ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

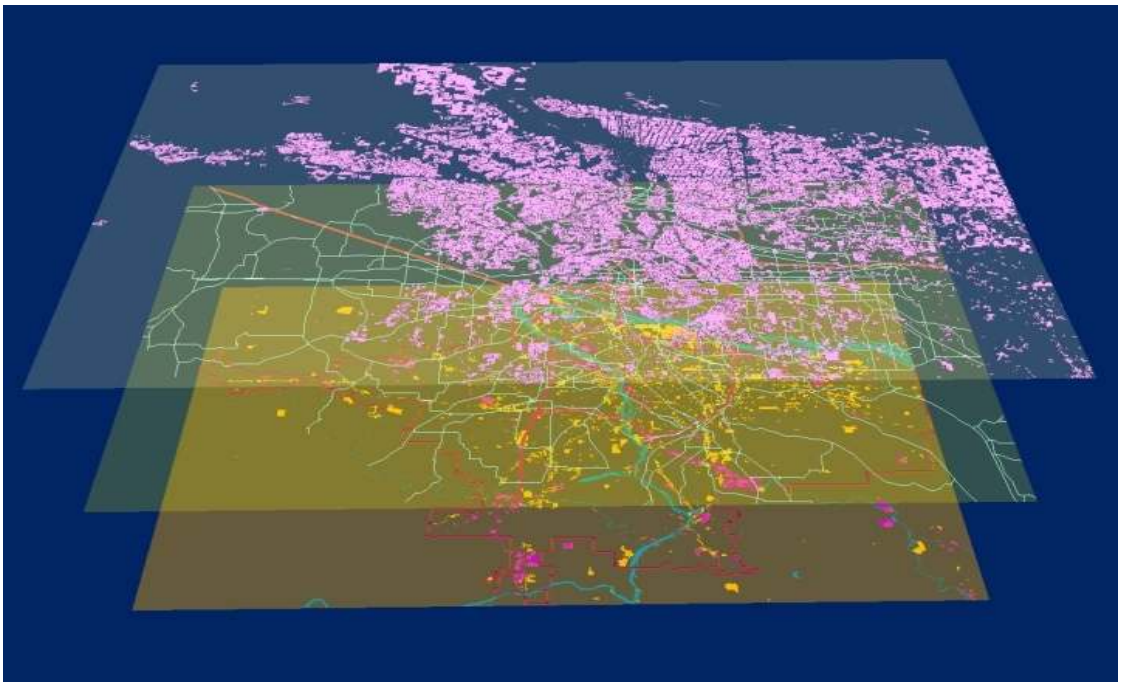
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ ΜΕΤΑΦΟΡΩΝ ΚΑΙ ΣΥΓΚΟΙΝΩΝΙΑΚΗΣ ΥΠΟΔΟΜΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑΤΑ ΠΛΟΗΓΗΣΗΣ ΣΤΑ ΣΥΓΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ

Επιβλέπων : Αντώνιος Σταθόπουλος, Καθηγητής Ε.Μ.Π.



ΓΚΙΟΤΣΑΛΙΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2010

Αφιερωμένο

στο θείο μου Σωτήρη Γκιοτσαλίτη

και στον καθηγητή μου στα μαθηματικά Φίλιππα Λιανό

ΣΥΝΟΨΗ

Τίτλος: «Συστήματα πλοήγησης στα συγκοινωνιακά δίκτυα»

