετικότητα. Ως συνδετικότητα ενός δικτύου ορίζεται ο βαθμός σύνδεσης μεταξύ των

κόμβων. Ο βαθμός συνδετικότητας ενός συγκοινωνιακού δικτύου είναι ενδεικτικός της πολυπλοκότητας του χωρικού συστήματος που εξυπηρετείται από το συγκεκριμένο δίκτυο.

Τόσο για την αξιολόγηση της δομής ενός δικτύου όσο και για την σύγκριση της πολυπλοκότητας της δομής δύο ή περισσοτέρων δικτύων χρησιμοποιούνται οι παρακάτω δείκτες:

- **Δείκτης Γάμμα:** Μετρά την πυκνότητα των συνδέσεων. Ως συνδέσεις ορίζονται οι ακμές ενός δικτύου. Ο δείκτης γάμμα συγκρίνει τον πραγματικό αριθμό συνδέσεων που υπάρχουν σε ένα δίκτυο με τον μέγιστο δυνατό αριθμό συνδέσεων που μπορούν να υπάρξουν στο δίκτυο αυτό και ορίζεται ως εξής:

  $\gamma = \frac{\text{υπάρχοντες συνδέσεις}}{\text{μέγιστος αριθμός συνδέσεων}} = \frac{\sigma}{\sigma max}$, για επίπεδο δίκτυο σmax = 3*(κ-2), όπου κ είναι ο αριθμός των κόμβων
- **Δείκτης Άλφα:** Μετρά την πυκνότητα των κυκλικών συνδέσμων. Ως κυκλικοί σύνδεσμοι ορίζονται οι συγκεκριμένες διαδρομές στο δίκτυο όπου ο αρχικός κόμβος της αλληλουχίας των συνδέσμων (ακμών) συμπίπτει με τον τελικό κόμβο. Ο δείκτης άλφα συγκρίνει τον αριθμό των πραγματικών κυκλικών συνδέσμων που υπάρχουν σε ένα δίκτυο με τον μέγιστο δυνατό αριθμό τους και ορίζεται ως εξής:

  $\alpha = \frac{\text{υπάρχοντες κυκλικοί σύνδεσμοι}}{\text{μέγιστος αριθμός κυκλικών συνδέσμων}} = \frac{\sigma - \kappa + 1}{2*\kappa - 5}$
- **Διάμετρος Δικτύου:** Ως διάμετρος ενός συνδεδεμένου δικτύου ορίζεται ο μέγιστος αριθμός συνδέσμων (συνδέσεων) που απαιτείται για την μετακίνηση από έναν κόμβο σε έναν άλλο μέσω μιας ελάχιστης διαδρομής.

2.5 Σύνοψη

Η Θεωρία των Γράφων αποτέλεσε τη βάση γιατην ανάπτυξη των προβλημάτων δρομολόγησης. Γι'αυτόν το λόγο δόθηκαν κάποιοι βασικοί ορισμοί στη Θεωρία των Γράφων, οι τρόποι αναπαράστασης αυτών, ενώ ακόμη έγινε και αναφορά στα δίκτυα. Στο επόμενο κεφάλαιο θα γίνει αναλυτικά η μοντελοποίηση του προβλήματος δρομολόγησης με αστική συγκοινωνία για την πόλη του Βόλου.

ΚΕΦΑΛΑΙΟ 3: ΕΥΡΕΣΗ ΣΥΝΤΟΜΟΤΕΡΗΣ ΔΙΑΔΡΟΜΗΣ ΜΕΤΑΞΥ ΔΙΑΦΟΡΩΝ ΣΗΜΕΙΩΝ ΤΗΣ ΠΟΛΗΣ ΤΟΥ ΒΟΛΟΥ ΜΕ ΜΟΝΤΕΛΟ ΑΚΕΡΑΙΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα ασχοληθούμε με την μοντελοποίηση του προβλήματος δρομολόγησης που μας απασχολεί. Αρχικά θα γίνει γενική αναφορά περί δρομολόγησης αλλά και ακέραιου προγραμματισμού, καθώς το πρόβλημα στηρίζεται στον συνδυασμό αυτών των δύο όρων. Στη συνέχεια, θα παραθέσουμε κάποιες πληροφορίες για τα GTFS δεδομένα και θα διατυπωθεί το πρόβλημα. Στο υπόλοιπο κεφάλαιο θα γίνει η παρουσίαση των δύο μοντέλων, που πραγματευόμαστε, προκείμενου να επιλυθούν στο δίκτυο της πόλης του Βόλου.

3.2 Γενικά περί Ακέραιου Προγραμματισμού και Δρομολόγησης

Σε αυτή την ενότητα θα αναφερθούμε τόσο στον ακέραιο προγραμματισμό όσο και στην δρομολόγηση, αφού η προσέγγιση του πρόβληματος βελτιστοποίησης, που μελετάμε, βασίζεται σε αυτές τις έννοιες.

3.2.1 Ακέραιος Προγραμματισμός

Αποτελεί μία από τις πλέον διαδεδομένες τεχνικές της Επιχειρησιακής Έρευνας και υπάγεται στη γενικότερη κατηγορία του μαθηματικού προγραμματισμού. Ο ακέραιος προγραμματισμός (integer programming) περιλαμβάνει όλα τα προβλήματα στα οποία οι μεταβλητές απόφασης μπορούν να λάβουν ακέραιες τιμές. Ένα πρόβλημα ακέραιου προγραμματισμού μπορεί να είναι είτε γραμμικό είτε μη γραμμικό. Έτσι, τόσο η συνάρτηση προς βελτιστοποίηση μπορεί να είναι γραμμική ή μη γραμμική όσο και οι μαθηματικές σχέσεις που αποτελούν τους περιορισμούς, δηλαδή οι σχέσεις που συνδέουν τις διάφορες μεταβλητές του προβλήματος μεταξύ τους, μπορούν να είναι γραμμικές ή μη γραμμικές. Επίσης, οι περιορισμοί είναι είτε εξισώσεις είτε ανισότητες. Οι λύσεις εμφανίζονται στην εφικτή περιοχή, η οποία καθορίζεται από τους περιορισμούς, ενώ ως βέλτιστη ορίζεται εκείνη η λύση που βελτιστοποιεί (μεγιστοποιεί ή ελαχιστοποιεί ανάλογα με το είδος του προβλήματος) την τιμή της αντικειμενικής συνάρτησης.

Ο ακέραιος προγραμματισμός χωρίζεται σε τρεις κατηγορίες:

- **Μεικτός ακέραιος προγραμματισμός (mixed integer programming):** είναι η κατηγορία προβλημάτων ακέραιου προγραμματισμού όπου κάποιες από τις μεταβλητές απόφασης περιορίζονται σε ακέραιες τιμές και κάποιες όχι.
- **Αμιγώς ακέραιος προγραμματισμός (pure integer programming):** είναι η κατηγορία προβλημάτων ακέραιου προγραμματισμού όπου όλες οι μεταβλητές απόφασης λαμβάνουν μόνον ακέραιες τιμές.
- **Δυαδικός ακέραιος προγραμματισμός (binary integer programming):** είναι μια ειδική κατηγορία προβλημάτων ακέραιου προγραμματισμού όπου οι μεταβλητές απόφασης μπορούν να πάρουν μόνο τιμές 0 ή 1.

3.2.2 Δρομολόγηση

Ο όρος δρομολόγηση αναφέρεται στη διαδικασία με την οποία επιλέγεται η διαδρομή μέσα σε ένα δίκτυο που περιέχει δεδόμενα. Το δίκτυο αποτελείται από κόμβους και ακμές (δυνατές μεταβιβάσεις μεταξύ κόμβων), οι οποίες χαρακτηρίζονται από κόστη που μπορεί να είναι αποστάσεις, χρόνος μετακίνησης, οικονομική επιβάρυνση κτλ, ανάλογα με το είδος του προβλήματος που μελετάται. Η δρομολόγηση ως διαδικασία σχετίζεται με οποιοδήποτε δίκτυο, π.χ ηλεκτρονικό δίκτυο, δίκτυο μεταφορών, δίκτυο επικοινωνίας.

Στόχος της δρομολόγησης είναι η ανακάλυψη μονοπατιών και βάσει αυτών η κατασκευή και ενημέρωση των πινάκων δρομολόγησης, αλλά κυριότερα είναι η εύρεση της βέλτιστης διαδρομής. Η βέλτιστη διαδρομή καθορίζεται από το είδος του προβλήματος και συχνότερα είναι τέτοια που να ελαχιστοποιεί το συνολικό κόστος μετακίνησης ή τον συνολικό χρόνο μεταφοράς. Γι’ αυτόν το λόγο και το πρόβλημα της δρομολόγησης αποτελεί ένα από τα πιο πολυσυζητημένα προβλήματα στον επιστημονικό χώρο της βελτιστοποίησης.

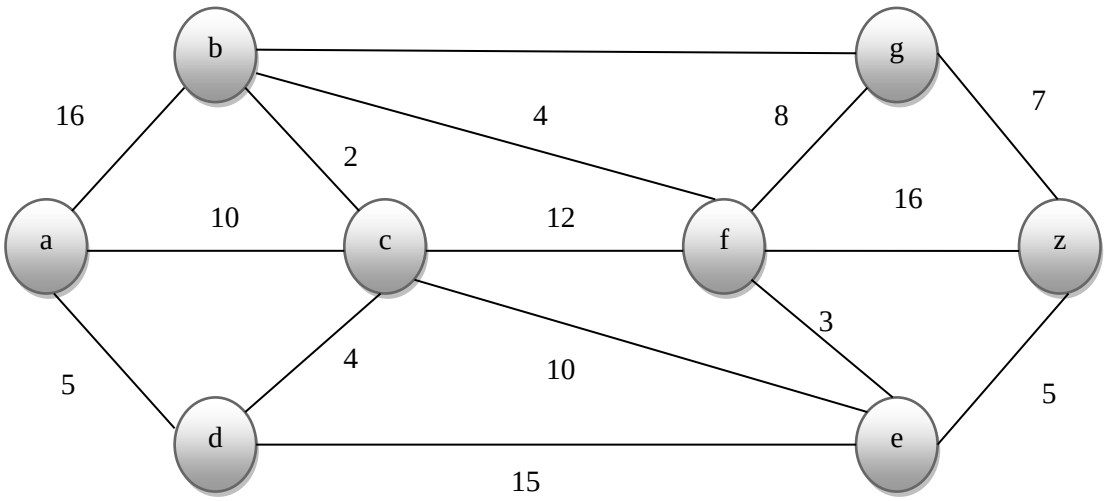
Τα σημαντικότερα προβλήματα δρομολόγησης είναι: το πρόβλημα του πλανόδιου πωλητή, το πρόβλημα δρομολόγησης οχημάτων και το πρόβλημα της συντομότερης διαδρομής. Στην παρακάτω υποενότητα γίνεται αναφορά στο πρόβλημα συντομότερης διαδρομής, καθότι το πρόβλημα βελτιστοποίησης που πραγματεύεται η παρούσα εργασία ανήκει σε αυτή την κατηγορία.

3.2.3 Πρόβλημα Συντομότερης Διαδρομής

Το πρόβλημα της συντομότερης διαδρομής είναι ένα από τα βασικότερα προβλήματα δρομολόγησης. Αναφέρεται στην εύρεση της βέλτιστης διαδρομής μεταξύ ενός κόμβου εκκίνησης (αφετηρία) και ενός κόμβου τερματισμού (προορισμός) διαμέσου ενός συνεκτικού δικτύου, του οποίου είναι γνωστές οι αποστάσεις των αντίστοιχων κλάδων. Σε αυτό το πρόβλημα η παραγωγή στον αρχικό κόμβο και η ζήτηση στον τελικό κόμβο είναι ακριβώς μία μονάδα και ακόμη η απόσταση μίας διαδρομής είναι ίση με το κόστος κατά μήκος ενός τόξου.

Η τεχνική της συντομότερης διαδρομής στηρίζεται στο γεγονός πως σε κάθε βήμα μπορεί να βρεθεί ένας τουλάχιστον κόμβος, για τον οποίο η διαδρομή από την αφετηρία μέχρι αυτόν δεν μπορεί να βελτιωθεί περαιτέρω. Ο κόμβος αυτός, τότε, ονομάζεται μόνιμος (permanent). Εν συνεχεία, εξετάζεται αν ο κόμβος αυτός μπορεί να χρησιμοποιηθεί ως ενδιάμεσος, δηλαδή για τη βελτίωση των προσωρινών διαδρομών που έχουν βρεθεί για τους υπόλοιπους κόμβους στο δικτύου συμπεριλαμβανομένου και του προορισμού. Η διαδικασία επαναλαμβάνεται μέχρι να γίνει μόνιμος ο προορισμός. Ο αλγόριθμος τερματίζει όταν όλοι οι κόμβοι γίνουν μόνιμοι και βρίσκουμε έτσι τη συντομότερη διαδρομή προς κάθε κόμβο από την αφετηρία.

Έχουν προταθεί διάφορες μέθοδοι επίλυσης, ωστόσο ο αλγόριθμος που δίνει τις πιο γρήγορες και εύχρηστες μεθόδους είναι ο αλγόριθμος Dijkstra.



Εικόνα 3.1: Παράδειγμα Προβλήματος Συντομότερης Διαδρομής

3.3 Διατύπωση Προβλήματος

Η παρούσα διπλωματική πραγματεύεται ένα πρόβλημα με το οποίο έχει έρθει αντιμέτωπος σχεδόν οποιοσδήποτε χρειάστηκε να μετακινηθεί με αστική συγκοινωνία από ένα σημείο της πόλης του Βόλου σε κάποιο άλλο για οποιονδήποτε σκοπό. Δηλαδή, ποιο λεωφορείο του Αστικού Κτελ Βόλου θα χρησιμοποιήσει κάποιος προκειμένου να πληρώσει το ελάχιστο συνολικό κόστος και να αφιχθεί στο σημείο προορισμού στην ώρα που επιθυμεί.

Όπως γίνεται αντιληπτό, αντικείμενο προς βελτιστοποίηση είναι η ελαχιστοποίηση του συνολικού κόστους δοθέντος του σημείου εκκίνησης (σημείο αναχώρησης-Origin) και προορισμού (σημείο άφιξης-Destination). Ο χρήστης επιλέγει τόσο το μέρος αναχώρησης όσο και του προορισμού, καθώς και την ώρα που επιθυμεί να αφιχθεί στο σημείο προορισμού. Όπως προαναφέρθηκε, για να πραγματοποιηθεί αυτή η μετακίνηση ο χρήστης θα χρησιμοποιήσει την αστική συγκοινωνία που διαθέτει η πόλη του Βόλου. Δεδομένου ότι τα Αστικά Κτελ Βόλου έχουν συγκεκριμένα δρομολόγια αναχώρησης και επειδή θέλουμε η λύση να είναι όσο το δυνατόν πιο κοντά στην πραγματικότητα, αυτά συγκεντρώθηκαν από τη βάση δεδομένων του Αστικού Κτελ.

Το συγκεκριμένο πρόβλημα θα δοκιμαστεί τόσο για θεωρητικά δεδομένα (theoretical data) όσο και για ρεαλιστικά δεδομένα (GTFS data). Τα θεωρητικά δεδομένα δεν είναι τα δεδομένα, τα οποία διαθέτουν τα Αστικά Κτελ Βόλου, αλλά κάποια υποθετικά. Αντίθετα, τα GTFS είναι πραγματικά δεδομένα. Στις δύο επόμενες υποενότητες γίνεται αναφορά στα GTFS δεδομένα.

3.3.1 Γενικά περί GTFS

Το GTFS, γνωστό και ως GTFS στατικό ή στατική διέλευση GTFS, για να διαφοροποιηθεί από την επέκταση GTFS σε πραγματικό χρόνο (GTFS-Realtime), ορίζει μία κοινή μορφή για τα χρονοδιαγράμματα μέσων μαζικής μεταφοράς και τις σχετικές γεωγραφικές τους πληροφορίες. Μια υπηρεσία συγκοινωνιών μπορεί να παράγει μια ροή GTFS για να μοιραστεί τις πληροφορίες της με προγραμματιστές, οι οποίοι αναπτύσσουν εργαλεία, που καταλώνουν ροές GTFS, για να ενσωματώσουν τις πληροφορίες αυτές στις εφαρμογές τους. Τα GTFS ξεκίνησαν ως ένα πλευρικό έργο (είναι κάτι που κάνει κάποιος εκτός από έναν κύριο στόχο του ή τη δουλειά του) του υπαλλήλου της Google Chris Harrelson το 2005, ο οποίος προσπάθησε με διάφορους τρόπους να ενσωματώσει δεδομένα μεταφοράς στους χάρτες της Google.

Στις Ηνωμένες Πολιτείες, δεν υπήρχε κανένα πρότυπο για τα χρονοδιαγράμματα των μέσων μαζικής μεταφοράς πριν από την εμφάνιση των GTFS. Οι, δημοσίως και ελευθέρως, διαθέσιμες προδιαγραφές μορφοποίησης (shapefile, csv, zip κτλ), καθώς και η διαθεσιμότητα των χρονοδιαγραμμάτων GTFS, γρήγορα έκαναν τους προγραμματιστές να βασίζουν το σχετικό με τη μεταφορά (με συγκοινωνία) λογισμικό τους σε αυτή τη μορφή (GTFS). Αυτό είχε σαν αποτέλεσμα, εκαντοντάδες χρήσιμες και δημοφιλείς εφαρμογές μεταφοράς με συγκοινωνία να είναι διαθέσιμες σε ροές GTFS. Λόγω της κοινής μορφής δεδομένων που τηρούν αυτές οι εφαρμογές, οι λύσεις δεν χρειάζεται να είναι προσαρμοσμένες στις ανάγκες ενός φορέα συγκοινωνιών, αλλά μπορούν εύκολα να επεκταθούν σε οποιαδήποτε περιοχή όπου υπάρχει διαθέσιμη ροή GTFS.

Τον Δεκέμβριο του 2005, το Portland έγινε η πρώτη πόλη που παρουσιάστηκε στην πρώτη έκδοση του προγράμματος της Google “Transit Trip Planner”. Τον Σεπτέμβριο του 2006 προστέθηκαν στο Google Transit Trip Planner πέντε ακόμη πόλεις των ΗΠΑ και η μορφή των δεδομένων κυκλοφόρησε ως Google Transit Feed Specification. Λόγω της ευρείας χρήσης της μορφής, το τμήμα “Google” του αρχικού ονόματος θεωρήθηκε ως μία εσφαλμένη ονομασία που έκανε ορισμένους δυνητικούς χρήστες να αποφεύγουν να υιοθετήσουν τον όρο GTFS. Αυτό έχει ως συνέπεια να αλλάξει όνομα η προδιαγραφή σε General Transit Feed Specification το 2009.

Μία ροή GTFS αποτελείται από μία σειρά αρχείων κειμένου .txt που συλλέγονται σε ένα αρχείο .zip. Κάθε αρχείο παρουσιάζει μία άλλη όψη των πληροφοριών μεταφοράς: στάσεις, διαδρομές, ταξίδια και άλλα δεδομένα χρονοδιαγράμματος. Τα GTFS βρίσκουν εφαρμογές στα Journey planning, Accescibility research, Comparing service levels. Το πρόβλημα που πραγματευόμαστε ανήκει στην κατηγορία του Journey planning. Στο Journey planning, τα GTFS συνήθως χρησιμοποιούνται για να παρέχουν δεδομένα μέσων μαζικής μεταφοράς για χρήση σε εφαρμογές προγραμματισμού πολυτροπικών ταξιδιών. Στις περισσότερες περιπτώσεις, τα GTFS συνδυάζονται με μία λεπτομερή απεικόνιση του δικτύου οδών/πεζοδρομίων ώστε να επιτρέπεται η δρομολόγηση από σημείο σε σημείο και όχι μόνο μεταξύ των στάσεων. Τα GTFS αρχικά σχεδιάστηκαν για χρήση στο Google Transit, μία διαδικτυακή εφαρμογή προγραμματισμού πολυτροπικών ταξιδιών, ενώ στις μέρες μας χρησιμοποιείται και σε άλλες εφαρμογές όπως το OpenTripPlanner ή ακόμη σε επέκταση του Arc Map Network Analyst.

Τέλος, ακολουθώντας τους οδηγούς δημιουργίας ενός GTFS πακέτου, μία εταιρία συγκοινωνιών μπορεί να δημοσιεύσει τα δεδομένα της και να δώσει τη δυνατότητα στους προγραμματιστές να συμπεριλάβουν τα δρομολόγιά της στις εφαρμογές τους. Παρακάτω ακολουθούν οι περιγραφές των πινάκων, που περιέχονται στο αρχείο .zip και που

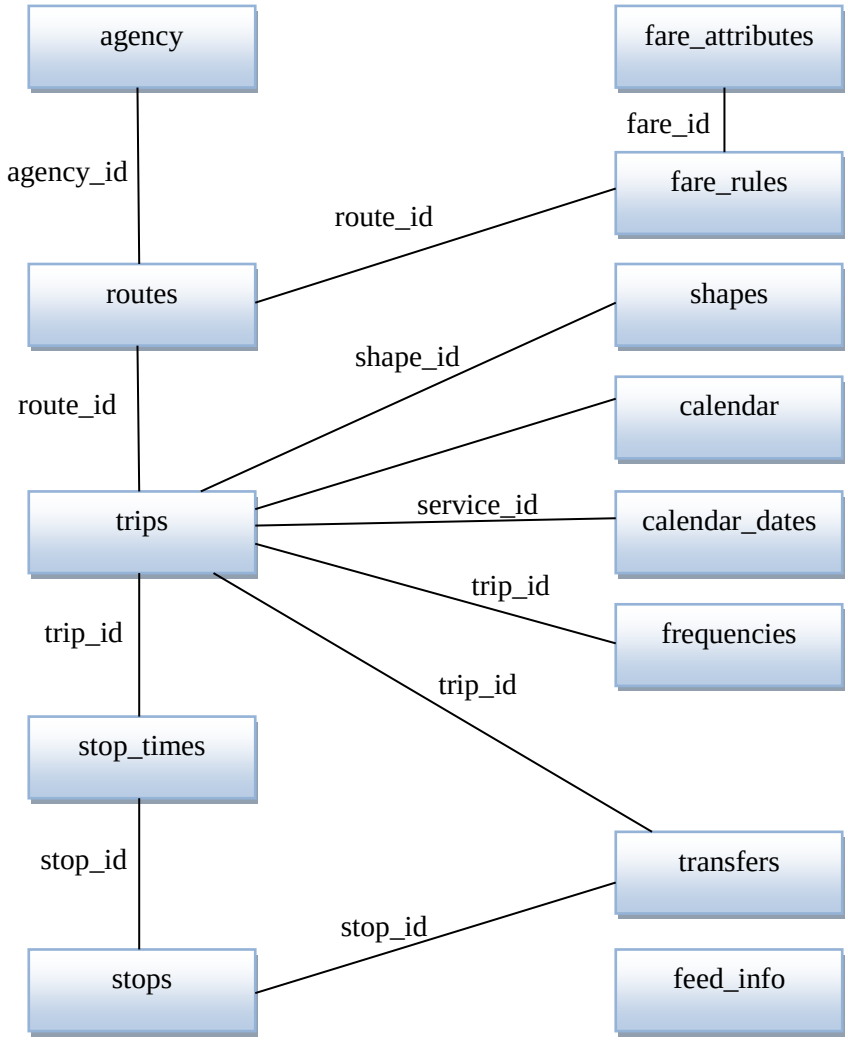
απαιτούνται για μια έγκυρη ροή GTFS δεδομένων. Κάθε πίνακας είναι κυριολεκτικά ένα αρχείο κειμένου CSV, του οποίου το όνομα είναι το όνομα του πίνακα που προστέθηκε με την ένδειξη .txt. Τα αρχεία είναι τα εξής:

- **Agency:** ο πίνακας πρακτορείων παρέχει πληροφορίες σχετικά με τον οργανισμό συγκοινωνιών συμπεριλαμβανομένου του ονόματος, της ιστοσελίδας και των πληροφοριών επικοινωνίας.
- **Routes:** ο πίνακας δρομολογίων προσδιορίζει ξεχωριστές διαδρομές. Μία διαδρομή είναι ένα σύνολο δρομολογίων που αποτελούν μία συγκεκριμένη, αναγωρίσιμη υπηρεσία και αυτό πρέπει να διακρίνεται από τα ξεχωριστά δρομολόγια, πολλά από τα οποία μπορεί να ανήκουν σε μία μόνο διαδρομή.
- **Trips:** ο πίνακας αυτός περιέχει όλα τα δρομολόγια που μπορεί να πραγματοποιήσει κάθε μέσο. Ένα δρομολόγιο είναι μία σειρά από δύο ή περισσότερες στάσεις σε συγκεκριμένο χρόνο.
- **Stop_times:** ο πίνακας αυτός περιλαμβάνει την χρονική στιγμή αναχώρησης, την χρονική στιγμή άφιξης, καθώς και αυτή που θα κάνει στάση το μέσο. Επίσης, περιέχεται και η συχνότητα των στάσεων.
- **Stops:** ο πίνακας σταματήματος ορίζει τις γεωγραφικές θέσεις κάθε πραγματικής στάσης καθώς και, προαιρετικά, μερικές από τις παροχές που σχετίζονται με αυτές τις στάσεις.
- **Calendar:** ο πίνακας ημερολογίου ορίζει τα πρότυπα εξυπηρέτησης που λειτουργούν επαναλαμβανόμενα όπως, για παράδειγμα, κάθε εβδομάδα.