Έτος Εκπόνησης: 2010

Επιβλέπων: Σταθόπουλος Α., Καθηγητής

Τις τελευταίες δεκαετίες έχει αυξηθεί έντονα η ζήτηση για μεταφορές. Ωστόσο, πολλές αστικές περιοχές δεν έχουν σχεδιαστεί ώστε να εξυπηρετήσουν αυτή τη ζήτηση και λόγω της πυκνής τους δόμησης δεν υπάρχουν πολλά περιθώρια για τη βελτίωση της υποδομής τους. Έτσι πολλά μητροπολιτικά κέντρα αντιμετωπίζουν έντονα προβλήματα κυκλοφοριακής συμφόρησης και ως συνέπεια καταναλώνονται κάθε χρόνο άσκοπα σημαντικοί οικονομικοί και ενεργειακοί πόροι, επιβαρύνοντας την οικονομία και το περιβάλλον. Τα τελευταία χρόνια έχουν γίνει πολλές προσπάθειες για την αντιμετώπιση αυτού του προβλήματος με χρήση νέων τεχνολογιών από τον τομέα της πληροφορικής και των τηλεπικοινωνιών, στοχεύοντας στην πιο αποδοτική διαχείριση της διαθέσιμης κυκλοφοριακής υποδομής. Μία από αυτές τις τεχνολογίες είναι και τα συστήματα πλοήγησης που χρησιμοποιούνται για την καθοδήγηση των μεταφορικών μέσων. Στην εργασία αυτή διερευνάται η χρησιμότητα των συστημάτων πλοήγησης στη μείωση του χρόνου μετακινήσεων και τον περιορισμό των εκπομπών αερίων που ευθύνονται για το φαινόμενο του θερμοκηπίου. Επίσης διερευνώνται πιθανά οφέλη που μπορούν να προκύψουν σχετικά με την αντιμετώπιση του προβλήματος της κυκλοφοριακής συμφόρησης μέσω της μαζικής χρήσης τους.

ABSTRACT

Title: «Navigation systems in transportation networks»

Year of execution: 2010

Supervisor: Stathopoulos A., Professor

Over the past decades the demand for transport has significantly increased. However, many urban areas are not designed to accommodate this demand and because of the dense building there is not much room to improve their infrastructure. So many metropolitan areas are facing intense problems of traffic congestion and as a result significant financial and energy resources are wasted unnecessarily every year burdening the economy and the environment. In recent years, there have been many attempts to confront this problem by using new technologies in the field of computer science and telecommunications, aiming at more efficient management of available road infrastructure. One of these technologies are the navigation systems which are used to guide vehicles. This paper investigates the use of navigation systems to reduce travel time and greenhouse gases which are responsible for global warming. It also explores potential benefits that may arise on tackling traffic congestion through mass public use.

ΠΡΟΛΟΓΟΣ

Τις τελευταίες δεκαετίες έχει αυξηθεί σημαντικά ο πληθυσμός των ανθρώπων που συγκεντρώνονται σε αστικά κέντρα. Το γεγονός αυτό έχει παρασύρει και τη ζήτηση για μεταφορές. Ωστόσο, η δομή των αστικών κέντρων εμποδίζει τις προσπάθειες για βελτίωση της συγκοινωνιακής υποδομής και ως συνέπεια προκαλούνται έντονα προβλήματα κυκλοφοριακής συμφόρησης. Οι διοικητικές αρχές μεγάλων μητροπολιτικών περιοχών έχουν αντιληφθεί ότι η κυκλοφοριακή συμφόρηση αποτελεί τροχοπέδη για την ανάπτυξη και έχει αρνητικές επιπτώσεις στο περιβάλλον. Για το λόγο αυτό έχουν διεξαχθεί έρευνες προς αυτή την κατεύθυνση, οι οποίες δυστυχώς δείχνουν ότι στο μέλλον το πρόβλημα θα επιδεινωθεί και απαιτούνται δραστικά μέτρα για την αντιμετώπισή του.

Ως κάτοικος μίας αστικής περιοχής βιώνω καθημερινά το πρόβλημα αυτό, το οποίο με επηρεάζει έντονα καθώς διαθέτω σημαντικό χρόνο από την ημέρα μου για μεταφορές και δαπανώ άσκοπα χρηματικούς πόρους επιβαρύνοντας ταυτόχρονα και το περιβάλλον. Το γεγονός αυτό με οδήγησε από τα πρώτα έτη της φοίτησής μου στην αναζήτηση τρόπων ώστε να βελτιωθεί αυτή η κατάσταση. Επειδή όμως δε μπορεί να συμβούν ριζικές αλλαγές στη διαθέσιμη μεταφορική υποδομή αντιλήφθηκα γρήγορα ότι η ένταση του προβλήματος μπορεί να μειωθεί μέσω της καλύτερης διαχείρισής της. Για το σκοπό αυτό έχουν διενεργηθεί πολλές έρευνες τα τελευταία χρόνια και έχουν αναπτυχθεί νέες τεχνολογίες. Έτσι αποφάσισα να ασχοληθώ με μία από τις τεχνολογίες αυτές που υπόσχεται πολλά στον τομέα της διαχείρισης της συγκοινωνιακής υποδομής και δεν είναι άλλη από τα συστήματα πλοήγησης.