Αναφέρθηκαν τα αρχεία που απαιτούνται για την δημιουργία ενός αρχείου GTFS. Ωστόσο υπάρχουν και κάποια προαιρετικά αρχεία που είναι στην διάθεση του χρήστη αν θέλει να τα δημιουργήσει και τα οποία είναι:

- **Fare_attribute:** παρέχει πληροφορίες σχετικά με τα ναύλα για τις διαδρομές ενός οργανισμού διέλευσης.
- **Fare_rules:** περιλαμβάνει κανόνες που διέπουν την πολιτική ναύλων για κάθε διαδρομή.
- **Shapes:** περιέχει κανόνες για τη σχεδίαση διαδρομών σε έναν χάρτη.
- **Calendar_dates:** περιλαμβάνει εξαιρέσεις λειτουργίας υπηρεσιών που συμπεριλαμβάνονται στο αρχείο calendar.txt. Αν αυτό το αρχείο περιλαμβάνει όλες τις ημερομηνίες υπηρεσίας, τότε μπορεί να οριστεί αντί για calendar.txt.
- **Frequencies:** ορίζει τον προσανατολισμό (χρόνος μεταξύ ταξιδιών) για διαδρομές με μεταβλητή συχνότητα υπηρεσίας.

- **Transfers:** περιέχει κανόνες για την πραγματοποίηση συνδέσεων μεταξύ διαδρομών
- **Feed.info:** περιλαμβάνει πληροφορίες σχετικά με την ίδια τη ροή, συμπεριλαμβανομένων των πληροφοριών εκδότη, έκδοσης και λήξης.



Εικόνα 3.2: Κλασσικό Διάγραμμα GTFS

3.3.2 Δημιουργία GTFS Δεδομένων

Είναι σημαντικό να αναφερθεί πως για να έχουμε αξιόπιστη επίλυση ενός προβλήματος δρομολόγησης, χρειάζεται μία πλήρης και απόλυτα έγκυρη βάση δεδομένων του δικτύου των Μέσων Μαζικής Μεταφοράς (στο πρόβλημα που μελετάμε είναι το Αστικό Κτελ Βόλου). Έτσι η διαδικασία από την οποία περνούν τα δεδομένα ανάγεται σε τρία στάδια :

1. **Εύρεση δεδομένων**

   Οι πληροφορίες που δημιουργούν τελικά το επιθυμητό αποτέλεσμα για κάθε πάροχο κάποιας υπηρεσίας ΜΜΜ είναι:

   - Η τοποθεσία των στάσεων ή το στίγμα των στάσεων. Εκφράζεται με το γεωγραφικό πλάτος και μήκος.
   - Οι διαδρομές των δρομολογίων, δηλαδή η ακριβής γραμμή πάνω στον δρόμο, την οποία διασχίζει κάθε φορά το λεωφορείο.
   - Τα ονόματα των στάσεων. Αν δεν υπάρχουν ή δεν μπορούμε να βρούμε κάποια από αυτά, τότε αρκεί να γνωρίζουμε την συμβολή των οδών/διασταύρωση που βρίσκεται πιο κοντά στην κάθε στάση.
   - Το πρόγραμμα των δρομολογίων ή με άλλα λόγια το πότε κάθε γραμμή περνάει από κάθε στάση. Αυτό πρέπει να γίνει τόσο για τα καλοκαιρινά όσο και για τα χειμερινά δρομολόγια.
   - Το όνομα του κάθε δρομολογίου και ο αριθμός του αν υπάρχει. Μας ενδιαφέρει το όνομα που αναγράφεται πάνω στο παρμπρίζ του λεωφορείου.
   - Μερικές βασικές πληροφορίες σχετικά με τον πάροχο όπως το όνομά του, η χώρα που ανήκει, η ιστοσελίδα του ακόμη και τηλέφωνα επικοινωνίας.

Οι παραπάνω πληροφορίες παρέχονται συνήθως μέσω KML αρχείων, των οποίων η χρήση είναι ευρέως διαδεδομένη στα ΜΜΜ.

2. **Μηχανογράφηση/Ψηφιοποίηση δεδομένων**

   Αφού συλλέξουμε όλα τα γεωχωρικά δεδομένα, πρέπει να τα φέρουμε στην κατάλληλη μορφή για να πραγματοποιηθεί σωστή εισαγωγή στη βάση δεδομένων. Αυτό επιτυγχάνεται με επεξεργασία των δεδομένων με το πρόγραμμα QGIS, το οποίο χρησιμοποιείται για την προβολή, την επεξεργασία και την ανάλυση γεωγραφικών δεδομένων, και με χρήση του OpenStreetMap, το οποίο είναι ένα συνεργατικό έργο για τη δημιουργία ενός δωρεάν επεξεργάσιμου χάρτη του κόσμου και αποτελεί βάση για την παράσταση των δεδομένων αυτών. Αφού γίνει η διαδικασία στο QGIS με χρήση του χάρτη του OSM προκύπτει το τελικό αρχείο, που είναι σε μορφή shapefile (.shp).

3. **Εισαγωγή στη βάση δεδομένων**

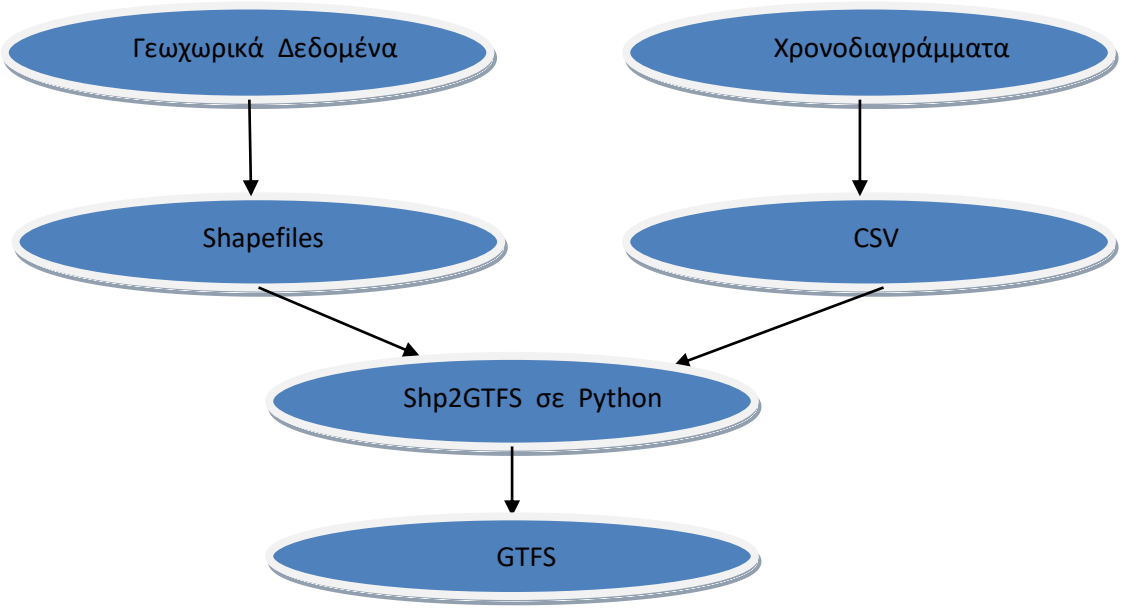
   Για την επιτυχή εισαγωγή των δεδομένων στη βάση χρειάζεται το λογισμικό PostgreSQL, στο οποίο αποθηκεύουμε όπως επίσης αναλύουμε και επεξεργαζόμαστε τα δεδομένα.

Τα λογισμικά QGIS, PostgreSQL, καθώς και οι χάρτες OpenStreetMap αποτελούν εργαλεία τα οποία όταν συνδυάζονται κάτω από μια κοινή βάση, τότε παράγεται το επιθυμητό αποτέλεσμα (GTFS). Η κοινή αυτή βάση είναι το πρόγραμμα shp2GTFS που ΄΄γράφεται΄΄ σε γλώσσα Python, η οποία αποτελεί βασικό μοχλό για την λειτουργία της όλης διαδικασίας.

Μετά την δημιουργία των shapefiles μέσω του QGIS, σειρά έχει αυτή των αρχείων CSV. Τα CSVείναι αρχεία τύπου .txt και κάθε γραμμή τους αντιπροσωπεύει ένα αρχείο δεδομένων που περιλαμβάνει πληροφορίες σχετικά με τις ώρες πραγματοποίησης των δρομολογίων αλλά και με τα μοτίβα που αυτές ακολουθούν. Όταν έχουμε τόσο τα shapefiles όσο και τα χρονοδιαγράμματα του ΜΜΜ, του οποίου τα GTFS θέλουμε να παράξουμε, προχωράμε σε αντιστοίχιση των δρομολογίων που παρουσιάζονται μέσα από τα ωρολόγια προγράμματα με τα shapefiles που έχουμε δημιουργήσει, δηλαδή το κάθε σύνολο από shapefiles που αναφέρεται στο ίδιο δρομολόγιο, πρέπει να έχει το δικό του αρχείο .txt.

Αφού τελειώσουμε και με τη δημιουργία των CSV αρχείων, προχωράμε στη δημιουργία του shp2GTFS. Το shp2GTFS είναι μία εφαρμογή βασισμένη σε Python που δημιουργήσαμε και μας επιτρέπει να μετασχηματίζουμε shapefiles σε ροές GTFS με λιγότερη προσπάθεια από οποιαδήποτε προτεινόμενη προσέγγιση που γνωρίζουμε. Το shp2GTFS είναι ένα εργαλείο που δημιουργήσαμε και βρίσκεται ακόμη σε φάση ανάπτυξης. Η χρήση του βασίζεται στα keyfiles. Τα keyfiles είναι αρχεία CSV που περιέχουν πληροφορίες σχετικά με το χρονοδιάγραμμα των ταξιδιών και άλλες λογικές που πρέπει να συμπεριληφθούν σε μια ροή GTFS. Στην πραγματικότητα, το shp2GTFS συνδέεται με τη βάση δεδομένων χρησιμοποιώντας μία μονάδα python, ανακτά τα δεδομένα που ζητάμε να ανακτήσει, και συνδυάζοντάς τα με τις πληροφορίες των keyfiles, παράγει τα αρχεία GTFS ένα προς ένα.

Η διαδικασία μετατροπής των πραγματικών δεδομένων στην επιθυμητή τους μορφή, που δεν είναι άλλη από τα GTFS, συνοψίζεται στο παρακάτω διάγραμμα:



Εικόνα 3.3: Γενικό Διάγραμμα Ροής για Παραγωγή GTFS

3.4 Μαθηματικά Μοντέλα

Το συγκεκριμένο πρόβλημα αποτελεί ένα πρόβλημα δρομολόγησης με αστική συγκοινωνία στο οποίο καλούμαστε να υπολογίσουμε τη συντομότερη διαδρομή ώστε να ελαχιστοποιείται το συνολικό κόστος. Επομένως, θεωρούμε έναν κατευθυνόμενο γράφο με βάρη, με τους κόμβους να αποτελούν τις στάσεις του Αστικού Κτελ Βόλου, τις ακμές τις δυνατές μεταβιβάσεις και τα βάρη να αποτελούν το κόστος μετακίνησης.

Προσεγγίζουμε δύο είδη προβλημάτων, τα οποία εφαρμόζονται και για θεωρητικά και για πραγματικά δεδομένα. Και τα δύο προβλήματα μοντελοποιούνται με πρόγραμμα ακέραιου προγραμματισμού όπου κάθε κόμβος αντιστοιχίζεται με μία στάση. Η διαφορά τους έγκειται στο γεγονός πως το πρώτο μοντέλο υπολογίζει αναλυτική λύση, ενώ το δεύτερο δίνει ευρετική λύση. Δηλαδή, στο δεύτερο πρόβλημα γίνεται χρήση ευρετικού κανόνα, ο οποίος χρησιμοποιείται για πιο γρήγορη επίλυση του αλγορίθμου. Ακριβώς στη συνέχεια διατυπώνουμε τα μαθηματικά μοντέλα.

3.4.1 Μοντέλο M1 με Αντιστοίχιση Κόμβου με Στάση

Σε αυτό το μοντέλο κάθε στάση της πόλης του Βόλου αντιστοιχεί σε έναν και μόνο κόμβο με το σύνολο των κόμβων να καθορίζουν το μέγεθος του δικτύου N. Ο πρώτος αριθμός 0

δηλώνει την αφετηρία ενώ ο τελευταίος (N - 1) την τερματική στάση. Πιο συγκεκριμένα, για την καλύτερη ανάλυση του μαθηματικού μοντέλου αναφερόμαστε στα επιμέρους χαρακτηριστικά στοιχεία του, δηλαδή στις παραμέτρους, στις μεταβλητές απόφασης, στην αντικειμενική συνάρτηση και στους περιορισμούς:

- *Δείκτες:*

  i, j, k : οι δείκτες όλων των κόμβων i, j, k = 0, 1, …, N,
  m, c : οι δείκτες των αστικών λεωφορείων m, c = 0, 1, ..., mmax ή cmax,
  n, z : οι δείκτες των δρομολογίων n, z = 0, 1, …, nmax ή zmax.

- *Σύνολα:*

  N : ο συνολικός αριθμός των κόμβων,
  mmax, cmax : το πλήθος των αστικών λεωφορείων,
  nmax, zmax : το πλήθος των δρομολογίων.

- *Παράμετροι:*

  O : ο κόμβος αναχώρησης (ορίζεται από τον χρήστη),
  D : ο κόμβος άφιξης (ορίζεται από τον χρήστη),
  ToA: η επιθυμητή ώρα άφιξης στον προορισμό D (ορίζεται από τον χρήστη),
  d_{ijmn}: η ώρα αναχώρησης από τον κόμβο i στον κόμβο j με το αστικό λεωφορείο m και το δρομολόγιο n,
  c_{ijm} : το κόστος μετακίνησης από τον κόμβο i στον κόμβο j με το αστικό λεωφορείο m,
  t_{ijm} : ο χρόνος μετακίνησης από τον κόμβο i στον κόμβο j με το αστικό λεωφορείο m,
  M : μεγάλος αριθμός.

Στις στάσεις της πόλης του Βόλου που δεν υπάρχει απευθείας σύνδεση με την αστική συγκοινωνία, το κόστος μετακίνησης και ο χρόνος μετακίνησης ορίζονται με έναν μεγάλο αριθμό που λαμβάνει την τιμή 500. Επίσης, ο πίνακας d_{ijmn} στους κόμβους που δεν υπάρχει σύνδεση παίρνει την τιμή 5000, επειδή είναι η μεγαλύτερη τιμή δρομολογίου και δεν αντιστοιχεί σε κάποιο χρόνο μέσα στη μέρα, οπότε δεν θα είναι δυνατόν να επιλεγεί. Θεωρούμε πως ο μεγάλος αριθμός M παίρνει την τιμή 1.000.000.

Τα δεδομένα (πίνακες δρομολόγησης d_{ijmn}, κόστος μετακίνησης c_{ijm}, χρόνος μετακίνησης t_{ijm}) είναι με την μορφή πινάκων και σε αρχεία CSV και από εκεί διαβάζονται στον αλγόριθμο.

- **<u>Μεταβλητές Απόφασης:</u>**

  X_{ijmn}: λαμβάνει την τιμή 1, αν επιλέγεται η διαδρομή i,j με το αστικό λεωφορείο m και το δρομολόγιο n, ενώ διαφορετικά την τιμή 0,

  DT_{ijmn} : η ώρα αναχώρησης από τον κόμβο i στον κόμβο j με το αστικό λεωφορείο m και το δρομολόγιο n,

  AT_{jimn} : η ώρα άφιξης από τον κόμβο j στον κόμβο i με το αστικό λεωφορείο m και το δρομολόγιο n,

  U_i: σειρά με την οποία ο κόμβος i έχει επισκεφθεί.

Οι μεταβλητές AT_{jimn}, DT_{ijmn} λαμβάνουν τιμές από 0 έως ένα άνω όριο, το οποίο εξαρτάται από τον τρόπο με τον οποίο δίνεται ο χρόνος στους πίνακες δρομολόγησης, ενώ η X_{ijmn} είναι δυαδική μεταβλητή. Στο πρόβλημα που μελετάμε, ο χρόνος δίνεται σε λεπτά μετά τα μεσάνυχτα. Παρακάτω αναγράφουμε την αντικειμενική συνάρτηση και τους περιορισμούς του προβλήματος:

ΑΝΤΙΚΕΙΜΕΝΙΚΗ ΣΥΝΑΡΤΗΣΗ

$$\text{Min } \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{ijmn} * C_{ijm}, \; i \neq j \qquad (1)$$

ΠΕΡΙΟΡΙΣΜΟΙ

$$\sum_{j=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{ijmn} \leq 1 \;, \forall i \neq j \qquad (2)$$

$$\sum_{j=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{Ojmn} = 1 \;, \forall j \neq O \qquad (3)$$

$$\sum_{i=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{iDmn} = 1 \;, \forall i \neq D \qquad (4)$$

$$\sum_{i=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{ijmn} - \sum_{k=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{jkmn} = 0 \quad , \qquad (5)$$

$$\forall j \neq D, i \neq D, k \neq O$$

$$\sum_{j=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{Djmn} = 0 \;, \forall j \neq D \qquad (6)$$

$$\sum_{i=0}^{N}\sum_{m=0}^{mmax}\sum_{n=0}^{nmax} X_{iOmn} = 0 \;, \forall i \neq O \qquad (7)$$

$$X_{ijmn} + X_{jimn} \leq 1 \;, \forall i, j, m, n, \; i \neq j \qquad (8)$$

$$U_i - U_j + N * X_{ijmn} \leq N - 1 \ , \forall i,j,m,n, i \neq j \quad (9)$$

$$d_{ijmn} - (1 - X_{ijmn}) * M \leq DT_{ijmn} \ , \forall i,j,m,n, i \neq j \quad (10)$$

$$DT_{ijmn} \leq d_{ijmn} + (1 - X_{ijmn}) * M \ , \forall i,j,m,n, i \neq j \quad (11)$$

$$-X_{ijmn} * M \leq DT_{ijmn} \leq X_{ijmn} * M \ , \forall i,j,m,n, i \neq j \quad (12)$$

$$DT_{ikmn} + t_{ikm} * X_{ikmn} - DT_{kjcz} \leq (2 - X_{ikmn} - X_{kjcz}) * M \quad , \quad (13)$$

$$\forall i,k,j,m,n,c,z, i \neq j, i \neq k, j \neq k$$

$$DT_{ijmn} + t_{ijm} * X_{ijmn} - (1 - X_{ijmn}) * M \leq AT_{jimn}, \forall i,j,m,n, i \neq j \quad (14)$$

$$AT_{jimn} \leq DT_{ijmn} + t_{ijm} * X_{ijmn} + (1 - X_{ijmn}) * M \ , \forall i,j,m,n, i \neq j \quad (15)$$

$$-X_{ijmn} * M \leq AT_{jimn} \leq X_{ijmn} * M \ , \forall i,j,m,n, i \neq j \quad (16)$$

$$\sum_{i=0}^{N} \sum_{m=0}^{mmax} \sum_{n=0}^{nmax} AT_{Dimn} \leq ToA \ , i \neq j \quad (17)$$

Η σχέση (1) αποτελεί την αντικειμενική συνάρτηση του προβλήματος που είναι η ελαχιστοποίηση του συνολικού κόστους. Ας διατυπώσουμε τώρα τους περιορισμούς που αναγράψαμε παραπάνω μαθηματικά. Η σχέση (2) ορίζει πως από κάθε κόμβο μπορούμε να αναχωρήσουμε το πολύ μία φορά προς οποιονδήποτε άλλο κόμβο με οποιοδήποτε αστικό λεωφορείο και με οποιοδήποτε δρομολόγιο. Η σχέση (3) καθορίζει πως πρέπει να εκκινήσουμε από τον κόμβο αναχώρησης (εμείς τον ορίζουμε) με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο ακριβώς μία φορά, ενώ η σχέση (4) να αφιχθούμε στον κόμβο προορισμού (εμείς τον ορίζουμε) με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο μία φορά ακριβώς.

Σύμφωνα με τον περιορισμό (5) καθορίζεται η συνέχεια του δικτύου, δηλαδή όταν ο χρήστης αφιχθεί στον κόμβο j θα πρέπει και να αναχωρήσει από αυτόν όταν δεν είναι κόμβος τελικού προορισμού. Η σχέση (6) ορίζει ότι δεν μπορεί να επιλεχθεί διαδρομή που να αναχωρεί από τον κόμβο τερματισμού με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο, ενώ η σχέση (7) που να καταλήγει στον κόμβο αφετηρίας με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο. Ο περιορισμός (8) δείχνει πως η δρομολόγηση πραγματοποιείται προς μόνον μία κατεύθυνση χωρίς επιστροφή, δηλαδή απαγορεύεται να επιστρέψουμε σε κάποιον κόμβο από τον οποίο αναχωρήσαμε νωρίτερα. Η σχέση (9) χρησιμοποιείται για την εξάλειψη των υποδιαδρομών που δημιουργούνται κατά την επίλυση του προβλήματος. Πιο συγκεκριμένα, πρέπει όλοι οι σταθμοί του δικτύου να συνδεθούν σε τέτοια σειρά ώστε ο μέγιστος αριθμός κόμβων που παρεμβάλλονται ανάμεσά τους να είναι N-1.

Οι σχέσεις (10), (11) και (12) εξασφαλίζουν πως όταν πραγματοποιείται η διαδρομή i,j με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο, τότε η ώρα αναχώρησης παίρνει συγκεκριμένη τιμή που δίνεται από τους πίνακες δρομολόγησης, ενώ διαφορετικά λαμβάνει την τιμή 0. Ο περιορισμός (13) επιβάλλει η αναχώρηση από τον κόμβο k για τον j να είναι χρονικά μετά την άφιξη του κόμβου k στον κόμβο i, δηλαδή αποτελεί μία σχέση μεταξύ των διαδοχικών σημείων αναχώρησης ikj με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο. Οι σχέσεις (14), (15) και (16) καθορίζουν ότι όταν πραγματοποιείται η διαδρομή i,j με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο, τότε η ώρα άφιξης λαμβάνει μία συγκριμένη τιμή που δίνεται από συνδυασμό των πινάκων δρομολόγησης και των χρόνων μετακίνησης, ενώ σε αντίθετη περίπτωση μηδενίζεται. Τέλος, ο περιορισμός (17) ορίζει πως η ώρα άφιξης του κόμβου προορισμού από έναν κόμβο i με οποιοδήποτε αστικό λεωφορείο και δρομολόγιο πρέπει να είναι μικρότερη ή ίση από την επιθυμητή ώρα άφιξης στον προορισμό.

3.4.2 Μοντέλο M2 με Αντιστοίχιση Κόμβου με Στάση με Χρήση Ευρετικού Κανόνα

Σε αυτή την ενότητα ασχολούμαστε με το δεύτερο πρόβλημα δρομολόγησης με αστική συγκοινωνία για την πόλη του Βόλου. Το μοντέλο M2 είναι ακριβώς ίδιο με το M1 όσον αφορά τόσο τους περιορισμούς όσο και την αντικειμενική συνάρτηση. Επίσης, οι δείκτες, παράμετροι, μεταβλητές απόφασης και τα σύνολα είναι ακριβώς τα ίδια. Δηλαδή, ότι αναφέρθηκε στην υποενότητα 3.4.1 ισχύει και σε αυτήν εδώ.

Το μοντέλο M2 δημιουργείται από το μοντέλο M1και με επιπλέον προσθήκη ενός ευρετικού κανόνα. Με τη χρήση αυτού του κανόνα βελτιώνεται ο υπολογιστικός χρόνος επίλυσης του προβλήματος χωρίς να υποβαθμίζεται σημαντικά η ποιότητα των λύσεων. Ανάλογα με το είδος του προβλήματος αλλά και με αυτό των δεδομένων δημιουργούμε τον ευρετικό κανόνα. Στο πρόβλημα που μελετάμε, ο ευρετικός κανόνας είναι διαφορετικός για τα θεωρητικά από ότι για τα GTFS δεδομένα.

Η λογική του ευρετικού κανόνα για τα θεωρητικά δεδομένα είναι η εξής: δημιουργούμε μία ακόμη μεταβλητή απόφασης $C1_{ijm}$, η οποία λαμβάνει τις τιμές του C_{ijm} (δηλαδή $C1_{ijm} = C_{ijm}$). Παρατηρώντας το εύρος των τιμών του C_{ijm} (τιμές του C_{ijm} από τους αντίστοιχους πίνακες c_{ijm}),θεωρούμε πως το $C1_{ijm}$ θα ανήκει σε ένα διάστημα (δηλαδή $C1_{ijm} >$ ή $<$ από έναν συγκεκριμένο αριθμό), μέσα στο οποίο όλα τα C_{ijm} θα λαμβάνουν μία συγκεκριμένη τιμή (δηλαδή C_{ijm} = αριθμητική τιμή). Το άνω ή κάτω όριο που τοποθετούμε στην $C1_{ijm}$ είναι τέτοιο που να δίνει λύση, η οποία δεν θα έχει ιδιαίτερη απόκλιση από τη

βέλτιστη και θα υπολογίζεται ταχύτερα. Ο ακριβής ευρετικός κανόνας αναγράφεται στην υποενότητα 4.3.1.

Ενώ για τα GTFS δεδομένα η λογική του ευρετικού κανόνα είναι η εξής: δημιουργούμε μία μεταβλητή απόφασης $t1_{ijm}$, η οποία λαμβάνει τις τιμές του t_{ijm} (δηλαδή $t1_{ijm} = t_{ijm}$). Παρατηρώντας το εύρος των τιμών του t_{ijm} (τιμές του t_{ijm} σύμφωνα με τους αντίστοιχους πίνακες t_{ijm}), υποθέτουμε πως για ένα διάστημα τιμών του $t1_{ijm}$ (δηλαδή για $t1_{ijm}$ > ή < από έναν συγκεκριμένο αριθμό), όλα τα t_{ijm} θα λαμβάνουν μία συγκεκριμένη τιμή (t_{ijm} = αριθμητική τιμή). Με αυτόν τον τρόπο εξασφαλίζουμε εφικτή λύση, η οποία μπορεί να είναι είτε η βέλτιστη είτε κοντά στη βέλτιστη και υπολογίζεται σε συντομότερο χρόνο. Ο ακριβής ευρετικός κανόνας αναγράφεται στην υποενότητα 4.3.2.

3.5 Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκε και αναλύθηκε το πρόβλημα της δρομολόγησης που πραγματεύεται η παρούσα διπλωματική, δηλαδή η εύρεση βέλτιστης διαδρομής για διάφορα σημεία της πόλης του Βόλου με αστική συγκοινωνία. Ακόμη έγινε και μία εκτενής αναφορά στα GTFS δεδομένα και τον ρόλο που επιτελούν, ενώ παρουσιάστηκαν εν συντομία οι όροι: δρομολόγηση και ακέραιος προγραμματισμός. Στη συνέχεια διατυπώθηκαντα δύο μοντέλα: το ένα δίνει τη βέλτιστη λύση, ενώ το άλλο δίνει είτε τη βέλτιστη είτε πλησιέστερα σε αυτή λύση σε βελτιωμένο υπολογιστικό χρόνο. Οι μοντελοποιήσεις βασίστηκαν σε ακέραιο προγραμματισμό και εφαρμόζονται και για θεωρητικά και για πραγματικά (GTFS) δεδομένα. Στο επόμενο κεφάλαιο θα εφαρμοστούν παραδείγματα και για τα δύο μοντέλα και τα αποτελέσματα αυτών θα χρησιμεύσουν στο να γίνουν οι απαραίτητες συγκρίσεις.

ΚΕΦΑΛΑΙΟ 4: ΕΠΙΛΥΣΗ ΚΑΙ ΣΥΓΚΡΙΣΗ ΜΟΝΤΕΛΩΝ

4.1 Εισαγωγή

Στο κεφάλαιο αυτό τα δύο μοντέλα, που παρουσιάστηκαν στο κεφάλαιο 3, θα επιλυθούν μέσω συγκεκριμένου λογισμικού και ακόμη θα παρουσιαστούν τα αποτελέσματα από κάποια παραδείγματα. Τα παραδείγματα θα γίνουν τόσο με θεωρητικά δεδομένα όσο και με δεδομένα GTFS και θα φανούν χρήσιμα στην καλύτερη κατανόηση των δύο μοντέλων, καθώς και στη σύγκριση αυτών.

4.2 Ανάπτυξη Κώδικα

Τα μοντέλα, που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, αναπτύχθηκαν σε γλώσσα προγραμματισμού C++ και το λογισμικό που χρησιμοποιήθηκε ήταν το Microsoft Visual Studio 2013. Για την επίλυσή τους χρησιμοποιήθηκε solver IBM ILOG CPLEX Optimization Studio Version 12.7. Η υλοποίηση των προγραμμάτων αυτών έλαβε χώρα στον σταθερό μου υπολογιστή. Τα χαρακτηριστικά του υπολογιστή μου είναι τα εξής:

- Επεξεργαστής: Intel(R) Pentium(R) CPU G850 @ 2.90GHz 2.90GHz
- Εγκατεστημένη μνήμη (RAM): 4,00 GB
- Τύπος συστήματος: Λειτουργικό σύστημα 64 bit
- Λογισμικό: Windows 7 Ultimate

Τα στοιχεία για τα Αστικά Κτελ Βόλου ελήφθησαν τόσο από την αντίστοιχη ιστοσελίδα όσο και από απευθείας επαφή με τους αρμόδιους. Διαθέτουμε για τις ανάγκες του προβλήματος τόσο θεωρητικά όσο και GTFS δεδομένα. Και τα δύο είδη δεδομένων είναι αρχεία τύπου .txt και περιέχονται σε μορφή πινάκων. Και στα δύο μοντέλα, που θα επιλυθούν στο κεφάλαιο αυτό, δεν χρειάζεται να περάσουμε ένα ένα τα δεδομένα (ώρα αναχώρησης, χρόνος μετακίνησης, κόστος μετακίνησης), αλλά τα εισάγουμε απευθείας μέσω κατάλληλων εντολών, οι οποίες διαβάζουν αυτά τα CSV αρχεία μέσα στο πρόγραμμα. Η διαδικασία που ακολουθείται προκειμένου να αποκτήσουν τα πραγματικά δεδομένα την τελική τους μορφή, που δεν είναι άλλη από τα GTFS, έχει αναλυθεί στο προηγούμενο κεφάλαιο.

4.3 Παραδείγματα Δρομολόγησης

Για την αξιολόγηση των μοντέλων που αναφέρθηκαν στο κεφάλαιο 3 (μοντέλο χωρίς ευρετικό κανόνα και μοντέλο με ευρετικό κανόνα) επιλέχθηκαν κάποια παραδείγματα δρομολόγησης, τα οποία επιλύθηκαν με το προαναφερθέν λογισμικό. Και τα δύο μοντέλα εφαρμόστηκαν τόσο για θεωρητικά (theoritical) όσο και για πραγματικά (GTFS) δεδομένα.

Για τα θεωρητικά δεδομένα ο συνολικός αριθμός στάσεων είναι 25.

Όσον αφορά τα πραγματικά δεδομένα, το Αστικό Κτελ Βόλου διαθέτει τα δρομολόγια στην αρχική τους μορφή (KML αρχεία). Η διαδικασία μετατροπής των δεδομένων αυτών σε GTFS αναφέρθηκε γενικά στις υποενότητες 3.3.1 και 3.3.2 και πραγματοποιήθηκε αναλυτικά με τη βοήθεια της ομάδας του GreenYourMove, στην οποία συμμετείχα. Αφού παραχθούν τα GTFS δεδομένα για την πόλη του Βόλου, παρατηρούμε πως ο συνολικός αριθμός στάσεων είναι 389. Στα μοντέλα ακέραιου προγραμματισμού, όμως, θέλουμε τα δεδομένα να είναι σε μορφή CSV αρχείων, δηλαδή πρέπει να εκφραστούν τα GTFS δεδομένα σε αυτή τη μορφή. Αυτό πραγματοποιείται μέσω ενός κώδικα που έχει γράφτεί σε γλώσσα python για τις ανάγκες του GreenYourMove. Επειδή όμως, ο συνολικός αριθμός των στάσεων είναι αρκετά μεγάλος, η μετατροπή τους θα δημιουργήσει πάρα πολλά αρχεία CSV με αποτέλεσμα να υπερφορτώνονται τα μοντέλα (M1 και M2) και να αργούν πολύ οι επιλύσεις αυτών. Για αυτόν το λόγο, παίρνουμε ενδεικτικά τις 20 πρώτες στάσεις και τα 11 πρώτα αστικά λεωφορεία και κάνουμε την μετατροπή αυτή. Δηλαδή πήραμε τους πίνακες δρομολόγησης (πίνακες d_{ijmn}, t_{ijm}, c_{ijm}) για τις πρώτες 20 στάσεις και τα πρώτα 11 αστικά λεωφορεία. Επομένως, ο συνολικός αριθμός στάσεων για τα GTFS δεδομένα είναι 20.

4.3.1 Παραδείγματα Δρομολόγησης για Θεωρητικά Δεδομένα

Σε αυτή την ενότητα θα αναφερθούμε σε παραδείγματα δρομολόγησης και για τα δύο μοντέλα με χρήση θεωρητικών δεδομένων. Ο συνολικός αριθμός κόμβων N είναι 25. Θεωρούμε πως ο επιθυμητός χρόνος άφιξης ToA είναι 960 (ο χρόνος στο πρόβλημα που μελετάμε είναι εκφρασμένος σε λεπτά μετά τα μεσάνυχτα, π.χ 960 λεπτά μετά τα μεσάνυχτα = 16.00 μ.μ.) για όλα τα παραδείγματα. Η μέγιστη τιμή που μπορεί να λάβει η μεταβλητή ToA είναι 1440 (24.00 π.μ). Για την επίλυση των παραδειγμάτων χρειάζεται να δώσουμε σημείο εκκίνησης και σημείο προορισμού κάθε φορά. Κάθε σημείο αντιστοιχεί σε μία στάση. Στόχος της επίλυσης των προβλημάτων είναι η εύρεση ελάχιστου κόστους (εκφρασμένο σε μονάδες) και χρόνου στον οποίο βρίσκεται αυτό (υπολογιστικός χρόνος).

Όπως έχουμε αναφέρει στην παρούσα εργασία η μοντελοποίηση του M2 είναι ίδια ακριβώς με αυτήν του M1 με την προσθήκη όμως ενός ευρετικού κανόνα. Ο ευρετικός κανόνας που χρησιμοποιείται στα προβλήματα με θεωρητικά δεδομένα είναι ο εξής:

Έστω $C1_{ijm} = C_{ijm}$, τότε αν $C1_{ijm} < 5$ ισχύει ότι $C_{ijm} = 2$.

Κάθε μέτρηση τόσο για το μοντέλο M1 όσο και για το M2 πραγματοποιείται 4 φορές, ώστε να διασφαλιστεί η αξιοπιστία των αποτελεσμάτων. Κάθε πρόγραμμα εκτυπώνει χρόνο υλοποίησης, τιμή αντικειμενικής συνάρτησης, μεταβλητές απόφασης υλοποίησης των διαδρομων i,j και μεταβλητές χρόνου αναχώρησης και χρόνου άφιξης για τις διαδρομές που πραγματοποιούνται. Εμάς μας ενδιαφέρουν κυρίως η τιμή της αντικειμενικής συνάρτησης και ο χρόνος υλοποίησης και γι' αυτόν το λόγο παραθέτουμε τα αποτελέσματα των δύο αυτών μεγεθών στην παρούσα εργασία.

Ακολουθούν τα παραδείγματα που επιλύθηκαν και με τα δύο μοντέλα:

- **ΠΑΡΑΔΕΙΓΜΑ 1**

  Θεωρούμε σημείο εκκίνησης O=1 και σημείο προορισμού D=9.

  <u>ΜΟΝΤΕΛΟ M1</u>

  Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

  Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

  Χρόνος υλοποίησης = 13,08 sec

  <u>ΜΟΝΤΕΛΟ M2</u>

  Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

  Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

  Χρόνος υλοποίησης = 12,63 sec

- **ΠΑΡΑΔΕΙΓΜΑ 2**

  Θεωρούμε σημείο εκκίνησης O=4 και σημείο προορισμού D=20.

  <u>ΜΟΝΤΕΛΟ M1</u>

  Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

  Τιμή αντικειμενικής συνάρτησης (κόστος) = 3 μονάδες

  Χρόνος υλοποίησης = 13,86 sec

  <u>ΜΟΝΤΕΛΟ M2</u>

  Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

  Τιμή αντικειμενικής συνάρτησης (κόστος) = 6 μονάδες

  Χρόνος υλοποίησης = 13,50 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 3**

Θεωρούμε σημείο εκκίνησης O=5 και σημείο προορισμού D=18.

ΜΟΝΤΕΛΟ Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

Χρόνος υλοποίησης = 14,04 sec

ΜΟΝΤΕΛΟ Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

Χρόνος υλοποίησης = 13,62 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 4**

Θεωρούμε σημείο εκκίνησης O=6 και σημείο προορισμού D=15.

ΜΟΝΤΕΛΟ Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 3 μονάδες

Χρόνος υλοποίησης = 13,25 sec

ΜΟΝΤΕΛΟ Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 6 μονάδες

Χρόνος υλοποίησης = 12,77 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 5**

Θεωρούμε σημείο εκκίνησης O=10 και σημείο προορισμού D=23.

ΜΟΝΤΕΛΟ Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 2 μονάδες

Χρόνος υλοποίησης = 13,75 sec

ΜΟΝΤΕΛΟ Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

Χρόνος υλοποίησης = 13,52 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 6**

Θεωρούμε σημείο εκκίνησης O=17 και σημείο προορισμού D=24.

ΜΟΝΤΕΛΟ Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

Χρόνος υλοποίησης = 13,71 sec

ΜΟΝΤΕΛΟ M2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 4 μονάδες

Χρόνος υλοποίησης = 12,60 sec

ΠΑΡΑΔΕΙΓΜΑ	ΜΟΝΤΕΛΟ	ToA (λεπτά μετά τα μεσάνυχτα)	ΧΡΟΝΟΣ ΥΛΟΠΟΙΗΣΗΣ (sec)	ΤΙΜΗ ΑΝΤΙΚΕΙΜΕΝΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ (μονάδες)
1	M1	960	13,08	4
2	M1	960	13,86	3
3	M1	960	14,04	4
4	M1	960	13,25	3
5	M1	960	13,75	2
6	M1	960	13,71	4

Πίνακας 4.1: Αποτελέσματα των 6 Παραδειγμάτων για το Μοντέλο M1 για Θεωρητικά Δεδομένα

ΠΑΡΑΔΕΙΓΜΑ	ΜΟΝΤΕΛΟ	ToA(λεπτά μετά τα μεσάνυχτα)	ΧΡΟΝΟΣ ΥΛΟΠΟΙΗΣΗΣ (sec)	ΤΙΜΗ ΑΝΤΙΚΕΙΜΕΝΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ (μονάδες)
1	M2	960	12,63	4
2	M2	960	13,50	6
3	M2	960	13,62	4
4	M2	960	12,77	6
5	M2	960	13,52	4
6	M2	960	12,60	4

Πίνακας 4.2: Αποτελέσματα των 6 Παραδειγμάτων για το Μοντέλο M2 για Θεωρητικά Δεδομένα

Παρατηρούμε πως με το μοντέλο M2 το πρόβλημα δρομολόγησης επιλύεται χωρίς να υπάρχει σημαντική απόκλιση στην τιμή της αντικειμενικής συνάρτησης σε σχέση με το μοντέλο M1. Δηλαδή, ο ευρετικός κανόνας δίνει μία εφικτή λύση, της οποίας η τιμή είναι είτε η βέλτιστη είτε κοντά στη βέλτιστη. Επίσης, από τους χρόνους αναχώρησης και άφιξης,

τους οποίους δεν αναγράφουμε στα αποτελέσματα, συμπεραίνουμε ότι όλοι οι περιορισμοί ικανοποιούνται. Τέλος, είναι προφανές πως αν αλλάξουμε την μεταβλητή ToA τότε μπορεί να έχουμε αλλαγές στην τιμή της αντικειμενικής συνάρτησης,στους χρόνους αναχώρησης και άφιξης, αλλά και στους χρόνους επίλυσης.

4.3.2 Παραδείγματα Δρομολόγησης για GTFS Δεδομένα

Στην παρούσα ενότητα θα αναφερθούμε σε παραδείγματα δρομολόγησης με χρήση GTFS δεδομένων και για τα δύο μαθηματικά μοντέλα. Ο συνολικός αριθμός κόμβων N είναι 20. Θεωρούμε πως ο επιθυμητός χρόνος άφιξης ToA είναι 720, ο οποίος είναι εκφρασμένος σε λεπτά μετά τα μεσάνυχτα και αντιστοιχεί στις 12.00 μ.μ, για όλα τα παραδείγματα. Η μέγιστη τιμή που μπορεί να πάρει η μεταβλητή ToA είναι 720 (12.00 μ.μ). Για την επίλυση των παραδειγμάτων χρειάζεται να δώσουμε σημείο εκκίνησης και σημείο προορισμού. Κάθε σημείο αντιστοιχεί σε μία στάση. Στόχος της επίλυσης των προβλημάτων είναι η εύρεση ελάχιστου κόστους (εκφρασμένο σε μονάδες) και χρόνου στον οποίο βρίσκεται αυτό (υπολογιστικός χρόνος).

Ο ευρετικός κανόνας που χρησιμοποιείται στα προβλήματα με GTFS δεδομένα είναι ο εξής:

Έστω $t1_{ijm} = t_{ijm}$, τότε αν $t1_{ijm} < 500$ ισχύει ότι $t_{ijm} = 0$.

Κάθε μέτρηση τόσο για το μοντέλο M1 όσο και για το M2 πραγματοποιείται 4 φορές, ώστε να εξασφαλιστεί η αξιοπιστία των αποτελεσμάτων. Κάθε πρόγραμμα εκτυπώνει χρόνο επίλυσης, τιμή αντικειμενικής συνάρτησης, μεταβλητές απόφασης υλοποίησης των διαδρομων i,j και μεταβλητές χρόνου αναχώρησης και χρόνου άφιξης για τις διαδρομές που πραγματοποιούνται. Εμάς μας ενδιαφέρουν κυρίως η τιμή της αντικειμενικής συνάρτησης και ο χρόνος υλοποίησης και γι' αυτόν το λόγο παραθέτουμε μόνο τα αποτελέσματα των δύο αυτών μεγεθών στην παρούσα εργασία.

Εάν προσπαθήσουμε να επιλύσουμε κάποιο παράδειγμα στο πρόγραμμα, παρατηρούμε ό,τι δεν δίνεται λύση και εμφανίζεται στην οθόνη το μήνυμα: Not enough memory. Αυτό σημαίνει πως δεν υπάρχει αρκετή μνήμη RAM στον υπολογιστή ώστε να μπορεί το πρόγραμμα να διαβάσει όλα τα δεδομένα και κατ' επέκταση να επιλυθεί. Πιο συγκεκριμένα, το πρόβλημα παρατηρείται στον περιορισμό (13), ο οποίος πρέπει να επαναληφθεί πάρα πολλές φορές για τα δεδομένα των ωρών αναχώρησης d_{ijmn}, τα οποία είναι και αυτά πολλά, με αποτέλεσμα να μην επαρκεί η μνήμη RAM του υπολογιστή. Όμως διαπιστώνουμε πως δεν μας είναι χρήσιμες για βελτιστοποίηση οι μεταβιβάσεις που αντιστοιχούν σε d_{ijmn} με τιμή 5000, επειδή δεν είναι δυνατή η σύνδεση μεταξύ αυτών των στάσεων i και j, ενώ και τα

κόστη είναι πολύ μεγάλα σε αυτές τις μεταβιβάσεις και δεν συμμετέχουν στη βελτιστοποίηση της τιμής της αντικειμενικής συνάρτησης. Οπότε, ένας τρόπος να αντιμετωπιστεί το πρόβλημα είναι να παραβλέπει την τιμή 5000 σε κάποιους περιορισμούς ώστε να λαμβάνουμε τη σωστή βέλτιστη λύση. Για το λόγο αυτόν στους ακόλουθους περιορισμούς κάνουμε τα εξής:

Για περιορισμούς (2), (9), (12), (16): αν $d_{ijmn} \neq 5000$, τότε θα ισχύουν οι μαθηματικές εκφράσεις των περιορισμών αυτών.

Για περιορισμό (3): αν $d_{Ojmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (4): αν $d_{iDmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (6): αν $d_{Djmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (7): αν $d_{iOmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (3): αν $d_{Ojmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (8): αν $d_{jimn} \neq 5000$ και $d_{ijmn} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Για περιορισμό (13): αν $d_{ikmn} \neq 5000$ και $d_{kjcz} \neq 5000$, τότε θα ισχύει η μαθηματική έκφραση του περιορισμού αυτού.

Επίσης, από τα δεδομένα καταλήγουμε στο συμπέρασμα πως εφικτή λύση δίνεται μόνο αν το σημείο εκκίνησης O έχει μεγαλύτερη τιμή από το σημείο προορισμού π.χ αν O=4 και D=12 δεν υπάρχει εφικτή λύση, ενώ αν O=12 και D=4 προκύπτει βέλτιστη λύση.

Ακολουθούν τα παραδείγματα που επιλύθηκαν και με τα δύο μοντέλα:

- **ΠΑΡΑΔΕΙΓΜΑ 1**

  Θεωρούμε σημείο εκκίνησης O=11 και σημείο προορισμού D=2.

  <u>ΜΟΝΤΕΛΟ Μ1</u>

  Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

  Τιμή αντικειμενικής συνάρτησης (κόστος) = 15 μονάδες

  Χρόνος υλοποίησης = 6,32 sec

ΜΟΝΤΕΛΟ_Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 15 μονάδες

Χρόνος υλοποίησης = 5,87 sec

- **ΠΑΡΑΔΕΙΓΜΑ 2**

Θεωρούμε σημείο εκκίνησης O=9 και σημείο προορισμού D=1.

ΜΟΝΤΕΛΟ_Μ1

Η επίλυση του προβλήματος δίνειτα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 14 μονάδες

Χρόνος υλοποίησης = 6,03 sec

ΜΟΝΤΕΛΟ_Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 14 μονάδες

Χρόνος υλοποίησης = 5,89 sec

- **ΠΑΡΑΔΕΙΓΜΑ 3**

Θεωρούμε σημείο εκκίνησης O=19 και σημείο προορισμού D=12.

ΜΟΝΤΕΛΟ_Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 12 μονάδες

Χρόνος υλοποίησης = 3,78 sec

ΜΟΝΤΕΛΟ_Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 12 μονάδες

Χρόνος υλοποίησης = 3,15 sec

- **ΠΑΡΑΔΕΙΓΜΑ 4**

Θεωρούμε σημείο εκκίνησης O=18 και σημείο προορισμού D=5.

ΜΟΝΤΕΛΟ_Μ1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 21 μονάδες

Χρόνος υλοποίησης = 4,75 sec

ΜΟΝΤΕΛΟ_Μ2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 21 μονάδες

Χρόνος υλοποίησης = 4,16 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 5**

Θεωρούμε σημείο εκκίνησης O=10 και σημείο προορισμού D=4.

ΜΟΝΤΕΛΟ M1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 12 μονάδες

Χρόνος υλοποίησης = 4,36 sec

ΜΟΝΤΕΛΟ M2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 12 μονάδες

Χρόνος υλοποίησης = 3,66 sec

❖ **ΠΑΡΑΔΕΙΓΜΑ 6**

Θεωρούμε σημείοεκκίνησης O=15 και σημείο προορισμού D=6.