Στο σημείο αυτό θέλω να ευχαριστήσω τον καθηγητή μου, Αντώνιο Σταθόπουλο, που από την πρώτη συνάντησή μας με ενθάρρυνε να ασχοληθώ με αυτό τον τομέα και μου συμπαραστάθηκε όποτε και αν τον χρειάστηκα κατά τη διάρκεια της εκπόνησης αυτής της εργασίας. Οι συναντήσεις μας ήταν ιδιαίτερα εποικοδομητικές και με βοήθησαν να αντιληφθώ πολλά πράγματα, ιδιαίτερα σε τεχνικά ζητήματα. Αισθάνομαι τυχερός που συνεργαστήκαμε αυτό το διάστημα διότι δεν αποθάρρυνε ποτέ τις ιδέες μου και βοήθησε τα μέγιστα ώστε να προκύψει αυτό το αποτέλεσμα.

Επίσης θέλω να ευχαριστήσω τους καθηγητές του τομέα διότι εκτός από τεχνικές γνώσεις με βοήθησαν να αντιληφθώ τα προβλήματα που υπάρχουν στο χώρο των μεταφορών. Τέλος, ευχαριστώ τους γονείς μου που μου συμπαραστέκονται οικονομικά και ψυχολογικά όλα αυτά τα χρόνια και τους φίλους μου και ιδιαίτερα την Ευσταθία Σαρλά για τη στήριξή τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1^{ο}

ΕΙΣΑΓΩΓΗ

1.1.ΣΚΟΠΟΣ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ

Οι αστικές περιοχές αποτελούν το κέντρο των δραστηριοτήτων και καλούνται να ικανοποιήσουν αυξημένη μεταφορική ζήτηση. Ωστόσο πολλά αστικά κέντρα δεν είναι σχεδιασμένα για να εξυπηρετήσουν αυτήν τη ζήτηση και η πυκνή τους δόμηση δυσχεραίνει τις προσπάθειες βελτίωσης της ήδη υπάρχουσας υποδομής. Σκοπός της εργασίας είναι η διερεύνηση της συμβολής των νέων τεχνολογιών και συγκεκριμένα των συστημάτων πλοήγησης στη βελτίωση της χρήσης της διατιθέμενη υποδομής με οφέλη στο χρόνο μετακίνησης, την κατανάλωση ενέργειας και το περιβάλλον. Για το λόγο αυτό τίθενται οι επιμέρους στόχοι :

- Ανάλυση και συγκριτική αξιολόγηση των μεθόδων που μπορούν να χρησιμοποιηθούν από ένα σύστημα πλοήγησης σχετικά με τον υπολογισμό της διαδρομής που προτείνεται στο χρήστη και των κυριότερων εναλλακτικών της.
- Διερεύνηση της χρησιμότητας των συστημάτων πλοήγησης στον περιορισμό των περιβαλλοντικών επιπτώσεων που σχετίζονται με τις μεταφορές. Αναζήτηση και πρόταση μεθόδων ώστε να προτείνονται διαδρομές με μειωμένες εκπομπές CO_2, το οποίο ευθύνεται κατά κύριο λόγο για την κλιματική αλλαγή του πλανήτη.
- Αξιολόγηση της χρησιμότητας των συστημάτων πλοήγησης σε συγκοινωνιακά δίκτυα με προβλήματα κυκλοφοριακής συμφόρησης, όσο το ποσοστό των χρηστών που διαθέτει συστήματα πλοήγησης παραμένει σε χαμηλά επίπεδα.
- Διερεύνηση ενδεχόμενης βελτίωσης στην απόδοση των συγκοινωνιακών δικτύων όταν αυξηθεί το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης.

1.2.ΔΟΜΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

1^{ο} ΚΕΦΑΛΑΙΟ : Εισαγωγή. Παρουσιάζεται ο σκοπός και η δομή της εργασίας.