ΜΟΝΤΕΛΟ M1

Η επίλυση του προβλήματος δίνει τα εξής αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 15 μονάδες

Χρόνος υλοποίησης = 5,62 sec

ΜΟΝΤΕΛΟ M2

Η επίλυση του προβλήματος δίνει τα ακόλουθα αποτελέσματα:

Τιμή αντικειμενικής συνάρτησης (κόστος) = 15 μονάδες

Χρόνος υλοποίησης = 5,45 sec

ΠΑΡΑΔΕΙΓΜΑ	ΜΟΝΤΕΛΟ	ToA(λεπτά μετά τα μεσάνυχτα)	ΧΡΟΝΟΣ ΥΛΟΠΟΙΗΣΗΣ (sec)	ΤΙΜΗ ΑΝΤΙΚΕΙΜΕΝΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ (μονάδες)
1	M1	720	6,32	15
2	M1	720	6,03	14
3	M1	720	3,78	12
4	M1	720	4,75	21
5	M1	720	4,36	12
6	M1	720	5,62	15

Πίνακας 4.3: Αποτελέσματα των 6 Παραδειγμάτων για το Μοντέλο M1 για GTFS Δεδομένα

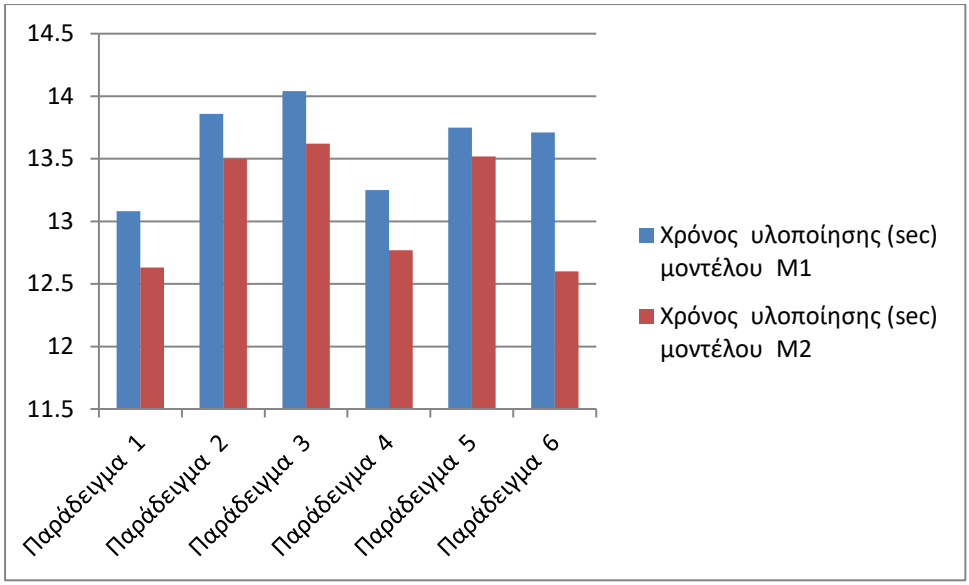
ΠΑΡΑΔΕΙΓΜΑ	ΜΟΝΤΕΛΟ	ToA (λεπτά μετά τα μεσάνυχτα)	ΧΡΟΝΟΣ ΥΛΟΠΟΙΗΣΗΣ (sec)	ΤΙΜΗ ΑΝΤΙΚΕΙΜΕΝΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ (μονάδες)
1	M2	720	5,87	15
2	M2	720	5,89	14
3	M2	720	3,15	12
4	M2	720	4,16	21
5	M2	720	3,66	12
6	M2	720	5,45	15

Πίνακας 4.4: Αποτελέσματατων 6 Παραδειγμάτων για το Μοντέλο M2 για GTFS Δεδομένα

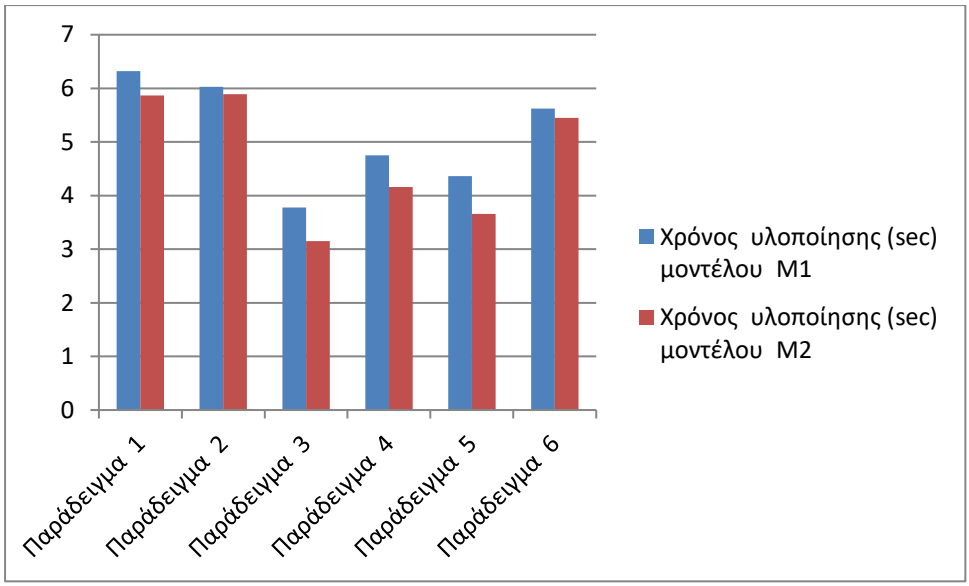
Παρατηρούμε πως το μοντέλο M2 δίνει την ίδια τιμή με το μοντέλο M1 για τα παραδείγματα που επιλύσαμε, η οποία είναι προφανώς και βέλτιστη. Δηλαδή η τιμή της αντικειμενικής συνάρτησης δεν αλλάζει με την προσθήκη του ευρετικού κανόνα. Ωστόσο, σε κάποια άλλα παραδείγματα μπορεί η λύση να διαφέρει για τα δύο μοντέλα. Ακόμη, αν μεταβάλλουμε την μεταβλητή ToA προκύπτουν πιθανότερα διαφορετικές λύσεις για τα δύο μοντέλα.

4.4 Σύγκριση Μοντέλων

Σε αυτή την ενότητα θα συγκρίνουμε τα δύο μοντέλα ως προς τον χρόνο επίλυσής τους με τη βοήθεια σχεδιαγραμμάτων. Τα σχεδιαγράμματα περιέχουν στον οριζόντιο άξονα τον αριθμό του παραδείγματος και στον κάθετο τον χρόνο επίλυσής τους. Τα δύο διαγράμματα που παρουσιάζονται παρακάτω έχουν συγκεκριμένο χρόνο επιθυμητής άφιξης ToA, ο οποίος καθορίζεται από τον χρήστη. Για τα θεωρητικά δεδομένα ToA=960 και για τα GTFS ToA=720. Προφανώς, για διαφορετικά ToA προκύπτουν συνήθως διαφορετικά διαγράμματα.



Διάγραμμα 4.1: Χρόνοι Επίλυσης για Θεωρητικά Δεδομένα των δύο Μοντέλων



Διάγραμμα 4.2: Χρόνοι Επίλυσης για GTFS Δεδομένα των δύο Μοντέλων

Και από τα δύο διαγράμματα παρατηρούμε ότι το μοντέλο M2 επιλύει τα παραδείγματα σε ταχύτερο υπολογιστικό χρόνο από το μοντέλο M1. Όπως διαπιστώσαμε στην ενότητα 4.3, η τιμή της αντικειμενικής συνάρτησης όχι μόνο δεν αποκλίνει ιδιαίτερα μεταξύ των δύο μοντέλων, αλλά και είναι ίδια σε αρκετά παραδείγματα. Άρα προκύπτει πως το μοντέλο M2 είναι ένα βελτιωμένο μοντέλο του αρχικού μοντέλου M1. Τα δύο προγράμματα διαφέρουν

συνήθως και ως προς τους χρόνους αναχώρησης και άφιξης. Ενώ πρέπει να τονιστεί ότι το μοντέλο M1 είναι ρεαλιστικό, σε αντίθεση με το M2 που είναι προσεγγιστικό.

4.5 Σύνοψη

Σε αυτό το κεφάλαιο εφαρμόσαμε κάποια παραδείγματα τόσο για τα δύο μοντέλα (M1και M2) όσο και για τους δύο τύπους δεδομένων (θεωρητικά και GTFS). Τα αποτελέσματα αυτών των παραδειγμάτων μας βοήθησαν στο να γίνει σύγκριση μεταξύ των μοντέλων. Στο επόμενο κεφάλαιο θα αναφερθούμε στο πρόγραμμα OTP, στο οποίο θα πραγματοποιήσουμε κάποια παραδείγματα για την πόλη του Βόλου (όπως κάναμε και με τα μοντέλα ακέραιου προγραμματισμού), ενώ θα γίνει και σύγκριση μεταξύ των προγραμμάτων OTP και C++.

ΚΕΦΑΛΑΙΟ 5: ΠΡΟΓΡΑΜΜΑ OTP ΚΑΙ ΣΥΓΚΡΙΣΗ ΜΕ ΠΡΟΓΡΑΜΜΑ C++

5.1 Εισαγωγή

Στο παρόν κεφάλαιο θα παρουσιάσουμε κάποιες βασικές πληροφορίες σχετικά με το πρόγραμμα OTP, ενώ θα γίνει και σύντομη αναφορά στο OpenStreetMap. Στη συνέχεια θα δημιουργήσουμε στο OTP το πρόβλημα που επιλύσαμε στο κεφάλαιο 3 για την πόλη του Βόλου και θα τρέξουμε κάποια παραδείγματα. Τέλος, θα γίνει σύγκριση μεταξύ των προγραμμάτων C++ και OTP.

5.2 Γενικά περί OTP

Το OpenTripPlanner είναι μία πλατφόρμα ανοιχτού κώδικα για τον σχεδιασμό πολυτροπικών ταξιδιών και την ανάλυση δικτύου μεταφορών, η οποία εκτελείται σε Linux, Mac, Windows ή σε οποιαδήποτε πλατφόρμα με εικονική μηχανή Java. Αποτελεί μία συνεργατική προσπάθεια μεταξύ του TriMet (του οργανισμού δημοσίων μεταφορών που εξυπηρετεί το Portland, OR, ΗΠΑ), του OpenPlans και των προγραμματιστών FivePoints, OneBusAway, Graphserver, καθώς και αρκετών ανεξάρτητων προγραμματιστών. Παρέχει τόσο μια διεπαφή δικτύου όσο και μια διεπαφή προγραμματισμού εφαρμογών (API) για εξωτερικές εφαρμογές, επιτρέποντας στους χρήστες να αναζητούν για δρομολόγια που συμπεριλαμβάνουν τα μέρη: πεζός, ποδήλατο, μέσο μαζικής μεταφοράς και αυτοκίνητο.

Το OpenTripPlanner είναι πλήρως πολυτροπικό όσον αφορά την πρόσβαση σε μέσο μαζικής μεταφοράς και οι μεταφορές μπορούν να πραγματοποιηθούν με τα πόδια ή με το ποδήλατο για δρόμους μονής κατεύθυνσης, δρόμους ποδηλάτων και λοφώδη εδάφη. Βασίζεται σε ανοιχτά πρότυπα δεδομένων, συμπεριλαμβανομένων των GTFS για τα μέσα μαζικής μεταφοράς και του OpenStreetMap για τα οδικά δίκτυα. Επιτρέπει διαγραμμίσεις (μεταφορές) μεταξύ διαφορετικών συστημάτων μέσων μαζικής μεταφοράς, καθώς επίσης και στους χρήστες να σχεδιάσουν ένα ταξίδι που μπορεί να συνδυάσει πολλαπλούς τρόπους μεταφοράς, όπως ποδηλασία ή περπάτημα για να φτάσει κάποιος στα μέσα μαζικής μεταφοράς. Επιπροσθέτως, το OTP δίνει τη δυνατότητα στον χρήστη να μπορεί να καθορίσει τις ώρες αναχώρησης ή άφιξης για τα μέσα μαζικής μεταφοράς, ενώ διαθέτει και κάποια προαιρετικά κριτήρια δρομολόγησης, όπως γρηγορότερο ταξίδι, λιγότερες μετεπιβιβάσεις, πρόσβαση σε αναπηρικά καροτσάκια, μέγιστη απόσταση με τα πόδια από το μέσο μαζικής

μεταφοράς. Μία ακόμη δυνατότητα του προγράμματος είναι να μπορεί ο χρήστης να ορίζει την αφετηρία και τον προορισμό.

Ο πυρήνας του OpenTripPlanner είναι ένας αλγόριθμος δρομολόγησης που χρησιμοποιεί ιεραρχίες συρρίκνωσης (είναι μία μέθοδος που επιταχύνει τη δρομολόγηση της συντομότερης διαδρομής, δημιουργώντας πρώτα προϋπολογισμένες "συμβατικές" εκδόσεις του γραφήματος σύνδεσης) για το οδικό δίκτυο και δρομολόγηση σε ακριβή χρόνο για το συγκοινωνιακό δίκτυο, και αποτελείται από τρία βασικά στοιχεία λογισμικού: έναν κατασκευαστή γραφημάτων, έναν μηχανισμό δρομολόγησης και ένα περιβάλλον χρήστη. Ενώ το πρωτεύον περιβάλλον χρήστη (δηλαδή ο χάρτης και η οπτική αναπαράσταση της διαδρομής) γράφεται σε JavaScript, τα άλλα συστατικά γράφονται στη γλώσσα προγραμματισμού Java στην πλατφόρμα Java Enterprise Edition (JavaEE). Το OTP κυκλοφορεί υπό την άδεια LGPL (Lesser General Public License).

Ξεκινώντας το 2009, το έργο έχει προσελκύσει μια ευημερούσα κοινότητα χρηστών και προγραμματιστών και έχει λάβει επενδύσεις από δημόσιους οργανισμούς, νεοσύστατες επιχειρήσεις και συμβούλους μεταφορών.

Στην παρακάτω υποενότητα θα αναφερθούμε στο πρόγραμμα OpenStreetMap, το οποίο διαδραματίζει ρόλο τόσο στα GTFS δεδομένα όσο και στο πρόγραμμα OpenTripPlanner.

5.2.1 Πληροφορίες για OpenStreetMap

Το OpenStreetMap είναι ένα συνεργατικό έργο για τη δημιουργία ενός δωρεάν επεξεργάσιμου χάρτη του κόσμου. Η δημιουργία και η ανάπτυξη του OSM προκλήθηκε από τους περιορισμούς στη χρήση ή τη διαθεσιμότητα πληροφοριών χάρτη σε ολόκληρο τον κόσμο και την εμφάνιση φθηνών φορητών συσκευών δορυφορικής πλοήγησης. Το OSM είναι μια παγκόσμια βάση δεδομένων που ανταγωνίζεται ή ξεπερνά την ποιότητα των εμπορικών χαρτών σε πολλές τοποθεσίες και θεωρείται σημαντικό παράδειγμα εθελοντικής γεωγραφικής πληροφορίας.

Δημιουργήθηκε από τον Steve Coast στο Ηνωμένο Βασίλειο το 2004, ο οποίος εμπνεύστηκε από την επιτυχία της Wikipedia και την κυριαρχία των ιδιόκτητων δεδομένων χάρτη στο Ηνωμένο Βασίλειο και αλλού. Έκτοτε έχει αυξηθεί σε πάνω από 2 εκατομμύρια εγγεγραμμένους χρήστες, οι οποίοι μπορούν να συλλέγουν δεδομένα χρησιμοποιώντας χειρωνακτική έρευνα, συσκευές GPS, αεροφωτογράφηση και άλλες ελεύθερες πηγές. Αυτά τα δεδομένα, που είναι αποθηκευμένα σε πλήθος δεδομένων, διατίθενται υπό την άδεια ODL (Open Database Licence).

Τα δεδομένα χαρτών συλλέγονται από την αρχή από εθελοντές που πραγματοποιούν συστηματικές έρευνες εδάφους χρησιμοποιώντας εργαλεία όπως μια συσκευή χειρός GPS, ένα σημειωματάριο, μια φωτογραφική μηχανή ή μια συσκευή εγγραφής φωνής. Στη συνέχεια, τα δεδομένα εισάγονται στη βάση δεδομένων OpenStreetMap. Στη χαρτογράφηση μιας συγκεκριμένης περιοχής συνεισφέρουν και οι διοργανώσεις του Mapathon που διεξάγονται από την ομάδα του OSM και από μη κερδοσκοπικούς οργανισμούς και τοπικές κυβερνήσεις.

Το OSM χρησιμοποιεί μια δομή τοπολογικών δεδομένων με τέσσερα βασικά στοιχεία (επίσης γνωστά ως πρωτόγονα δεδομένα):

- Οι **κόμβοι (nodes)** είναι σημεία με γεωγραφική θέση, αποθηκευμένα ως συντεταγμένες (ζευγάρια γεωγραφικού πλάτους και γεωγραφικού μήκους) σύμφωνα με το WGS 84 (Παγκόσμιο Γεωδαιτικό Σύστημα, το οποίο περιλαμβάνει ένα τυποποιημένο σύστημα συντεταγμένων για τη Γη).
- Οι **τρόποι (ways)** είναι ταξινομημένοι κατάλογοι κόμβων, που αντιπροσωπεύουν μία πολυγραμμή ή πιθανόν ένα πολύγωνο αν σχηματίζουν έναν κλειστό βρόχο. Χρησιμοποιούνται τόσο για την απεικόνιση γραμμικών χαρακτηριστικών όπως δρόμους και ποτάμια, όσο και για περιοχές όπως δάση, πάρκα, χώρους στάθμευσης και λίμνες.
- Οι **σχέσεις (relations)** είναι ταξινομημένες λίστες κόμβων, τρόπων και σχέσεων (που ονομάζονται μαζί μέλη), όπου κάθε μέλος μπορεί προαιρετικά να έχει "ρόλο". Οι σχέσεις χρησιμοποιούνται για την αντιπροσώπευση της σχέσης των υφιστάμενων κόμβων και τρόπων.
- Οι **ετικέτες (tags)** είναι ζεύγη κλειδιών-τιμών. Χρησιμοποιούνται για την αποθήκευση μεταδεδομένων σχετικά με τα αντικείμενα του χάρτη (όπως τον τύπο τους, το όνομα τους και τις φυσικές ιδιότητες). Οι ετικέτες δεν είναι ανεξάρτητες, αλλά είναι πάντα προσαρτημένες σε ένα αντικείμενο: σε έναν κόμβο, έναν τρόπο ή μία σχέση.

Τα αρχέτυπα δεδομένων OSM αποθηκεύονται και επεξεργάζονται σε διαφορετικές μορφές. Το κύριο αντίγραφο των δεδομένων αυτών αποθηκεύεται στην κύρια βάση δεδομένων του OSM. Η κύρια βάση δεδομένων είναι μία βάση δεδομένων PostgreSQL, στην οποία συμβαίνουν όλες οι επεξεργασίες και από την οποία δημιουργούνται όλες οι άλλες μορφές. Για τη μεταφορά των δεδομένων δημιουργούνται διάφορες χωματερές δεδομένων, οι οποίες είναι διαθέσιμες για λήψη. Η πλήρης "χωματερή" ονομάζεται planet.osm. Αυτές οι "χωματερές" υπάρχουν σε δύο μορφές, μία χρησιμοποιώντας XML και μία χρησιμοποιώντας

τη ρυθμιστική δυαδική μορφή (PBF) πρωτοκόλλου. Τέλος, διάφορες δημοφιλείς υπηρεσίες ενσωματώνουν κάποιο είδος γεωγραφικού εντοπισμού βάσει χάρτη.

5.3 Δημιουργία OTP

Σε αυτή την ενότητα θα γίνει αναφορά πως να δημιουργήσουμε γενικά ένα παράδειγμα OTP. Η δημιουργία ενός OTP παραδείγματος συνεπάγεται δημιουργία ενός γραφήματος για την περιοχή που μας ενδιαφέρει, το οποίο διαθέτει διάφορες επιλογές για εξαγωγή αποτελεσμάτων για οποιαδήποτε διαδρομή θέλουμε να πραγματοποιήσουμε σε αυτή την περιοχή. Με αφορμή τη γενική αναφορά για δημιουργία γραφήματος στο OTP, θα παρουσιαστεί η δημιουργία γραφήματος για την πόλη του Βόλου.

5.3.1 Βασική Χρήση του OTP

Το OpenTripPlannereίναι γραμμένο σε Java και διανέμεται ως ένα ενιαίο αρχείο JAR που μπορεί να τρέξει. Αυτά τα JAR αναπτύσσονται στο αποθετήριο Maven Central. Μεταβαίνουμε στον φάκελο OTP στο Maven Central. Στη συνέχεια μεταβαίνουμε στον φάκελο με τον υψηλότερο αριθμό έκδοσης και κάνουμε λήψη του αρχείου του οποίου το όνομα λήγει με το .shaded.jar.

Αφού κατεβάσουμε το αρχείο που καταλήγει σε shaded.jar, θα χρειαστούμε κάποια δεδομένα για την κατασκευή του OTP παραδείγματος. Πρώτα θα χρειαστούμε δεδομένα GTFS για να δημιουργήσουμε ένα δίκτυο μεταφοράς. Οι υπηρεσίες συγκοινωνιών σε όλο τον κόσμο παρέχουν ροές GTFS στο κοινό.Συνήθως, θέλουμε να μεταφέρουμε τα δεδομένα απευθείας από τους φορείς ή οργανισμούς συγκοινωνιών για να βεβαιωθούμε ότι έχουμε την πιο ενημερωμένη έκδοση. Αν γνωρίζουμε μια ροή με την οποία θέλουμε να εργαστούμε, την κατεβάζουμε και την τοποθετούμε σε έναν κενό φάκελο που έχουμε δημιουργήσει για παράδειγμα otp όπως π.χ το C: / Users / username / otp στα Windows. Το όνομα του αρχείου πρέπει να καταλήγει σε .zip για να το εντοπίσει το otp. Αν δεν έχουμε υπόψη μας κάποια συγκεκριμένη ροή, τότε το πρακτορείο Trimet του Oregon είναι μια καλή επιλογή.

Επίσης, θα χρειαστούμε OSM δεδομένα για να δημιουργήσουμε ένα οδικό δίκτυο για περπάτημα, ποδηλασία και οδήγηση. Πολλές υπηρεσίες εξάγουν μικρότερες γεωγραφικές περιοχές απ' ότι το OSM. Η σελίδα Geofabrik παρέχει εκχυλίσματα για μεγάλες περιοχές όπως χώρες ή πολιτείες, από τα οποία μπορούμε να προετοιμάσουμε τα δικά μας μικρότερα εκχυλίσματα οριοθετημένων κουτιών χρησιμοποιώντας το Osmosisή osmconvert. Τα δεδομένα OSM μπορούν να διανεμηθούν σε μορφή XML ή σε πιο συμπαγή δυαδική μορφή

(PBF). Το OTP μπορεί να καταναλώνει και τα δυο, αλλά πάντα δουλεύουμε με το PBF αφού είναι μικρότερο και ταχύτερο. Κατεβάζουμε τα δεδομένα OSM PBF για την ίδια γεωγραφική περιοχή με αυτή των GTFS και τα τοποθετούμε στον φάκελο otp που έχουμε δημιουργήσει. Αν χρησιμοποιούμε τη ροή TriMet, το εκχύλισμα για το Portland θα κάνει τη δουλειά.

Αφού κατεβάσουμε τα GTFS και OSM δεδομένα, πρέπει να δώσουμε τις κατάλληλες εντολές προκειμένου να δημιουργήσουμε τον χάρτη OTP που επιθυμούμε. Ως πρόγραμμα Java, το OTP πρέπει να εκτελείται υπό εικονική μηχανή Java (JVM), η οποία παρέχεται ως τμήμα του Javaruntime (JRE) ή του Java development kit (JDK). Εκτελούμε την εντολή java-version για να ελέγξουμε ότι είναι εγκατεστημένη η έκδοση 1.8 ή κάποια νεότερη έκδοση του JVM. Αν δεν γίνει αυτό, τότε θα χρειαστεί να εγκαταστήσουμε ένα πρόσφατο πακέτο Open JDK ή Oracle Java για το λειτουργικό μας σύστημα.

Τα σύνολα δεδομένων GTFS και OSM είναι συχνά πολύ μεγάλα και το OTP "τρώει" αρκετή μνήμη. Χρειαζόμαστε τουλάχιστον 1GB μνήμης κατά την εργασία με το σύνολο δεδομένων Portland TriMet και μερικά gigabytes για μεγαλύτερες εισόδους. Η γενική εντολή για δημιουργία ενός γραφήματος είναι:

Java –XmxAG –jar otp-B.shaded.jar --build /home/username/otp --inMemory όπου Xmx: το όριο της ποσότητας που επιτρέπεται να καταναλώνει η μνήμη OTP, A: ο αριθμός σε GB π.χ 1,2,3,4,... ή και σε MB,KB, B: η έκδοση του JAR που κατεβάσαμε, home/username/otp ο φάκελος όπου έχουμε βάλει τα αρχεία εισόδου (OSM και GTFS δεδομένα)

Η διαδικασία δημιουργίας του γραφήματος θα διαρκέσει κάποια λεπτά και όταν εμφανιστεί το μήνυμα Grizzly server running σημαίνει πως είναι έτοιμο το γράφημα. Για να το εμφανίσουμε πρέπει να πληκτρολογήσουμε σε ένα πρόγραμμα περιήγησης ιστού (Mozilla Firefox, Google Chrome, Opera κτλ) την διεύθυνση http://localhost:8080/. Τώρα είναι έτοιμο το παράδειγμα OTP προς χρήση.

5.3.2 Δημιουργία OTP για την Πόλη του Βόλου

Αφού προαναφέραμε τη γενική διαδικασία δημιουργίας γραφήματος OTP, θα παρουσιάσουμε τώρα τα βήματα και τις εντολές για δημιουργία γραφήματος για την πόλη του Βόλου. Αρχικά κατεβάζουμε το αρχείο otp-1.2.0-shaded.jar ακολουθώντας την εξής διαδρομή:

https://repo1.maven.org/maven2/org/opentripplanner/otp/→ 1.2.0/ → otp-1.2.0-shaded.jar

Τοποθετούμε το αρχείο αυτό στην επιφάνεια εργασίας του υπολογιστή μας. Εν συνεχεία κατεβάζουμε το Java virtual machine (JVM) με έκδοση 1.8 από το site https://java.com/en/download/. Επόμενο βήμα είναι να δημιουργήσουμε έναν κενό φάκελο με το όνομα data και να τον τοποθετήσουμε στην επιφάνεια εργασίας. Σε αυτόν τον φάκελο θα προσθέσουμε τα GTFS και OSM δεδομένα. Τα GTFS δεδομένα τα διαθέτουμε, όπως έχουμε προαναφέρει στην εργασία, και σε .zip μορφή όπως επιτάσσει το πρόγραμμα OTP. Οπότε τα τοποθετούμε μέσα στον φάκελο data.

Τα OSM δεδομένα τα κατεβάζουμε από το διαδίκτυο και πιο συγκεκριμένα από την σελίδα Geofabrik ως εξής:

http://download.geofabrik.de/ → Europe → Greece → greece-latest.osm.pbf

Αφού κατεβάσουμε και το greece-latest.osm.pbf, το τοποθετούμε στον φάκελο data. Επομένως συμπληρώσαμε τον φάκελο data με τα δύο είδη δεδομένων που χρειαζόμασταν. Το μόνο που απομένει είναι να δώσουμε την κατάλληλη εντολή προκειμένου να ξεκινήσει η δημιουργία του γραφήματος που επιθυμούμε. Την εντολή θα την δώσουμε μέσω της Γραμμής Εντολών (Command Prompt).

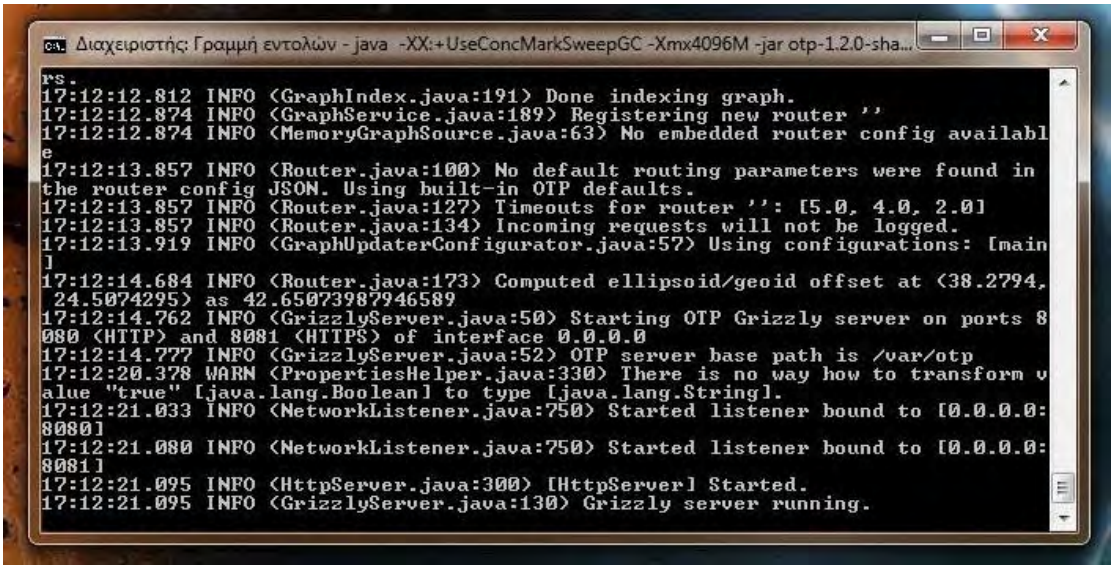
Ανοίγουμε την Γραμμή Εντολών. Γράφουμε cd desktop και πατάμε το πλήκτρο Enter. Και ακριβώς στην επόμενη γραμμή πληκτρολογούμε την εξής εντολή:

java -XX:+UseConcMarkSweepGC -Xmx4096M -jarotp-1.2.0-shaded.jar --builddata --inMemory

Εικόνα 5.1: Εντολές για Δημιουργία Γραφήματος OTP για την Πόλη του Βόλου

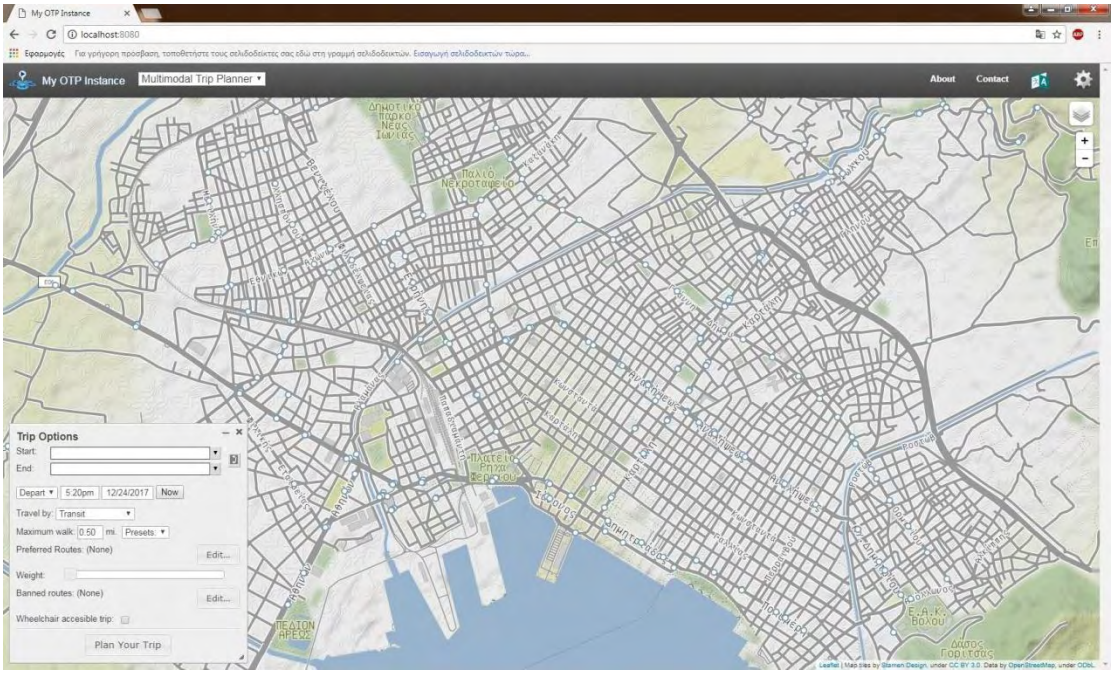
Μας βολεύει καλύτερα να αναγράψουμε την τιμή Xmx σε megabytes, ενώ η εντολή UseConcMarkSweepGC είναι απαραίτητη προκειμένου να ελευθερώσει μνήμη στον υπολογιστή και να δημιουργηθεί εν τέλει το γράφημα για τον Βόλο. Ο συλλέκτης αυτός χρησιμοποιείται σε εφαρμογές με σχετικά μεγάλο σύνολο δεδομένων μακράς διάρκειας και οι οποίες λειτουργούν σε υπολογιστή με δύο ή περισσότερους επεξεργαστές, όπως συμβαίνει ακριβώς στην περίπτωση που μελετάμε.

Αφού τρέξουμε και τις εντολές που προαναφέραμε και δείχνουμε στην εικόνα 5.1, περιμένουμε κάποια λεπτά προκειμένου να μας εμφανίσει το μήνυμα Grizzly server running που σημαίνει πως είναι έτοιμο το γράφημα.



Εικόνα 5.2: Το Γράφημα έχει δημιουργηθεί με Επιτυχία

Επομένως έχουμε ολοκληρώσει με την δημιουργία του γραφήματος για την πόλη του Βόλου. Ανοίγουμε το Google Chrome και πληκτρολογούμε localhost:8080 και εμφανίζεται ο χάρτης. Σαν αρχική εικόνα εμφανίζεται ο χάρτης της Ελλάδας. Με μεγέθυνση στην περιοχή του Βόλου εμφανίζονται όλες οι στάσεις.



Εικόνα 5.3: Γράφημα OTP για την Πόλη του Βόλου

5.4 Παραδείγματα OTP για την Πόλη του Βόλου

Αφού δημιουργήσαμε το γράφημα OTP για τον Βόλο, θα εφαρμόσουμε κάποια παραδείγματα για να κατανοήσουμε το πως λειτουργεί αυτό. Στη θέση Start αναγράφουμε το σημείο εκκίνησης (δηλαδή, την στάση από την οποία επιθυμούμε να αναχωρήσουμε) και στη θέση End το σημείο προορισμού (δηλαδή, την στάση στην οποία επιθυμούμε να αφιχθούμε). Επιλέγουμε την ώρα και την ημερομηνία που επιθυμούμε είτε να αναχωρήσουμε (depart) είτε να αφιχθούμε (arrive). Όλες οι επιλογές που διαθέτουμε φαίνονται στην εικόνα 5.3.

Στα παραδείγματα που πραγματοποιήσαμε, ρυθμίσαμε την επιλογή σε Arrive, ώρα 6:00 pm και ημερομηνία 10/19/2017. Στο κουτί Travel by επιλέγουμε την εντολή Transit, ενώ στο Maximum walk (δηλαδή, μέγιστο περπάτημα μέχρικάποια στάση) 0,10 miles (160 μέτρα). Οι άλλες επιλογές που συμπληρώνουμε είναι οι Start και End και οι οποίες για κάθε παράδειγμα είναι οι μοναδικές που αλλάζουν. Στα αποτελέσματα των παραδειγμάτων αναγράφουμε αριθμό λεωφορείου, ώρα αναχώρησης, ώρα άφιξης, πιθανές μετεπιβιβάσεις και πιθανή πεζοπορία προς κάποια στάση. Κάθε παράδειγμα δίνει τρεις λύσεις, οι οποίες λαμβάνουν υπόψην όλα τα παραπάνω κριτήρια.

❖ **ΠΑΡΑΔΕΙΓΜΑ 1**

Θεωρούμε στάση αναχώρησης: Αγίου Βασιλείου και στάση άφιξης: Δημαρχείο.

1η λύση: **Ώρα αναχώρησης**: 5:42, **λεωφορείο** ΝΟ4, **ώρα άφιξης**: 5:49

2η λύση: **Ώρα αναχώρησης**: 5:38, **λεωφορείο** ΝΟ15, **ώρα άφιξης**: 5:45

3ηλύση: **Ώρα αναχώρησης**: 5:27, **λεωφορείο** ΝΟ4, **ώρα άφιξης**: 5:34

❖ **ΠΑΡΑΔΕΙΓΜΑ 2**

Θεωρούμε σημείο αναχώρησης: Αφετηρία και σημείο άφιξης: Κασσαβέτη (επί της Πολυμέρη).

1ηλύση: **Ώρα αναχώρησης**: 5:40, **λεωφορείο** ΝΟ5, ώρα άφιξης στην στάση Δεληγιώργη: 5:48 και στη συνέχεια πεζοπορία για 120 μέτρα, **ώρα άφιξης**: 5:49

2η λύση: **Ώρα αναχώρησης**: 5:39, πεζοπορία για 82 μέτρα μέχρι την στάση Κτελ (προς Ιάσονος) και στη συνέχεια ώρα αναχώρησης από την στάση: 5:40, **λεωφορείο** ΝΟ3, ώρα άφιξης στην στάση Σόλωνος (επί της Ιάσονος): 5:44 και στη συνέχεια ώρα αναχώρησης από την στάση: 5:46, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Δεληγιώργη: 5:51 και στη συνέχεια πεζοπορία για 120 μέτρα, **ώρα άφιξης**: 5:52

3η λύση: **Ώρα αναχώρησης**: 5:20, **λεωφορείο** ΝΟ5, ώρα άφιξης στην στάση Δεληγιώργη: 5:28 και στη συνέχεια πεζοπορία για 120 μέτρα, **ώρα άφιξης**: 5:29

❖ **ΠΑΡΑΔΕΙΓΜΑ 3**

Θεωρούμε σημείο αναχώρησης: Ερμού (επί της Ιωλκού) και σημείο άφιξης: Κασσαβέτη.

1η λύση: **Ώρα αναχώρησης**: 5:42, πεζοπορία για 83 μέτρα μέχρι την στάση Ερμού (επί Ιάσονος) και στη συνέχεια ώρα αναχώρησης από την στάση: 5:43, **λεωφορείο** ΝΟ15, **ώρα άφιξης**: 5:51

2η λύση: **Ώρα αναχώρησης**: 5:27, πεζοπορία για 83 μέτρα μέχρι την στάση Ερμού (επί Ιάσονος) και στη συνέχεια ώρα αναχώρησης από την στάση: 5:28, **λεωφορείο** ΝΟ15, **ώρα άφιξης**: 5:36

3η λύση: **Ώρα αναχώρησης**: 5:12, πεζοπορία για 83 μέτρα μέχρι την στάση Ερμού (επί Ιάσονος) και στη συνέχεια ώρα αναχώρησης από την στάση: 5:13, **λεωφορείο** ΝΟ15, **ώρα άφιξης**: 5:21

❖ **ΠΑΡΑΔΕΙΓΜΑ 4**

Θεωρούμε σημείο αναχώρησης: Γέφυρα και σημείο άφιξης: Σπύρου Σπυρίδη.

1η λύση: **Ώρα αναχώρησης:** 5:40, **λεωφορείο** ΝΟ1, ώρα άφιξης στη Σόλωνος (επί Ιάσονος): 5:46 και στη συνέχεια ώρα αναχώρησης από την στάση: 5:51, **λεωφορείο**

ΝΟ4, ώρα άφιξης στην στάση Λόρδου Βύρωνος: 5:56 και στη συνέχεια πεζοπορία για 159 μέτρα, **ώρα άφιξης**: 5:58

2η λύση: **Ώρα αναχώρησης**: 5:33, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Δημαρχείο Νέας Ιωνίας (Μαιάνδρου): 5:35 και στη συνέχεια πεζοπορία για 91 μέτρα μέχρι την στάση Δημαρχείο Νέας Ιωνίας και ώρα αναχώρησης από την στάση: 5:42, **λεωφορείο** ΝΟ11, ώρα άφιξης στη στάση Σπύρου Σπυρίδη (προς Αφετηρία): 5:53 και στη συνέχεια πεζοπορία για 125 μέτρα, **ώρα άφιξης**: 5:55

3η λύση: **Ώρα αναχώρησης**: 5:28, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Αγίου Παντελεήμωνα: 5:29 και στη συνέχεια ώρα αναχώρησης από την στάση: 5:36, **λεωφορείο** ΝΟ15, ώρα άφιξης στην στάση Λόρδου Βύρωνος: 5:47 και στη συνέχεια πεζοπορία για 159 μέτρα, **ώρα άφιξης**: 5:49

- **ΠΑΡΑΔΕΙΓΜΑ 5**

Θεωρούμε σημείο αναχώρησης: Δημαρχείο Νέας Ιωνίας (Μαιάνδρου) και σημείο άφιξης: Ταχυδρομείο (επί της Λαρίσης).

1η λύση: **Ώρα αναχώρησης**: 5:37, πεζοπορία για 91 μέτρα μέχρι την στάση Δημαρχείο Νέας Ιωνίας και στη συνέχεια ώρα αναχώρησης από την στάση: 5:38, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Σόλωνος (επί Ιάσονος): 5:46 και στη συνέχεια πεζοπορία για 91 μέτρα μέχρι την στάση Σόλωνος και ώρα αναχώρησης από την στάση: 5:49, **λεωφορείο** ΝΟ3, **ώρα άφιξης**: 5:55

2η λύση: **Ώρα αναχώρησης**: 5:35, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Πέτρου και Παύλου: 5:41 και στη συνέχεια πεζοπορία για 28 μέτρα μέχρι την στάση Πέτρου και Παύλου (επί Μυτιλήνης) και ώρα αναχώρησης από την στάση: 5:47, **λεωφορείο** ΝΟ3, ώρα άφιξης στην στάση Ταχυδρομείο (επί Γρ. Λαμπράκη): 5:53 και στη συνέχεια πεζοπορία για 26 μέτρα, **ώρα άφιξης**: 5:54

3η λύση: **Ώρα αναχώρησης**: 5:23, **λεωφορείο** ΝΟ1, ώρα άφιξης στην στάση Πέτρου και Παύλου: 5:29 και στη συνέχεια πεζοπορία για 28 μέτρα μέχρι την στάση Πέτρου και Παύλου (επί Μυτιλήνης) και ώρα αναχώρησης από την στάση: 5:32, **λεωφορείο** ΝΟ3, ώρα άφιξης στην στάση Ταχυδρομείο (επί Γρ. Λαμπράκη): 5:38 και στη συνέχεια πεζοπορία για 26 μέτρα, **ώρα άφιξης**: 5:39

- **ΠΑΡΑΔΕΙΓΜΑ 6**

Θεωρούμε σημείο αναχώρησης: Κουταρέλια και σημείο άφιξης: Κωνσταντά (επί της Παγασών).

1η λύση: **Ώρα αναχώρησης**: 5:39, **λεωφορείο** ΝΟ2, **ώρα άφιξης**: 5:45

2η λύση: **Ώρα αναχώρησης**: 5:24, **λεωφορείο** ΝΟ2, **ώρα άφιξης**: 5:30

3η λύση: **Ώρα αναχώρησης**: 5:09, **λεωφορείο** ΝΟ2, **ώρα άφιξης**: 5:15

Τέλος, ο χρόνος επίλυσης των παραδειγμάτων υπολογίζεται με τη βοήθεια ενός χρονομέτρου. Δηλαδή αφού συμπληρώσουμε όλες τις επιλογές που θέλουμε για ένα παράδειγμα OTP, όταν "πατήσουμε" την εντολή PlanYourTrip (σχεδιασμός της διαδρομής που επιθυμούμε), τότε χρησιμοποιούμε μια εξωτερική συσκευή που διαθέτει χρονόμετρο. Ταυτόχρονα με την εντολή Plan Your Trip ξεκινάει να τρέχει και το χρονόμετρο. Με το που επιλυθεί το παράδειγμα, σταματάμε το χρονόμετρο.

Ο υπολογιστικός χρόνος όλων των παραπάνω παραδειγμάτων είναι ίδιος και ίσος με 1 sec.

ΠΑΡΑΔΕΙΓΜΑ	ΠΙΘΑΝΕΣ ΛΥΣΕΙΣ	ΧΡΟΝΟΣ ΥΛΟΠΟΙΗΣΗΣ (sec)	ΤΙΜΗ ΑΝΤΙΚΕΙΜΕΝΙΚΗΣ ΣΥΝΑΡΤΗΣΗΣ (min)
1	1η 2η 3η	1 1 1	7 7 7
2	1η 2η 3η	1 1 1	9 13 9
3	1η 2η 3η	1 1 1	9 9 9
4	1η 2η 3η	1 1 1	18 22 21
5	1η 2η 3η	1 1 1	18 19 16
6	1η 2η 3η	1 1 1	6 6 6

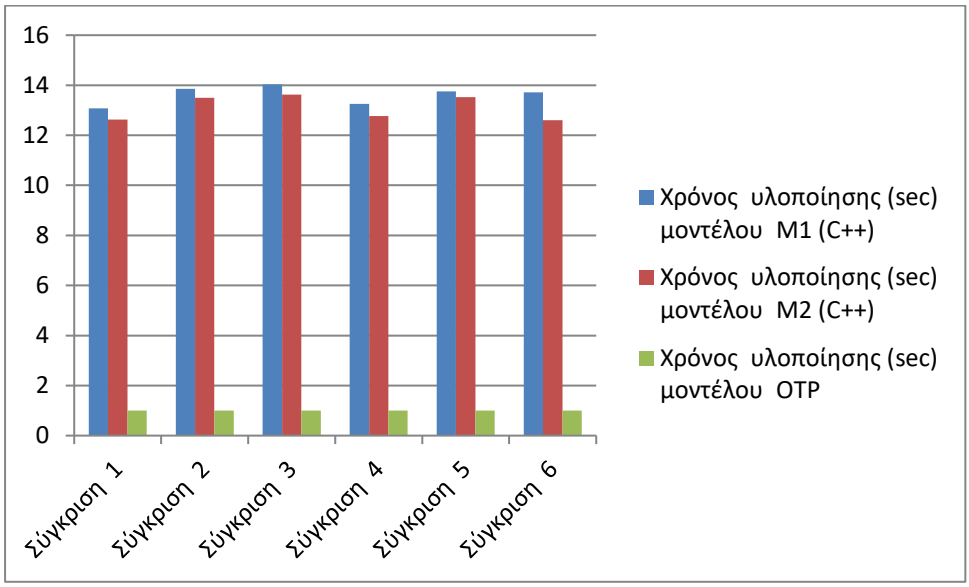
Πίνακας 5.1: Αποτελέσματα των 6 Παραδειγμάτων του OTP για την Πόλη του Βόλου

5.5 Σύγκριση OTP και C++

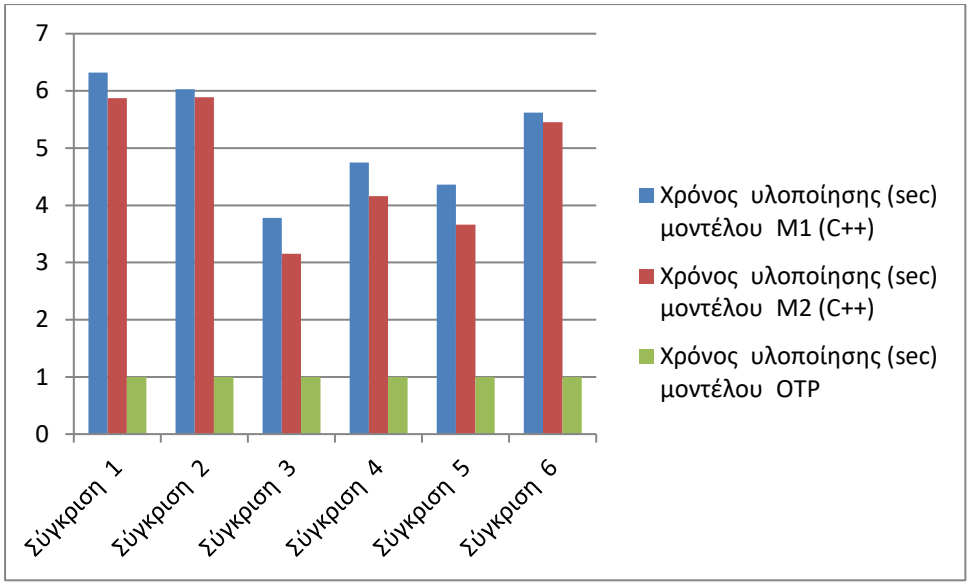
Σε αυτή την ενότητα θα πραγματοποιηθεί σύγκριση μεταξύ των προγραμμάτων OTP και C++. Βασικό στοιχείο για αυτή τη σύγκριση είναι ο υπολογιστικός χρόνος, δηλαδή ο χρόνος επίλυσης των παραδειγμάτων που πραγματοποιήσαμε. Στην προηγούνενη ενότητα υπολογίσαμε τους χρόνους επίλυσης για τα παραδείγματα OTP, ενώ στο κεφάλαιο 4 τους υπολογιστικούς χρόνους για τα παραδείγματα και με τα δύο μοντέλα ακέραιου

προγραμματισμού (C++). Από αυτούς τους χρόνους θα γίνει η σύγκριση των δύο προγραμμάτων, η οποία θα παρουσιαστεί με τη βοήθεια σχεδιαγραμμάτων.

Διαπιστώσαμε πως ο χρόνος επίλυσης για οποιοδήποτε παράδειγμα OTP για την πόλη του Βόλου είναι 1 sec, οπότε συγκρίνουμε αύτη την τιμή με τις τιμές (υπολογιστικούς χρόνους) κάθε παραδείγματος που αναφέρθηκε στο κεφάλαιο 4 και για τους δύο τύπους δεδομένων (Theoritical και GTFS). Δηλαδή, συγκρίνουμε τους υπολογιστικούς χρόνους του παραδείγματος 1 με το 1 sec, του παραδείγματος 2 με το 1 sec, του παραδείγματος 3 με το 1 sec και κάνουμε το ίδιο μέχρι το παράδειγμα 6. Κάθε παράδειγμα συμπεριλαμβάνει τους χρόνους υλοποίησης και για τα δύο μοντέλα (M1 και M2). Επειδή τα μοντέλα με ακέραιο προγραμματισμό πραγματοποιήθηκαν τόσο για θεωρητικά όσο και για GTFS δεδομένα, δημιουργούμε δύο διαγράμματα που περιέχουν στον οριζόντιο άξονα τους αριθμούς σύγκρισης και στον κάθετο άξονα τους χρόνους επίλυσης των C++ και OTP. Ως αριθμό σύγκρισης ορίζουμε τον αριθμό παραδείγματος των Theoritical ή GTFS δεδομένων που συγκρίνεται με οποιοδήποτε παράδειγμα OTP (που έχει χρόνο επίλυσης πάντα 1 sec για την πόλη του Βόλου), π.χ σύγκριση 1 για GTFS δεδομένα είναι η σύγκριση χρόνου επίλυσης του παραδείγματος 1 (M1 και M2) για GTFS δεδομένα με το 1 sec του OTP.



Διάγραμμα 5.1: Χρόνοι Σύγκρισης C++ και OTP για Θεωρητικά Δεδομένα



Διάγραμμα 5.2: Χρόνοι Σύγκρισης C++ και OTP για GTFS Δεδομένα

Και από τα δύο διαγράμματα παρατηρούμε πως ο υπολογιστικός χρόνος του OTP είναι πολύ μικρότερος από αυτόν της C++. Σαν C++ συμπεριλαμβάνουμε και τα δύο μοντέλα δρομολόγησης που παρουσιάστηκαν στο κεφάλαιο 3. Επομένως, το πρόγραμμα OTP είναι ταχύτερο στην εύρεση βέλτιστης διαδρομής από το πρόγραμμα C++.

5.6 Σύνοψη

Στο παρόν κεφάλαιο έγινε αναφορά στην εφαρμογή OpenTripPlanner. Παρουσιάστηκαν κάποιες βασικές πληροφορίες του προγράμματος αυτού, καθώς και η διαδικασία που ακολουθείται προκειμένου να δημιουργηθεί το OTP παράδειγμα για την πόλη που επιθυμούμε. Ακόμη πραγματοποιήθηκε δημιουργία χάρτη OTP για την πόλη του Βόλου και παρουσιάστηκαν τα αποτελέσματα από κάποια παραδείγματα που πραγματοποιήθηκαν. Τέλος, έγινε σύγκριση των προγραμμάτων C++ και OTP ως προς τους χρόνους επίλυσής τους.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα διπλωματική εργασία πραγματεύτηκε ένα πρόβλημα δρομολόγησης, το οποίο υπολογίζει τη βέλτιστη διαδρομή που πρέπει να ακολουθήσει ένας χρήστης, ο οποίος θα μετακινηθεί με αστική συγκοινωνία από ένα σημείο της πόλης του Βόλου σε ένα άλλο. Το πρόβλημα αυτό στηρίχθηκε στη Θεωρία των Γράφων, η οποία πρωτοεμφανίστηκε το 1736 και με την πάροδο του χρόνου παρουσίασε σημαντική ανάπτυξη. Επίσης, θα μπορούσε να ειπωθεί πως το πρόβλημα ανήκει στην κατηγορία των οδικών δικτύων. Τα δίκτυα και οι γράφοι συνδέονται μεταξύ τους, καθώς η απλούστευση των δικτύων γίνεται με μετατροπή τους σε γραφήματα.

Ακόμη, το πρόβλημα εύρεσης βέλτιστης διαδρομής βασίστηκε τόσο στο πρόβλημα δρομολόγησης της συντομότερης διαδρομής όσο και στον ακέραιο προγραμματισμό. Όπως έχουμε προαναφέρει, εφαρμόστηκε και για θεωρητικά και για GTFS δεδομένα. Τα θεωρητικά δεδομένα είναι υποθετικά, ενώ τα GTFS είναι πραγματικά και χρησιμοποιούνται κυρίως για την παροχή δεδομένων μέσων μαζικής μεταφοράς για χρήση πολυτροπικών ταξιδιών. Επειδή τα GTFS δεδομένα για την πόλη του Βόλου διαθέτουν μεγάλο αριθμό στάσεων, επιλύεται το πρόβλημα που μελετάμε για πολύ μικρότερο αριθμό στάσεων.

Το πρόβλημα δρομολόγησης μοντελοποιήθηκε με ακέραιο προγραμματισμό. Έγιναν δύο μοντελοποιήσεις: μοντέλο Μ1 και μοντέλο Μ2. Στο μοντέλο Μ1 έγινε αντιστοίχιση κόμβου με στάση, ενώ το μοντέλο Μ2 είναι ακριβώς ίδιο με το Μ1 με επιπλέον προσθήκη ενός ευρετικού κανόνα. Το Μ1 είναι ρεαλιστικό μοντέλο και δίνει ακριβή λύση, ενώ το Μ2 είναι προσεγγιστικό και δίνει ευρετική λύση. Ο ευρετικός κανόνας εμπλέκει κόστος μετακίνησης στα θεωρητικά δεδομένα και χρόνο μετακίνησης στα GTFS δεδομένα.

Και τα δύο μοντέλα αναπτύχθηκαν σε γλώσσα προγραμματισμού C++ και για την επίλυσή τους χρησιμοποιήθηκε solver IBM ILOG CPLEX Optimization Studio Version 12.7. Από παραδείγματα που εφαρμόστηκαν και για τα δύο μοντέλα και για τους δύο τύπους δεδομένων, προκύπτει πως το μοντέλο Μ2 επιλύει ταχύτερα το πρόβλημα και οι τιμές της αντικειμενικής συνάρτησης όχι μόνο δε διαφέρουν πολύ από αυτές του μοντέλου Μ1, αλλά σε αρκετές περιπτώσεις είναι ακριβώς ίδιες. Επομένως, το μοντέλο Μ2 μπορεί να θεωρηθεί μία βελτίωση του Μ1.

Η εύρεση βέλτιστης διαδρομής με χρήση αστικής συγκοινωνίας μπορεί να υπολογιστεί όχι μόνο με μοντέλο ακέραιου προγραμματισμού, αλλά και με χρήση της εφαρμογής OpenTripPlanner. Έτσι δημιουργήθηκε ένας χάρτης OTP για την πόλη του Βόλου, στον οποίο εφαρμόστηκαν κάποια παραδείγματα, τα οποία δίνουν τρεις λύσεις. Από τα

αποτελέσματα των παραδειγμάτων που πραγματοποιήθηκαν τόσο στην C++ (M1 και M2) όσο και στο OTP, διαπιστώθηκε πως το OTP επιλύει το πρόβλημα σε ταχύτερο υπολογιστικό χρόνο από την C++.

Καταληκτικά, στην εργασία παρουσιάσαμε δύο προγράμματα που υπολογίζουν τη βέλτιστη διαδρομή που ακολουθεί ένας χρήστης για να μετακινηθεί με αστική συγκοινωνία στην πόλη του Βόλου και ένα που υπολογίζει μία εφικτή διαδρομή, η οποία μπορεί να είναι και αυτή βέλτιστη κάποιες φορές: τα δύο είναι με μοντέλο ακέραιου προγραμματισμού και το άλλο με χρήση του OpenTripPlanner. Με αυτά τα προγράμματα μπορούμε να υπολογίσουμε για οποιαδήποτε πόλη τη βέλτιστη διαδρομή, εφόσον έχουμε τα κατάλληλα δεδομένα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Wikipedia, ‹‹ Θεωρία των Γράφων ›› [Ηλεκτρονικό]. Διαθέσιμο: https://el.wikipedia.org/wiki/Θεωρία_γράφων.

[2] Έννοιες πάνω στη Θεωρία των Γράφων [Ηλεκτρονικό].Διαθέσιμο: http://eclass.uth.gr/eclass/modules/document/file.php/INFS170/ΘΕΩΡΙΑ%20ΓΡΑΦΩΝ_1η%20Διάλεξη.pdf.

[3] Αναπαράσταση Γράφων [Ηλεκτρονικό]. Διαθέσιμο: https://repository.kallipos.gr/bitstream/11419/461/1/06_chapter_05.pdf.

[4] Ahuja RK, Magnanti TL and Orlin JB, (1993). “Network Flows: Theory, Algorithms and Applications,” Prentice Hall.

[5] Ακέραιος Προγραμματισμός [Ηλεκτρονικό]. Διαθέσιμο: http://www.mie.uth.gr/ekp_yliko/ΑΚΕΡΑΙΟΣΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣΚ1.pdf.

[6] Edward J. Taaffe, Howard L. Gauthier, Morton E. O’ Kelly, ‹‹ Geography of Transportation ››, Ohio State University.

[7] Moonis Raza, Yash Aggarwal, ‹‹ Transport Geography of India: Commodity Flows and the Regional Structure of th Indian Economy››.

[8] Wikipedia, General Transit Feed Specification [Ηλεκτρονικό]. Διαθέσιμο: https://en.wikipedia.org/wiki/General_Transit_Feed_Specification.

[9] Google, πληροφορίες για GTFS [Ηλεκτρονικό]. Διαθέσιμο: https://developers.google.com/transit/gtfs/.

[10] Γ. Κ. Δρ. Σαχαρίδης, Δημήτριος Ριζόπουλος, (2016). ‹‹ Οδηγός Green Your Move ››,Βόλος.

[11] Πληροφορίες για GTFS δεδομένα [Ηλεκτρονικό]. Διαθέσιμο: https://www.slideshare.net/LIFEGreenYourMove/presentation-data-collection-and-gtfs.

[12] Βασική χρήση Opentripplanner [Ηλεκτρονικό]. Διαθέσιμο: http://opentripplanner.readthedocs.io/en/latest/Basic-Usage/.

[13] Wikipedia, ‹‹ Opentripplanner ›› [Ηλεκτρονικό]. Διαθέσιμο: https://wiki.openstreetmap.org/wiki/OpenTripPlanner.

[14] Πληροφορίες για OTP [Ηλεκτρονικό]. Διαθέσιμο: http://www.opentripplanner.org/otp/.

[15] Wikipedia, ‹‹ World Geometric System ›› [Ηλεκτρονικό]. Διαθέσιμο: https://en.wikipedia.org/wiki/World_Geodetic_System.

[16] Wikipedia, ‹‹ Ιεραρχίες Συρρίκνωσης ›› [Ηλεκτρονικό]. Διαθέσιμο: https://en.wikipedia.org/wiki/Contraction_hierarchies.

[17] Αστικό Κτελ Βόλου [Ηλεκτρονικό]. Διαθέσιμο: https://astikovolou.gr.

[18] Luigi Moccia, Jean-Francois Cordeau, Gilbert Laporte, Stefan Ropke, Maria Pia Valentini (2008), "Modeling and Solving a Multimodal Routing Problem with Timetables and Time Windows" , Universite de Montreal , Pavillon Palasis-Prince.

[19] Aybike Ozdemirel, Burak Gokgur, Deniz Tursel Eliiyi (2012), "An assignment and routing problem with time windows and capacity restriction", Elsevier Ltd 54 149-158.

[20] Wikipedia, ‹‹ Routing ›› [Ηλεκτρονικό]. Διαθέσιμο: https://en.wikipedia.org/wiki/Routing.

ΠΑΡΑΡΤΗΜΑ Α

Κώδικας C++, Μοντέλο M1 για Theoritical Δεδομένα

```
#include"stdafx.h"
#include<ilcplex/ilocplex.h>

#include<string>
#include<iterator>
#include<iostream>
#include<fstream>
#include<istream>
#include<sstream>
#include<vector>
usingnamespace std;

//generic variables and constants
int i;
int j;
int k;
int c;
int m;
int n;
int z;
int a;
constint imax = 25;
constint jmax = 25;
constint kmax = 25;
constint cmax = 3;
constint mmax = 3;
constint nmax = 3;
constint zmax = 3;
constint amax = 1;
constint ToA = 960;
constint N = 25;
constint O = 17;
constint D = 24;
constint M = 1000000;


double C[imax][jmax][mmax];
double d[imax][jmax][mmax][nmax];
double t[imax][jmax][mmax];

//start of functions declaration, for loading data and declaring the mathematical model

classCSVRow
```

```
{
public:
  std::stringconst& operator[](std::size_tindex) const
{
return m_data[index];
}
  std::size_t size() const
{
return m_data.size();
}
void readNextRow(std::istream&str)
{
  std::string        line;
  std::getline(str, line);

  std::stringstream   lineStream(line);
  std::string        cell;

  m_data.clear();
while (std::getline(lineStream, cell, '       '))
{
  m_data.push_back(cell);
}
// This checks for a trailing comma with no data after it.
if (!lineStream && cell.empty())
{
// If there was a trailing comma then add an empty element.
m_data.push_back("");
          }
}
private:
  std::vector<std::string>    m_data;
};

// You forgot this function.
// It defines how to read from a stream one row of the CSV file.
  std::istream& operator>>(std::istream&str, CSVRow&data)
{
data.readNextRow(str);
returnstr;
}

classCSVIterator
{
public:

typedef std::input_iterator_tagiterator_category;
typedefCSVRowvalue_type;
```

```
typedef std::size_tdifference_type;
typedefCSVRow*                  pointer;
typedefCSVRow&reference;

  CSVIterator(std::istream&str) :m_str(str.good() ? &str : NULL) { ++(*this); }
  CSVIterator() :m_str(NULL) {}

// Pre Increment
CSVIterator& operator++()             { if (m_str) { if (!((*m_str) >> m_row)){ m_str = NULL;
} }return *this; }
// Post increment
CSVIterator operator++(int)           { CSVIterator    tmp(*this); ++(*this); return tmp; }
CSVRowconst& operator*()   const      { return m_row; }
CSVRowconst* operator->()  const      { return&m_row; }

bool operator==(CSVIteratorconst&rhs) { return ((this == &rhs) || ((this->m_str == NULL)
&& (rhs.m_str == NULL))); }
bool operator!=(CSVIteratorconst&rhs) { return !((*this) == rhs); }
private:
  std::istream*      m_str;
CSVRow             m_row;
};

//we declare the data of the problem, declared under the constants of as, ae, E, as, S and initial
int load_data_inteneraries_1()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
        std::ifstream  file_1("data1/iteneraries/basic_JP_departures-
Mode_1_Itineraries_1.csv");

        int counter = 0;
        for (CSVIterator loop(file_1); loop != CSVIterator(); ++loop){
                for (i = 0; i < imax; i++){
                        d[counter][i][0][0] = std::stod((*loop)[i]);
                }
                counter += 1;
        }

        return 0;
}

int load_data_inteneraries_2()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
        std::ifstream  file_2("data1/iteneraries/basic_JP_departures-
Mode_1_Itineraries_2.csv");

        int counter = 0;
```

```
	for (CSVIterator loop(file_2); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][0][1] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_3()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_3("data1/iteneraries/basic_JP_departures-
Mode_1_Itineraries_3.csv");

	int counter = 0;
	for (CSVIterator loop(file_3); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][0][2] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_4()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_4("data1/iteneraries/basic_JP_departures-
Mode_2_Itineraries_1.csv");

	int counter = 0;
	for (CSVIterator loop(file_4); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][1][0] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_5()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
```

```
	std::ifstream  file_5("data1/iteneraries/basic_JP_departures-
Mode_2_Itineraries_2.csv");

	int counter = 0;
	for (CSVIterator loop(file_5); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][1][1] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_6()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_6("data1/iteneraries/basic_JP_departures-
Mode_2_Itineraries_3.csv");

	int counter = 0;
	for (CSVIterator loop(file_6); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][1][2] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_7()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_7("data1/iteneraries/basic_JP_departures-
Mode_3_Itineraries_1.csv");

	int counter = 0;
	for (CSVIterator loop(file_7); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][2][0] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}
```

```
int load_data_inteneraries_8()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_8("data1/iteneraries/basic_JP_departures-
Mode_3_Itineraries_2.csv");

	int counter = 0;
	for (CSVIterator loop(file_8); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][2][1] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_inteneraries_9()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_9("data1/iteneraries/basic_JP_departures-
Mode_3_Itineraries_3.csv");

	int counter = 0;
	for (CSVIterator loop(file_9); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			d[counter][i][2][2] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_travel_cost_1()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_1a("data1/travel_cost/theoritical_1a.csv");

	int counter = 0;
	for (CSVIterator loop(file_1a); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			C[counter][i][0] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
```

```
}

int load_data_travel_cost_2()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_2a("data1/travel_cost/theoritical_2a.csv");

	int counter = 0;
	for (CSVIterator loop(file_2a); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			C[counter][i][1] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_travel_cost_3()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream   file_3a("data1/travel_cost/theoritical_3a.csv");

	int counter = 0;
	for (CSVIterator loop(file_3a); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			C[counter][i][2] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_time_1()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_1b("data1/time/theoritical_1.csv");

	int counter = 0;
	for (CSVIterator loop(file_1b); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			t[counter][i][0] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
```

```
}

int load_data_time_2()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_2b("data1/time/theoritical_2.csv");

	int counter = 0;
	for (CSVIterator loop(file_2b); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			t[counter][i][1] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int load_data_time_3()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio
	std::ifstream  file_3b("data1/time/theoritical_3.csv");

	int counter = 0;
	for (CSVIterator loop(file_3b); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			t[counter][i][2] = std::stod((*loop)[i]);
		}
		counter += 1;
	}

	return 0;
}

int main(intargc, char **argv)
{
int status;

status = load_data_inteneraries_1();
status = load_data_inteneraries_2();
status = load_data_inteneraries_3();
status = load_data_inteneraries_4();
status = load_data_inteneraries_5();
status = load_data_inteneraries_6();
status = load_data_inteneraries_7();
status = load_data_inteneraries_8();
status = load_data_inteneraries_9();
status = load_data_travel_cost_1();
```

```
status = load_data_travel_cost_2();
status = load_data_travel_cost_3();
status = load_data_time_1();
status = load_data_time_2();
status = load_data_time_3();


IloEnv env;
	try{
		IloModel model(env);
		typedefIloArray<IloNumArray>IloNumMatrix2x2;
		typedefIloArray<IloNumMatrix2x2>IloNumMatrix3x3;
		typedefIloArray<IloNumMatrix3x3>IloNumMatrix4x4;
		typedefIloArray<IloNumMatrix4x4>IloNumMatrix5x5;
		typedefIloArray<IloNumMatrix5x5>IloNumMatrix6x6;
		typedefIloArray<IloNumMatrix6x6>IloNumMatrix7x7;
		typedefIloArray<IloNumVarArray>IloNumVarMatrix2x2;
		typedefIloArray<IloNumVarMatrix2x2>IloNumVarMatrix3x3;
		typedefIloArray<IloNumVarMatrix3x3>IloNumVarMatrix4x4;
		typedefIloArray<IloNumVarMatrix4x4>IloNumVarMatrix5x5;
		typedefIloArray<IloNumVarMatrix5x5>IloNumVarMatrix6x6;
		typedefIloArray<IloNumVarMatrix6x6>IloNumVarMatrix7x7;
		typedefIloArray<IloRangeArray>IloRangeMatrix2x2;
		typedefIloArray<IloRangeMatrix2x2>IloRangeMatrix3x3;
		typedefIloArray<IloRangeMatrix3x3>IloRangeMatrix4x4;
		typedefIloArray<IloRangeMatrix4x4>IloRangeMatrix5x5;
		typedefIloArray<IloRangeMatrix5x5>IloRangeMatrix6x6;
		typedefIloArray<IloRangeMatrix6x6>IloRangeMatrix7x7;



	IloCplex cplex(env);
	//--------------------------------//
	//------Metavliti Apofasis X------//
	IloNumVarMatrix4x4 Xijmn(env, 0);
	for (i = 0; i < imax; i++){
		IloNumVarMatrix3x3 Xjmn(env, 0);
		for (j = 0; j < jmax; j++){
			IloNumVarMatrix2x2 Xmn(env, 0);
			for (m = 0; m < mmax; m++){
				IloNumVarArray Xn(env, 0);
				for (n = 0; n < nmax; n++){
					char Metakinisi[70];
					sprintf_s(Metakinisi, "Xijmn(i%d,j%d,m%d,n%d)",
i, j, m, n);
				IloNumVar X(env, 0, 1, ILOINT, Metakinisi);
				Xn.add(X);
				}
```

```
                        Xmn.add(Xn);
                    }
                    Xjmn.add(Xmn);
                }
                Xijmn.add(Xjmn);
            }
            //-----------------------------------//
            //------Metavliti Apofasis AT-------//
            IloNumVarMatrix4x4 ATjimn(env, 0);
            for (j = 0; j < jmax; j++){
                IloNumVarMatrix3x3 ATimn(env, 0);
                for (i = 0; i < imax; i++){
                    IloNumVarMatrix2x2 ATmn(env, 0);
                    for (m = 0; m < mmax; m++){
                        IloNumVarArray ATn(env, 0);
                        for (n = 0; n < nmax; n++){
                            char XronAna[70];
                            sprintf_s(XronAna, "ATjimn(j%d,i%d,m%d,n%d)",
j, i, m, n);
                            IloNumVar AT(env, 0, 1440, ILOFLOAT, XronAna);
                            ATn.add(AT);
                            }
                            ATmn.add(ATn);
                    }
                    ATimn.add(ATmn);
                }
                ATjimn.add(ATimn);
            }
            //-----------------------------------//
            //------Metavliti Apofashs DT------//
            IloNumVarMatrix4x4 DTijmn(env, 0);
            for (i = 0; i < imax; i++){
                IloNumVarMatrix3x3 DTjmn(env, 0);
                for (j = 0; j < jmax; j++){
                    IloNumVarMatrix2x2 DTmn(env, 0);
                    for (m = 0; m < mmax; m++){
                        IloNumVarArray DTn(env, 0);
                        for (n = 0; n < nmax; n++){
                            char XronAna[70];
                            sprintf_s(XronAna, "DTijmn(i%d,j%d,m%d,n%d)",
i, j, m, n);
                            IloNumVar DT(env, 0, 1440, ILOFLOAT, XronAna);
                            DTn.add(DT);
                            }
                            DTmn.add(DTn);
                    }
                    DTjmn.add(DTmn);
                }
```

```
		DTijmn.add(DTjmn);
	}
	//---------------------------------//
	//------Metavliti U Sub Tour------//
	IloNumVarArray Ui(env, 0);
	for (i = 0; i < imax; i++){
		char Seira[70];
		sprintf_s(Seira, "Ui(i%d)", i);
		IloNumVar U(env, 1, N, ILOFLOAT, Seira);
		Ui.add(U);
	}
	//-------------------------------------------------//
	//--------------------Periorismoi------------------//
	//----------------Periorismos ΣXijmn<=1-------------//
	IloRangeArray Per1i(env, 0);
	for (i = 0; i < imax; i++){
		IloExpr expr(env, 0);
		for (j = 0; j < jmax; j++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (i != j){
						expr += Xijmn[i][j][m][n];
					}
				}
			}
		}
		char P1[60];
		sprintf_s(P1, "Per1i(i%d)", i);
		float LB = -IloInfinity, UB = 1;
		IloRange Per1(env, LB, expr, UB, P1);
		expr.end();
		model.add(Per1);
		Per1i.add(Per1);
	}
	//-----------------------------------//
	//--------Periorismos ΣXojmn=1-------//
	IloRangeArray orgna(env, 0);
	for (a = 0; a < amax; a++){
		IloExpr expr(env, 0);
		for (j = 0; j < jmax; j++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (j != O){
						expr += Xijmn[O][j][m][n];
					}
				}
			}
		}
```

```
		char or[60];
		sprintf_s(or, "orgna(a%d)", a);
		float LB = 1, UB = 1;
		IloRange orgn(env, LB, expr, UB, or);
		expr.end();
		model.add(orgn);
		orgna.add(orgn);
}
//---------------------------------//
//------Periorismos ΣXidmn=1------//
IloRangeArray desta(env, 0);
for (a = 0; a < amax; a++){
		IloExpr expr(env, 0);
		for (i = 0; i < imax; i++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (i != D){
						expr += Xijmn[i][D][m][n];
					}
				}
			}
		}
		char de[60];
		sprintf_s(de, "desta(a%d)", a);
		float LB = 1, UB = 1;
		IloRange dest(env, LB, expr, UB, de);
		expr.end();
		model.add(dest);
		desta.add(dest);
}
//---------------------------------------//
//------Periorismos ΣXijmn-ΣXjkmn=0------//
IloRangeArray Afiksi_Anaxorishijmn(env, 0);
for (j = 0; j < jmax; j++){
		IloExpr expr(env, 0);
		IloExpr expr1(env, 0);
		IloExpr expr2(env, 0);
		for (i = 0; i < imax; i++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (i != D  &&  j != D){
						expr1 += Xijmn[i][j][m][n];
					}
				}
			}
		}
		for (k = 0; k < kmax; k++){
			for (m = 0; m < mmax; m++){
```

```
                        for (n = 0; n < nmax; n++){
                                if (k != O  &&  j != D){
                                        expr2 += Xijmn[j][k][m][n];
                                }
                        }
                }
        }
        expr += expr2 - expr1;
        char Demand_B[60];
        sprintf_s(Demand_B, "Afiksi_Anaxorishijmn(j%d)", j);
        float LB = 0, UB = 0;
        IloRange Afiksi_Anaxorish(env, LB, expr, UB, Demand_B);
        expr.end();
        expr1.end();
        expr2.end();
        model.add(Afiksi_Anaxorish);
        Afiksi_Anaxorishijmn.add(Afiksi_Anaxorish);
}
//------------------------------//
//-----Periorismos ΣXdjmn=0-----//
IloRangeArray terma(env, 0);
for (a = 0; a < amax; a++){
        IloExpr expr(env, 0);
        for (j = 0; j < jmax; j++){
                for (m = 0; m < mmax; m++){
                        for (n = 0; n < nmax; n++){
                                if (j != D){
                                        expr += Xijmn[D][j][m][n];
                                }
                        }
                }
        }
        char te[60];
        sprintf_s(te, "terma(a%d)", a);
        float LB = 0, UB = 0;
        IloRange term(env, LB, expr, UB, te);
        expr.end();
        model.add(term);
        terma.add(term);
}
//--------------------------------------------//
//----------Periorismos Xijmn+Xjimn<=1---------//
IloRangeMatrix4x4 Mh_Epistrofhijmn(env, 0);
for (i = 0; i < imax; i++){
        IloRangeMatrix3x3 Mh_Epistrofhjmn(env, 0);
        for (j = 0; j < jmax; j++){
                IloRangeMatrix2x2 Mh_Epistrofhmn(env, 0);
                for (m = 0; m < mmax; m++){
```

```
                                IloRangeArray Mh_Epistrofhn(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += Xijmn[i][j][m][n] +
Xijmn[j][i][m][n];
                                        }
                                        char Epistr[60];
                                        sprintf_s(Epistr,
"Mi_Epistrofhijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                        float LB = -IloInfinity, UB = 1;
                                        IloRange Mh_Epistrofh(env, LB, expr, UB, Epistr);
                                        expr.end();
                                        model.add(Mh_Epistrofh);
                                        Mh_Epistrofhn.add(Mh_Epistrofh);
                                }
                                Mh_Epistrofhmn.add(Mh_Epistrofhn);
                        }
                        Mh_Epistrofhjmn.add(Mh_Epistrofhmn);
                }
                Mh_Epistrofhijmn.add(Mh_Epistrofhjmn);
        }
        //-----------------------------------------//
        //-----------Periorismos ΣXiomn=0-----------//
        IloRangeArray afeta(env, 0);
        for (a = 0; a < amax; a++){
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != O){
                                                expr += Xijmn[i][O][m][n];
                                        }
                                }
                        }
                }
                char af[60];
                sprintf_s(af, "afeta(a%d)", a);
                float LB = 0, UB = 0;
                IloRange afet(env, LB, expr, UB, af);
                expr.end();
                model.add(afet);
                afeta.add(afet);
        }
        //-------------------------------------------------//
        //-----------------Periorismoi Xronou---------------//
        //------Periorismos dijmn-(1-Xijmn)*M<=DTijmn-------//
        IloRangeMatrix4x4 Per8ijmn(env, 0);
```

```
	for (i = 0; i < imax; i++){
		IloRangeMatrix3x3 Per8jmn(env, 0);
		for (j = 0; j < jmax; j++){
			IloRangeMatrix2x2 Per8mn(env, 0);
			for (m = 0; m < mmax; m++){
				IloRangeArray Per8n(env, 0);
				for (n = 0; n < nmax; n++){
					IloExpr expr(env, 0);
					if (i != j){
						expr += (d[i][j][m][n]) - (1 -
Xijmn[i][j][m][n])*M - DTijmn[i][j][m][n];
					}
					char P8[60];
					sprintf_s(P8, "Per8ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
					float LB = -IloInfinity, UB = 0;
					IloRange Per8(env, LB, expr, UB, P8);
					expr.end();
					model.add(Per8);
					Per8n.add(Per8);
				}
				Per8mn.add(Per8n);
			}
			Per8jmn.add(Per8mn);
		}
		Per8ijmn.add(Per8jmn);
	}
	//-------------------------------------------------//
	//------Periorismos DTijmn<=dijmn+(1-Xijmn)*M------//
	IloRangeMatrix4x4 Per9ijmn(env, 0);
	for (i = 0; i < imax; i++){
		IloRangeMatrix3x3 Per9jmn(env, 0);
		for (j = 0; j < jmax; j++){
			IloRangeMatrix2x2 Per9mn(env, 0);
			for (m = 0; m < mmax; m++){
				IloRangeArray Per9n(env, 0);
				for (n = 0; n < nmax; n++){
					IloExpr expr(env, 0);
					if (i != j){
						expr += DTijmn[i][j][m][n] - (d[i][j][m][n])
- (1 - Xijmn[i][j][m][n])*M;
					}
					char P9[60];
					sprintf_s(P9, "Per9ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
					float LB = -IloInfinity, UB = 0;
					IloRange Per9(env, LB, expr, UB, P9);
					expr.end();
```

```
                                        model.add(Per9);
                                        Per9n.add(Per9);
                                }
                                Per9mn.add(Per9n);
                        }
                        Per9jmn.add(Per9mn);
                }
                Per9ijmn.add(Per9jmn);
        }
        //----------------------------------------//
        //--------Periorismos Xijmn*M<=DTijmn-------//
        IloRangeMatrix4x4 Per10ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per10jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per10mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per10n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += -(Xijmn[i][j][m][n])*M -
DTijmn[i][j][m][n];
                                        }
                                        char P10[60];
                                        sprintf_s(P10, "Per10ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per10(env, LB, expr, UB, P10);
                                        expr.end();
                                        model.add(Per10);
                                        Per10n.add(Per10);
                                }
                                Per10mn.add(Per10n);
                        }
                        Per10jmn.add(Per10mn);
                }
                Per10ijmn.add(Per10jmn);
        }
        //----------------------------------------//
        //-------Periorismos DTijmn<=Xijmn*M--------//
        IloRangeMatrix4x4 Per11ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per11jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per11mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per11n(env, 0);
```

```
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += DTijmn[i][j][m][n] -
(Xijmn[i][j][m][n])*M;
                                        }
                                        char P11[60];
                                        sprintf_s(P11, "Per11ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per11(env, LB, expr, UB, P11);
                                        expr.end();
                                        model.add(Per11);
                                        Per11n.add(Per11);
                                }
                                Per11mn.add(Per11n);
                        }
                        Per11jmn.add(Per11mn);
                }
                Per11ijmn.add(Per11jmn);
        }
        //-----------------------------------------------------------------//
        //------Periorismos DTkjcz+(2-Xikmn-Xkjcz)*M>=DTikmn+tikm*Xikmn------//
        IloRangeMatrix7x7 Per12ikjmcnz(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix6x6 Per12kjmcnz(env, 0);
                for (k = 0; k < kmax; k++){
                        IloRangeMatrix5x5 Per12jmcnz(env, 0);
                        for (j = 0; j < jmax; j++){
                                IloRangeMatrix4x4 Per12mcnz(env, 0);
                                for (m = 0; m < mmax; m++){
                                        IloRangeMatrix3x3 Per12cnz(env, 0);
                                        for (c = 0; c < cmax; c++){
                                                IloRangeMatrix2x2 Per12nz(env, 0);
                                                for (n = 0; n < nmax; n++){
                                                        IloRangeArray Per12z(env, 0);
                                                        for (z = 0; z < zmax; z++){
                                                                IloExpr expr(env, 0);
                                                                if (i != j  &&  i != k  &&  j
!= k){
                                                                        expr +=
DTijmn[i][k][m][n] + t[i][k][m] * Xijmn[i][k][m][n] - DTijmn[k][j][c][z] - (2 -
Xijmn[i][k][m][n] - Xijmn[k][j][c][z])*M;
                                                                }
                                                                char P12[60];
                                                                sprintf_s(P12,
"Per12ikjmcnz(i%d,k%d,j%d,m%d,c%d,n%d,z%d)", i, k, j, m, c, n, z);
```

```
                                                    float LB = -IloInfinity, UB =
0;
                                                    IloRange Per12(env, LB,
expr, UB, P12);
                                                    expr.end();
                                                    model.add(Per12);
                                                    Per12z.add(Per12);
                                                }
                                                Per12nz.add(Per12z);
                                            }
                                            Per12cnz.add(Per12nz);
                                        }
                                        Per12mcnz.add(Per12cnz);
                                    }
                                    Per12jmcnz.add(Per12mcnz);
                                }
                                Per12kjmcnz.add(Per12jmcnz);
                            }
                            Per12ikjmcnz.add(Per12kjmcnz);
                        }
                        //-------------------------------------------------------------//
                        //-------Periorismos DTijmn+tijm*Xijmn-(1-Xijmn)*M<=ATjimn-------//
                        IloRangeMatrix4x4 Per13ijmn(env, 0);
                        for (i = 0; i < imax; i++){
                            IloRangeMatrix3x3 Per13jmn(env, 0);
                            for (j = 0; j < jmax; j++){
                                IloRangeMatrix2x2 Per13mn(env, 0);
                                for (m = 0; m < mmax; m++){
                                    IloRangeArray Per13n(env, 0);
                                    for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                            expr += DTijmn[i][j][m][n] + t[i][j][m] *
Xijmn[i][j][m][n] - (1 - Xijmn[i][j][m][n])*M - ATjimn[j][i][m][n];
                                        }
                                        char P13[60];
                                        sprintf_s(P13, "Per13ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per13(env, LB, expr, UB, P13);
                                        expr.end();
                                        model.add(Per13);
                                        Per13n.add(Per13);
                                    }
                                    Per13mn.add(Per13n);
                                }
                                Per13jmn.add(Per13mn);
                            }
```

```
                Per13ijmn.add(Per13jmn);
        }
        //-------------------------------------------------------------//
        //------Periorismos ATjimn<=DTijmn+tijm*Xijmn+(1-Xijmn)*M------//
        IloRangeMatrix4x4 Per14ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per14jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per14mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per14n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += ATjimn[j][i][m][n] -
DTijmn[i][j][m][n] - t[i][j][m] * Xijmn[i][j][m][n] - (1 - Xijmn[i][j][m][n])*M;
                                        }
                                        char P14[60];
                                        sprintf_s(P14, "Per14ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per14(env, LB, expr, UB, P14);
                                        expr.end();
                                        model.add(Per14);
                                        Per14n.add(Per14);
                                }
                                Per14mn.add(Per14n);
                        }
                        Per14jmn.add(Per14mn);
                }
                Per14ijmn.add(Per14jmn);
        }
        //-----------------------------------------//
        //-------Periorismos -Xijmn*M<=ATjimn------//
        IloRangeMatrix4x4 Per15ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per15jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per15mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per15n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += -(Xijmn[i][j][m][n])*M -
ATjimn[j][i][m][n];
                                        }
                                        char P15[60];
```

```
                                        sprintf_s(P15, "Per15ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per15(env, LB, expr, UB, P15);
                                        expr.end();
                                        model.add(Per15);
                                        Per15n.add(Per15);
                                }
                                Per15mn.add(Per15n);
                        }
                        Per15jmn.add(Per15mn);
                }
                Per15ijmn.add(Per15jmn);
        }
        //---------------------------------------//
        //-------Periorismos ATjimn<=Xijmn*M------//
        IloRangeMatrix4x4 Per16ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per16jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per16mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per16n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += ATjimn[j][i][m][n] -
(Xijmn[i][j][m][n])*M;
                                        }
                                        char P16[60];
                                        sprintf_s(P16, "Per16ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per16(env, LB, expr, UB, P16);
                                        expr.end();
                                        model.add(Per16);
                                        Per16n.add(Per16);
                                }
                                Per16mn.add(Per16n);
                        }
                        Per16jmn.add(Per16mn);
                }
                Per16ijmn.add(Per16jmn);
        }
        //-------------------------------------//
        //-------Periorismos ATdimn<=ToA-------//
        IloRangeArray Per17a(env, 0);
        for (a = 0; a < amax; a++){
```

```
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != D){
                                                expr += ATjimn[D][i][m][n];
                                        }
                                }
                        }
                }
                char P17[60];
                sprintf_s(P17, "Per17a(a%d)", a);
                int LB = -IloInfinity, UB = ToA;
                IloRange Per17(env, LB, expr, UB, P17);
                expr.end();
                model.add(Per17);
                Per17a.add(Per17);
        }
        //----------------------------//
        //-----Ui-Uj+N*Xijmn<=N-1-----//
        IloRangeMatrix4x4 Periorismos_Subrouteijmn(env, 0);
        for (i = 1; i < imax; i++){
                IloRangeMatrix3x3 Periorismos_Subroutejmn(env, 0);
                for (j = 1; j < jmax; j++){
                        IloRangeMatrix2x2 Periorismos_Subroutemn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Periorismos_Subrouten(env, 0);
                                for (n = 0; n < nmax; n++){
                                        if (i != j){
                                                IloExpr expr(env, 0);
                                                if (i != j) expr = Ui[i] - Ui[j] +
N*Xijmn[i][j][m][n] - N;
                                                char subroute[60];
                                                sprintf_s(subroute,
"Periorismos_Subrouteijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                                int LB = -IloInfinity, UB = -1;
                                                IloRange Periorismos_Subroute(env, LB,
expr, UB, subroute);
                                                expr.end();
                                                model.add(Periorismos_Subroute);
                                        Periorismos_Subrouten.add(Periorismos_Subroute);
                                        }
                                }
                                Periorismos_Subroutemn.add(Periorismos_Subrouten);
                        }
                        Periorismos_Subroutejmn.add(Periorismos_Subroutemn);
                }
                Periorismos_Subrouteijmn.add(Periorismos_Subroutejmn);
```

```
		}
		//--------------------------------------//
		//------Antikeimeniki Sinartisi--------//
		IloExpr expr1(env);
		for (i = 0; i < imax; i++){
			for (j = 0; j < jmax; j++){
				for (m = 0; m < mmax; m++){
					for (n = 0; n < nmax; n++){
						if (i != j){
							expr1 += Xijmn[i][j][m][n] * C[i][j][m];
						}
					}
				}
			}
		}
		model.add(IloMinimize(env, expr1));
		expr1.end();
		cplex.extract(model);
		cplex.exportModel("otinanai.lp");
		cplex.solve();
		if (!cplex.solve()){
			env.error() <<"Faild to optimize LP."<< endl;
			throw(-1);
		}
		env.out() <<"Solution status="<< cplex.getStatus() << endl;
		env.out() <<"Solution value="<< cplex.getObjValue() << endl;
for (i = 0; i < imax; i++){
			for (j = 0; j < jmax; j++){
				for (m = 0; m < mmax; m++){
					for (n = 0; n < nmax; n++){
						if (i != j){
							int g = cplex.getValue(Xijmn[i][j][m][n]);
							if (g != 0) cout <<"Xijmn"<<"("<< i
<<","<< j <<","<< m <<","<< n <<")"<<"="<< g << endl;
						}
					}
				}
			}
		}

		for (j = 0; j < jmax; j++){
			for (i = 0; i < imax; i++){
				for (m = 0; m < mmax; m++){
					for (n = 0; n < nmax; n++){
						if (i != j){
							float g =
cplex.getValue(ATjimn[j][i][m][n]);
```

```
                                                if (g != 0) cout <<"ATjimn"<<"("<< j
<<","<< i <<","<< m <<","<< n <<")"<<"="<< g << endl;
                                                }
                                        }
                                }
                        }
                }

                for (i = 0; i < imax; i++){
                        for (j = 0; j < jmax; j++){
                                for (m = 0; m < mmax; m++){
                                        for (n = 0; n < nmax; n++){
                                                if (i != j){
                                                float g =
cplex.getValue(DTijmn[i][j][m][n]);
                                                if (g != 0) cout <<"DTijmn"<<"("<< i
<<","<< j <<","<< m <<","<< n <<")"<<"="<< g << endl;
                                                }
                                        }
                                }
                        }
                }
        }
        catch (IloException& e){
                cerr <<"concert exception caught:"<< e << endl;
        }
        catch (...){
                cerr <<"Unknown exception caught"<< endl;
        }
        env.end();
        return 0;
        } //End main
```

ΠΑΡΑΡΤΗΜΑΒ

Κώδικας C++, Μοντέλο M1 για GTFS Δεδομένα

```
#include"stdafx.h"
#include<ilcplex/ilocplex.h>

#include<string>
#include<iterator>
#include<iostream>
#include<fstream>
#include<istream>
#include<sstream>
#include<vector>
usingnamespace std;

//generic variables and constants
int i;
int j;
int k;
int c;
int m;
int n;
int z;
int a;
constint imax = 20;
constint jmax = 20;
constint kmax = 20;
constint cmax = 11;
constint mmax = 11;
constint nmax = 29;
constint zmax = 29;
constint amax = 1;
constint ToA = 720;
constint N = 20;
constint O = 13;
constint D = 3;
constint M = 1000000;


double C[imax][jmax][mmax];
double d[imax][jmax][mmax][nmax];
double t[imax][jmax][mmax];


//start of functions declaration, for loading data and declaring the mathematical model
```

```
classCSVRow
{
public:
std::stringconst& operator[](std::size_tindex) const
{
return m_data[index];
}
std::size_t size() const
{
return m_data.size();
}
void readNextRow(std::istream&str)
{
std::string        line;
std::getline(str, line);

std::stringstream   lineStream(line);
std::string        cell;

m_data.clear();
while (std::getline(lineStream, cell, '       '))
{
m_data.push_back(cell);
}
// This checks for a trailing comma with no data after it.
if (!lineStream && cell.empty())
{
// If there was a trailing comma then add an empty element.
m_data.push_back("");
            }
}
private:
std::vector<std::string>    m_data;
};

// You forgot this function.
// It defines how to read from a stream one row of the CSV file.
std::istream& operator>>(std::istream&str, CSVRow&data)
{
data.readNextRow(str);
returnstr;
}

classCSVIterator
{
public:

typedef std::input_iterator_tagiterator_category;
```

```
typedefCSVRowvalue_type;
typedef std::size_tdifference_type;
typedefCSVRow*                  pointer;
typedefCSVRow&reference;

CSVIterator(std::istream&str) :m_str(str.good() ? &str : NULL) { ++(*this); }
CSVIterator() :m_str(NULL) {}

// Pre Increment
CSVIterator& operator++()             { if (m_str) { if (!((*m_str) >> m_row)){ m_str = NULL;
} }return *this; }
// Post increment
CSVIterator operator++(int)           { CSVIterator    tmp(*this); ++(*this); return tmp; }
CSVRowconst& operator*()   const      { return m_row; }
CSVRowconst* operator->()  const      { return&m_row; }

bool operator==(CSVIteratorconst&rhs) { return ((this == &rhs) || ((this->m_str == NULL)
&& (rhs.m_str == NULL))); }
bool operator!=(CSVIteratorconst&rhs) { return !((*this) == rhs); }
private:
std::istream*       m_str;
CSVRow              m_row;
};

//we declare the data of the problem, declared under the constants of as, ae, E, as, S and initial
int load_data_departures_1()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio

for (m = 0; m < mmax; m++){
        for (n = 0; n < nmax; n++){
                std::ifstream   file_1("gtfsvolos/data_departures/basic_JP_departures - Mode
" + std::to_string(m) + " Itineraries " + std::to_string(n) + ".csv");
                int counter = 0;
                for (CSVIterator loop(file_1); loop != CSVIterator(); ++loop){
                        for (i = 0; i < imax; i++){
                                d[counter][i][m][n] = std::stod((*loop)[i]);
                                }
                                counter += 1;
                        }
                }
        }

        return 0;
}

int load_data_travel_cost_1()
{
```

```
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio

for (m = 0; m < mmax; m++){
	std::ifstream file_2("gtfsvolos/data_travel_cost/basic_JP_travel_cost - Mode of transport " + std::to_string(m) + ".csv");
	int counter = 0;
	for (CSVIterator loop(file_2); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			C[counter][i][m] = std::stod((*loop)[i]);
			}
			counter += 1;
		}
	}

	return 0;
}

int load_data_travel_time_1()
{
//edo se auti ti grammi vazoume to meros sto opoio einai to arxeio

for (m = 0; m < mmax; m++){
	std::ifstream file_3("gtfsvolos/data_travel_time/basic_JP_travel_time - Mode of transport " + std::to_string(m) + ".csv");
	int counter = 0;
	for (CSVIterator loop(file_3); loop != CSVIterator(); ++loop){
		for (i = 0; i < imax; i++){
			t[counter][i][m] = std::stod((*loop)[i]);
			}
			counter += 1;
		}
	}

	return 0;
}

int main(intargc, char **argv)
{

int status;

status = load_data_departures_1();
status = load_data_travel_cost_1();
status = load_data_travel_time_1();


IloEnv env;
	try{
```

```
IloModel model(env);
typedefIloArray<IloNumArray>IloNumMatrix2x2;
typedefIloArray<IloNumMatrix2x2>IloNumMatrix3x3;
typedefIloArray<IloNumMatrix3x3>IloNumMatrix4x4;
typedefIloArray<IloNumMatrix4x4>IloNumMatrix5x5;
typedefIloArray<IloNumMatrix5x5>IloNumMatrix6x6;
typedefIloArray<IloNumMatrix6x6>IloNumMatrix7x7;
typedefIloArray<IloNumVarArray>IloNumVarMatrix2x2;
typedefIloArray<IloNumVarMatrix2x2>IloNumVarMatrix3x3;
typedefIloArray<IloNumVarMatrix3x3>IloNumVarMatrix4x4;
typedefIloArray<IloNumVarMatrix4x4>IloNumVarMatrix5x5;
typedefIloArray<IloNumVarMatrix5x5>IloNumVarMatrix6x6;
typedefIloArray<IloNumVarMatrix6x6>IloNumVarMatrix7x7;
typedefIloArray<IloRangeArray>IloRangeMatrix2x2;
typedefIloArray<IloRangeMatrix2x2>IloRangeMatrix3x3;
typedefIloArray<IloRangeMatrix3x3>IloRangeMatrix4x4;
typedefIloArray<IloRangeMatrix4x4>IloRangeMatrix5x5;
typedefIloArray<IloRangeMatrix5x5>IloRangeMatrix6x6;
typedefIloArray<IloRangeMatrix6x6>IloRangeMatrix7x7;


IloCplex cplex(env);
//---------------------------------//
//------Metavliti Apofasis X------//
IloNumVarMatrix4x4 Xijmn(env, 0);
for (i = 0; i < imax; i++){
        IloNumVarMatrix3x3 Xjmn(env, 0);
        for (j = 0; j < jmax; j++){
                IloNumVarMatrix2x2 Xmn(env, 0);
                for (m = 0; m < mmax; m++){
                        IloNumVarArray Xn(env, 0);
                        for (n = 0; n < nmax; n++){
                                char Metakinisi[70];
                                sprintf_s(Metakinisi, "Xijmn(i%d,j%d,m%d,n%d)",
i, j, m, n);
                        IloNumVar X(env, 0, 1, ILOINT, Metakinisi);
                        Xn.add(X);
                        }
                        Xmn.add(Xn);
                }
                Xjmn.add(Xmn);
        }
        Xijmn.add(Xjmn);
}
//---------------------------------//
//------Metavliti Apofasis AT-------//
IloNumVarMatrix4x4 ATjimn(env, 0);
for (j = 0; j < jmax; j++){
```

```
                IloNumVarMatrix3x3 ATimn(env, 0);
                for (i = 0; i < imax; i++){
                        IloNumVarMatrix2x2 ATmn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloNumVarArray ATn(env, 0);
                                for (n = 0; n < nmax; n++){
                                        char XronAna[70];
                                        sprintf_s(XronAna, "ATjimn(j%d,i%d,m%d,n%d)",
j, i, m, n);
                                                IloNumVar AT(env, 0, 1440, ILOFLOAT,
XronAna);
                                        ATn.add(AT);
                                }
                                ATmn.add(ATn);
                        }
                        ATimn.add(ATmn);
                }
                ATjimn.add(ATimn);
        }
        //---------------------------------//
        //------Metavliti Apofashs DT------//
        IloNumVarMatrix4x4 DTijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloNumVarMatrix3x3 DTjmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloNumVarMatrix2x2 DTmn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloNumVarArray DTn(env, 0);
                                for (n = 0; n < nmax; n++){
                                        char XronAna[70];
                                        sprintf_s(XronAna, "DTijmn(i%d,j%d,m%d,n%d)",
i, j, m, n);
                                        IloNumVar DT(env, 0, 1440, ILOFLOAT,
XronAna);
                                        DTn.add(DT);
                                }
                                DTmn.add(DTn);
                        }
                        DTjmn.add(DTmn);
                }
                DTijmn.add(DTjmn);
        }
        //---------------------------------//
        //------Metavliti U Sub Tour------//
        IloNumVarArray Ui(env, 0);
        for (i = 0; i < imax; i++){
                char Seira[70];
                sprintf_s(Seira, "Ui(i%d)", i);
```

```
		IloNumVar U(env, 1, N, ILOFLOAT, Seira);
		Ui.add(U);
	}
	//-----------------------------------------------------//
	//---------------------Periorismoi-------------------//
	//----------------Periorismos ΣXijmn<=1-------------//
	IloRangeArray Per1i(env, 0);
	for (i = 0; i < imax; i++){
		IloExpr expr(env, 0);
		for (j = 0; j < jmax; j++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (d[i][j][m][n] != 5000){
						if (i != j){
							expr += Xijmn[i][j][m][n];
						}
					}
				}
			}
		}
		char P1[60];
		sprintf_s(P1, "Per1i(i%d)", i);
		float LB = -IloInfinity, UB = 1;
		IloRange Per1(env, LB, expr, UB, P1);
		expr.end();
		model.add(Per1);
		Per1i.add(Per1);
	}
	//-----------------------------------//
	//--------Periorismos ΣXojmn=1-------//
	IloRangeArray orgna(env, 0);
	for (a = 0; a < amax; a++){
		IloExpr expr(env, 0);
		for (j = 0; j < jmax; j++){
			for (m = 0; m < mmax; m++){
				for (n = 0; n < nmax; n++){
					if (d[O][j][m][n] != 5000){
						if (j != O){
							expr += Xijmn[O][j][m][n];
						}
					}
				}
			}
		}
		char or[60];
		sprintf_s(or, "orgna(a%d)", a);
		float LB = 1, UB = 1;
		IloRange orgn(env, LB, expr, UB, or);
```

```
            expr.end();
            model.add(orgn);
            orgna.add(orgn);
}
//--------------------------------//
//------Periorismos ΣXidmn=1------//
IloRangeArray desta(env, 0);
for (a = 0; a < amax; a++){
            IloExpr expr(env, 0);
            for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                    for (n = 0; n < nmax; n++){
                                                if (d[i][D][m][n] != 5000){
                                                            if (i != D){
                                                                        expr += Xijmn[i][D][m][n];
                                                            }
                                                }
                                    }
                        }
            }
            char de[60];
            sprintf_s(de, "desta(a%d)", a);
            float LB = 1, UB = 1;
            IloRange dest(env, LB, expr, UB, de);
            expr.end();
            model.add(dest);
            desta.add(dest);
}
//---------------------------------------//
//------Periorismos ΣXijmn-ΣXjkmn=0------//
IloRangeArray Afiksi_Anaxorishijmn(env, 0);
for (j = 0; j < jmax; j++){
            IloExpr expr(env, 0);
            IloExpr expr1(env, 0);
            IloExpr expr2(env, 0);
            for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                    for (n = 0; n < nmax; n++){
                                                if (i != D  &&  j != D){
                                                            expr1 += Xijmn[i][j][m][n];
                                                }
                                    }
                        }
            }
            for (k = 0; k < kmax; k++){
                        for (m = 0; m < mmax; m++){
                                    for (n = 0; n < nmax; n++){
                                                if (k != O  &&  j != D){
```

```
                                        expr2 += Xijmn[j][k][m][n];
                                    }
                                }
                            }
                        }
                        expr += expr2 - expr1;
                        char Demand_B[60];
                        sprintf_s(Demand_B, "Afiksi_Anaxorishijmn(j%d)", j);
                        float LB = 0, UB = 0;
                        IloRange Afiksi_Anaxorish(env, LB, expr, UB, Demand_B);
                        expr.end();
                        expr1.end();
                        expr2.end();
                        model.add(Afiksi_Anaxorish);
                        Afiksi_Anaxorishijmn.add(Afiksi_Anaxorish);
                }
                //------------------------------//
                //-----Periorismos ΣXdjmn=0-----//
                IloRangeArray terma(env, 0);
                for (a = 0; a < amax; a++){
                        IloExpr expr(env, 0);
                        for (j = 0; j < jmax; j++){
                                for (m = 0; m < mmax; m++){
                                        for (n = 0; n < nmax; n++){
                                                if (d[D][j][m][n] != 5000){
                                                        if (j != D){
                                                                expr += Xijmn[D][j][m][n];
                                                        }
                                                }
                                        }
                                }
                        }
                        char te[60];
                        sprintf_s(te, "terma(a%d)", a);
                        float LB = 0, UB = 0;
                        IloRange term(env, LB, expr, UB, te);
                        expr.end();
                        model.add(term);
                        terma.add(term);
                }
                //-------------------------------------------//
                //----------Periorismos Xijmn+Xjimn<=1---------//
                IloRangeMatrix4x4 Mh_Epistrofhijmn(env, 0);
                for (i = 0; i < imax; i++){
                        IloRangeMatrix3x3 Mh_Epistrofhjmn(env, 0);
                        for (j = 0; j < jmax; j++){
                                IloRangeMatrix2x2 Mh_Epistrofhmn(env, 0);
                                for (m = 0; m < mmax; m++){
```

```
                                IloRangeArray Mh_Epistrofhn(env, 0);
                                for (n = 0; n < nmax; n++){
                                        if ((d[i][j][m][n] != 5000) && (d[j][i][m][n] !=
5000)){
                                                IloExpr expr(env, 0);
                                                if (i != j){
                                                        expr += Xijmn[i][j][m][n] +
Xijmn[j][i][m][n];
                                                }
                                                char Epistr[60];
                                                sprintf_s(Epistr,
"Mi_Epistrofhijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                                float LB = -IloInfinity, UB = 1;
                                                IloRange Mh_Epistrofh(env, LB, expr, UB,
Epistr);
                                                expr.end();
                                                model.add(Mh_Epistrofh);
                                                Mh_Epistrofhn.add(Mh_Epistrofh);
                                        }
                                }
                                Mh_Epistrofhmn.add(Mh_Epistrofhn);
                        }
                        Mh_Epistrofhjmn.add(Mh_Epistrofhmn);
                }
                Mh_Epistrofhijmn.add(Mh_Epistrofhjmn);
        }
        //-----------------------------------------//
        //----------Periorismos ΣXiomn=0----------//
        IloRangeArray afeta(env, 0);
        for (a = 0; a < amax; a++){
                IloExpr expr(env, 0);
                for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (d[i][O][m][n] != 5000){
                                                if (i != O){
                                                        expr += Xijmn[i][O][m][n];
                                                }
                                        }
                                }
                        }
                }
                char af[60];
                sprintf_s(af, "afeta(a%d)", a);
                float LB = 0, UB = 0;
                IloRange afet(env, LB, expr, UB, af);
                expr.end();
                model.add(afet);
```

```
                afeta.add(afet);
        }
        //-----------------------------------------------------//
        //------------------Periorismoi Xronou----------------//
        //------Periorismos dijmn-(1-Xijmn)*M<=DTijmn-------//
        IloRangeMatrix4x4 Per8ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per8jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per8mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per8n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += (d[i][j][m][n]) - (1 -
Xijmn[i][j][m][n])*M - DTijmn[i][j][m][n];
                                        }
                                        char P8[60];
                                        sprintf_s(P8, "Per8ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per8(env, LB, expr, UB, P8);
                                        expr.end();
                                        model.add(Per8);
                                        Per8n.add(Per8);
                                }
                                Per8mn.add(Per8n);
                        }
                        Per8jmn.add(Per8mn);
                }
                Per8ijmn.add(Per8jmn);
        }
        //-----------------------------------------------------//
        //------Periorismos DTijmn<=dijmn+(1-Xijmn)*M------//
        IloRangeMatrix4x4 Per9ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per9jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per9mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per9n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += DTijmn[i][j][m][n] - (d[i][j][m][n])
- (1 - Xijmn[i][j][m][n])*M;
                                        }
```

```
                                        char P9[60];
                                        sprintf_s(P9, "Per9ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per9(env, LB, expr, UB, P9);
                                        expr.end();
                                        model.add(Per9);
                                        Per9n.add(Per9);
                                }
                                Per9mn.add(Per9n);
                        }
                        Per9jmn.add(Per9mn);
                }
                Per9ijmn.add(Per9jmn);
        }
        //----------------------------------------//
        //--------Periorismos Xijmn*M<=DTijmn-------//
        IloRangeMatrix4x4 Per10ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per10jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per10mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per10n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        if (d[i][j][m][n] != 5000){
                                                IloExpr expr(env, 0);
                                                if (i != j){
                                                        expr += -(Xijmn[i][j][m][n])*M -
DTijmn[i][j][m][n];
                                                }
                                                char P10[60];
                                                sprintf_s(P10,
"Per10ijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                                float LB = -IloInfinity, UB = 0;
                                                IloRange Per10(env, LB, expr, UB, P10);
                                                expr.end();
                                                model.add(Per10);
                                                Per10n.add(Per10);
                                        }
                                }
                                Per10mn.add(Per10n);
                        }
                        Per10jmn.add(Per10mn);
                }
                Per10ijmn.add(Per10jmn);
        }
        //----------------------------------------//
```

```
		//-------Periorismos DTijmn<=Xijmn*M--------//
		IloRangeMatrix4x4 Per11ijmn(env, 0);
		for (i = 0; i < imax; i++){
			IloRangeMatrix3x3 Per11jmn(env, 0);
			for (j = 0; j < jmax; j++){
				IloRangeMatrix2x2 Per11mn(env, 0);
				for (m = 0; m < mmax; m++){
					IloRangeArray Per11n(env, 0);
					for (n = 0; n < nmax; n++){
						if (d[i][j][m][n] != 5000){
							IloExpr expr(env, 0);
							if (i != j){
								expr += DTijmn[i][j][m][n] -
(Xijmn[i][j][m][n])*M;
							}
							char P11[60];
							sprintf_s(P11,
"Per11ijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
							float LB = -IloInfinity, UB = 0;
							IloRange Per11(env, LB, expr, UB, P11);
							expr.end();
							model.add(Per11);
							Per11n.add(Per11);
						}
					}
					Per11mn.add(Per11n);
				}
				Per11jmn.add(Per11mn);
			}
			Per11ijmn.add(Per11jmn);
		}
		//-----------------------------------------------------------------//
		//------Periorismos DTkjcz+(2-Xikmn-Xkjcz)*M>=DTikmn+tikm*Xikmn------//
		IloRangeMatrix7x7 Per12ikjmcnz(env, 0);
		for (i = 0; i < imax; i++){
			IloRangeMatrix6x6 Per12kjmcnz(env, 0);
			for (k = 0; k < kmax; k++){
				IloRangeMatrix5x5 Per12jmcnz(env, 0);
				for (j = 0; j < jmax; j++){
					IloRangeMatrix4x4 Per12mcnz(env, 0);
					for (m = 0; m < mmax; m++){
						IloRangeMatrix3x3 Per12cnz(env, 0);
						for (c = 0; c < cmax; c++){
							IloRangeMatrix2x2 Per12nz(env, 0);
							for (n = 0; n < nmax; n++){
								IloRangeArray Per12z(env, 0);
								for (z = 0; z < zmax; z++){
```

```
                                                                    if ((d[i][k][m][n] != 5000)
&& (d[k][j][c][z] != 5000)){
                                                                        IloExpr expr(env,
0);
                                                                        if (i != j  &&  i != k
&&  j != k){
                                                                            expr +=
DTijmn[i][k][m][n] + t[i][k][m] * Xijmn[i][k][m][n] - DTijmn[k][j][c][z] - (2 -
Xijmn[i][k][m][n] - Xijmn[k][j][c][z])*M;
                                                                        }
                                                                        char P12[60];
                                                                        sprintf_s(P12,
"Per12ikjmcnz(i%d,k%d,j%d,m%d,c%d,n%d,z%d)", i, k, j, m, c, n, z);
                                                                        float LB = -
IloInfinity, UB = 0;
                                                                        IloRange Per12(env,
LB, expr, UB, P12);
                                                                        expr.end();
                                                                        model.add(Per12);
                                                                        Per12z.add(Per12);
                                                                    }
                                                                }
                                                                Per12nz.add(Per12z);
                                                            }
                                                            Per12cnz.add(Per12nz);
                                                        }
                                                        Per12mcnz.add(Per12cnz);
                                                    }
                                                    Per12jmcnz.add(Per12mcnz);
                                                }
                                                Per12kjmcnz.add(Per12jmcnz);
                                            }
                                            Per12ikjmcnz.add(Per12kjmcnz);
                                        }
                                        //-------------------------------------------------------------//
                                        //-------Periorismos DTijmn+tijm*Xijmn-(1-Xijmn)*M<=ATjimn-------//
                                        IloRangeMatrix4x4 Per13ijmn(env, 0);
                                        for (i = 0; i < imax; i++){
                                            IloRangeMatrix3x3 Per13jmn(env, 0);
                                            for (j = 0; j < jmax; j++){
                                                IloRangeMatrix2x2 Per13mn(env, 0);
                                                for (m = 0; m < mmax; m++){
                                                    IloRangeArray Per13n(env, 0);
                                                    for (n = 0; n < nmax; n++){
                                                        IloExpr expr(env, 0);
                                                        if (i != j){
                                                            expr += DTijmn[i][j][m][n] + t[i][j][m] *
Xijmn[i][j][m][n] - (1 - Xijmn[i][j][m][n])*M - ATjimn[j][i][m][n];
```

```
                                        }
                                        char P13[60];
                                        sprintf_s(P13, "Per13ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per13(env, LB, expr, UB, P13);
                                        expr.end();
                                        model.add(Per13);
                                        Per13n.add(Per13);
                                }
                                Per13mn.add(Per13n);
                        }
                        Per13jmn.add(Per13mn);
                }
                Per13ijmn.add(Per13jmn);
        }
        //-------------------------------------------------------------//
        //------Periorismos ATjimn<=DTijmn+tijm*Xijmn+(1-Xijmn)*M------//
        IloRangeMatrix4x4 Per14ijmn(env, 0);
        for (i = 0; i < imax; i++){
                IloRangeMatrix3x3 Per14jmn(env, 0);
                for (j = 0; j < jmax; j++){
                        IloRangeMatrix2x2 Per14mn(env, 0);
                        for (m = 0; m < mmax; m++){
                                IloRangeArray Per14n(env, 0);
                                for (n = 0; n < nmax; n++){
                                        IloExpr expr(env, 0);
                                        if (i != j){
                                                expr += ATjimn[j][i][m][n] -
DTijmn[i][j][m][n] - t[i][j][m] * Xijmn[i][j][m][n] - (1 - Xijmn[i][j][m][n])*M;
                                        }
                                        char P14[60];
                                        sprintf_s(P14, "Per14ijmn(i%d,j%d,m%d,n%d)", i, j,
m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per14(env, LB, expr, UB, P14);
                                        expr.end();
                                        model.add(Per14);
                                        Per14n.add(Per14);
                                }
                                Per14mn.add(Per14n);
                        }
                        Per14jmn.add(Per14mn);
                }
                Per14ijmn.add(Per14jmn);
        }
        //-----------------------------------------//
        //-------Periorismos -Xijmn*M<=ATjimn------//
```

```
	IloRangeMatrix4x4 Per15ijmn(env, 0);
	for (i = 0; i < imax; i++){
		IloRangeMatrix3x3 Per15jmn(env, 0);
		for (j = 0; j < jmax; j++){
			IloRangeMatrix2x2 Per15mn(env, 0);
			for (m = 0; m < mmax; m++){
				IloRangeArray Per15n(env, 0);
				for (n = 0; n < nmax; n++){
					if (d[i][j][m][n] != 5000){
						IloExpr expr(env, 0);
						if (i != j){
							expr += -(Xijmn[i][j][m][n])*M -
ATjimn[j][i][m][n];
						}
						char P15[60];
						sprintf_s(P15,
"Per15ijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
						float LB = -IloInfinity, UB = 0;
						IloRange Per15(env, LB, expr, UB, P15);
						expr.end();
						model.add(Per15);
						Per15n.add(Per15);
					}
				}
				Per15mn.add(Per15n);
			}
			Per15jmn.add(Per15mn);
		}
		Per15ijmn.add(Per15jmn);
	}
	//---------------------------------------//
	//-------Periorismos ATjimn<=Xijmn*M------//
	IloRangeMatrix4x4 Per16ijmn(env, 0);
	for (i = 0; i < imax; i++){
		IloRangeMatrix3x3 Per16jmn(env, 0);
		for (j = 0; j < jmax; j++){
			IloRangeMatrix2x2 Per16mn(env, 0);
			for (m = 0; m < mmax; m++){
				IloRangeArray Per16n(env, 0);
				for (n = 0; n < nmax; n++){
					if (d[i][j][m][n] != 5000){
						IloExpr expr(env, 0);
						if (i != j){
							expr += ATjimn[j][i][m][n] -
(Xijmn[i][j][m][n])*M;
						}
						char P16[60];
```

```
                                        sprintf_s(P16,
"Per16ijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                        float LB = -IloInfinity, UB = 0;
                                        IloRange Per16(env, LB, expr, UB, P16);
                                        expr.end();
                                        model.add(Per16);
                                        Per16n.add(Per16);
                                    }
                                }
                                Per16mn.add(Per16n);
                            }
                            Per16jmn.add(Per16mn);
                        }
                        Per16ijmn.add(Per16jmn);
                    }
                    //-------------------------------------//
                    //-------Periorismos ATdimn<=ToA-------//
                    IloRangeArray Per17a(env, 0);
                    for (a = 0; a < amax; a++){
                        IloExpr expr(env, 0);
                        for (i = 0; i < imax; i++){
                            for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                    if (i != D){
                                        expr += ATjimn[D][i][m][n];
                                    }
                                }
                            }
                        }
                        char P17[60];
                        sprintf_s(P17, "Per17a(a%d)", a);
                        int LB = -IloInfinity, UB = ToA;
                        IloRange Per17(env, LB, expr, UB, P17);
                        expr.end();
                        model.add(Per17);
                        Per17a.add(Per17);
                    }
                    //----------------------------//
                    //-----Ui-Uj+N*Xijmn<=N-1-----//
                    IloRangeMatrix4x4 Periorismos_Subrouteijmn(env, 0);
                    for (i = 1; i < imax; i++){
                        IloRangeMatrix3x3 Periorismos_Subroutejmn(env, 0);
                        for (j = 1; j < jmax; j++){
                            IloRangeMatrix2x2 Periorismos_Subroutemn(env, 0);
                            for (m = 0; m < mmax; m++){
                                IloRangeArray Periorismos_Subrouten(env, 0);
                                for (n = 0; n < nmax; n++){
                                    if (d[i][j][m][n] != 5000){
```

```
                                        if (i != j){
                                                IloExpr expr(env, 0);
                                                if (i != j) expr = Ui[i] - Ui[j] +
N*Xijmn[i][j][m][n] - N;
                                                char subroute[60];
                                                sprintf_s(subroute,
"Periorismos_Subrouteijmn(i%d,j%d,m%d,n%d)", i, j, m, n);
                                                int LB = -IloInfinity, UB = -1;
                                                IloRange Periorismos_Subroute(env,
LB, expr, UB, subroute);
                                                expr.end();
                                                model.add(Periorismos_Subroute);
                                        Periorismos_Subrouten.add(Periorismos_Subroute);
                                                }
                                        }
                                }
                                Periorismos_Subroutemn.add(Periorismos_Subrouten);
                        }
                        Periorismos_Subroutejmn.add(Periorismos_Subroutemn);
                }
                Periorismos_Subrouteijmn.add(Periorismos_Subroutejmn);
        }
        //---------------------------------------//
        //------Antikeimeniki Sinartisi--------//
        IloExpr expr1(env);
        for (i = 0; i < imax; i++){
                for (j = 0; j < jmax; j++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != j){
                                                expr1 += Xijmn[i][j][m][n] * C[i][j][m];
                                        }
                                }
                        }
                }
        }
        model.add(IloMinimize(env, expr1));
        expr1.end();
        cplex.extract(model);
        cplex.exportModel("otinanai.lp");
        cplex.solve();
        if (!cplex.solve()){
                env.error() <<"Faild to optimize LP."<< endl;
                throw(-1);
        }
        env.out() <<"Solution status="<< cplex.getStatus() << endl;
        env.out() <<"Solution value="<< cplex.getObjValue() << endl;
        for (i = 0; i < imax; i++){
```

```
                for (j = 0; j < jmax; j++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != j){
                                                int g = cplex.getValue(Xijmn[i][j][m][n]);
                                                if (g != 0) cout <<"Xijmn"<<"("<< i
<<","<< j <<","<< m <<","<< n <<")"<<"="<< g << endl;
                                        }
                                }
                        }
                }
        }

        for (j = 0; j < jmax; j++){
                for (i = 0; i < imax; i++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != j){
                                                float g =
cplex.getValue(ATjimn[j][i][m][n]);
                                                if (g != 0) cout <<"ATjimn"<<"("<< j
<<","<< i <<","<< m <<","<< n <<")"<<"="<< g << endl;
                                        }
                                }
                        }
                }
        }

        for (i = 0; i < imax; i++){
                for (j = 0; j < jmax; j++){
                        for (m = 0; m < mmax; m++){
                                for (n = 0; n < nmax; n++){
                                        if (i != j){
                                                float g =
cplex.getValue(DTijmn[i][j][m][n]);
                                                if (g != 0) cout <<"DTijmn"<<"("<< i
<<","<< j <<","<< m <<","<< n <<")"<<"="<< g << endl;
                                        }
                                }
                        }
                }
        }
}
catch (IloException& e){
        cerr <<"concert exception caught:"<< e << endl;
}
catch (...){
        cerr <<"Unknown exception caught"<< endl;
```

```
}
env.end();
return 0;
} //End main
```