2^{ο} ΚΕΦΑΛΑΙΟ : Αναφέρονται τα στάδια που απαιτούνται για το σχεδιασμό των μεταφορικών συστημάτων καθώς και τα κυριότερα προβλήματα στο χώρο των μεταφορών. Ιδιαίτερη έμφαση δίνεται στο πρόβλημα του καταμερισμού της μεταφορικής ζήτησης και αναφέρονται οι βασικότερες ισορροπίες που εχουν διατυπωθεί στο χώρο των συγκοινωνιακών δικτύων. Παρουσιάζονται στοιχεία σχετικά με τις οικονομικές και περιβαλλοντικές επιπτώσεις της κυκλοφοριακής συμφόρησης. Αναφέρονται επίσης και νέες μέθοδοι, όπως το αποτύπωμα άνθρακα, που μπορούν να ποσοτικοποιήσουν την επίδραση της κυκλοφοριακής συμφόρησης σε θέματα σχετικά με το περιβάλλον, όπως το φαινόμενου του θερμοκηπίου. Τέλος αναφέρονται τα λειτουργικά στοιχεία των συστημάτων πλοήγησης, τα οποία μπορεί να αποτελέσουν ένα μέσο για τη βελτίωση του καταμερισμού της ζήτησης.

3^{ο} ΚΕΦΑΛΑΙΟ : Βιβλιογραφική ανασκόπηση των κυριότερων μεθόδων που έχουν προταθεί και μπορεί να χρησιμοποιηθούν έμμεσα ή άμεσα από τα συστήματα πλοήγησης. Ιδιαίτερη έμφαση δίνεται σε έρευνες που σχετίζονται με την αξιοποίηση των συστημάτων πλοήγησης με στόχο τη βελτίωση του καταμερισμού της ζήτησης σε συγκοινωνιακά δίκτυα με κυκλοφοριακή συμφόρηση. Αναφέρονται επίσης και έρευνες με στόχο την εύρεση των συντομότερων διαδρομών με περιορισμούς διαθέσιμων πόρων που μπορούν να χρησιμοποιηθούν για την επιλογή μετακινήσεων φιλικών προς το περιβάλλον, που θα είναι ταυτόχρονα και αποδοτικές.

4^{ο} ΚΕΦΑΛΑΙΟ : Παρουσιάζονται οι βασικές αρχές των αλγορίθμων και οι σημαντικότερες μέθοδοι προγραμματισμού. Επίσης κατηγοριοποιούνται τα προβλήματα εύρεσης των βέλτιστων διαδρομών και παρουσιάζονται οι δύο βασικές μέθοδοι αναπαράστασης των συγκοινωνιακών δικτύων. Τέλος αναφέρονται στοιχεία από την ανάλυση αλγορίθμων σχετικά με τον υπολογισμό του χρόνου εκτέλεσης και των απαιτούμενων θέσεων μνήμης κατά την εκτέλεση ενός αλγορίθμου.

5^{ο} ΚΕΦΑΛΑΙΟ : Παρουσιάζονται αλγόριθμοι που μπορούν να χρησιμοποιηθούν από ένα σύστημα πλοήγησης με στόχο τη δρομολόγηση του χρήστη. Ορισμένοι από αυτούς χρησιμοποιούνται για τον υπολογισμό της συντομότερης διαδρομής, αναλύονται όμως και άλλες κατηγορίες, όπως αλγόριθμοι που υπολογίζουν και εναλλακτικές διαδρομές παρέχοντας στο χρήστη τη δυνατότητα επιλογής. Επίσης παρουσιάζονται μέθοδοι για τη μείωση του μεγέθους συγκοινωνιακών δικτύων ώστε να περιοριστεί το εύρος αναζήτησης των αλγορίθμων. Τέλος αναφέρονται τεχνικές για τον υπολογισμό μίας διαδρομής που θα είναι σύντομη από άποψη χρόνου διάνυσης και θα πληροί και άλλα κριτήρια, όπως ο περιορισμός των εκπομπών CO_2 ή η μείωση της κατανάλωσης καυσίμου κ.α.

6^{ο} ΚΕΦΑΛΑΙΟ : Γίνεται αναφορά στο βέλτιστο καταμερισμό της ζήτησης σε στατικά δίκτυα, όπου οι χρόνοι διάνυσης των συνδέσμων δε μεταβάλλονται με το χρόνο. Επίσης προτείνονται και δύο νέες μέθοδοι για τον υπολογισμό της ελάχιστης

διαδρομής από ένα σημείο προς όλα τα υπόλοιπα σημεία του δικτύου. Τέλος, παρουσιάζεται η χρησιμότητα των συστημάτων πλοήγησης στα δίκτυα αυτά.

7^{ο} ΚΕΦΑΛΑΙΟ : Αναφέρονται τα προβλήματα που προκαλούνται σε συγκοινωνιακά δίκτυα λόγω της κυκλοφοριακής συμφόρησης. Παρουσιάζονται τεχνολογίες μέτρησης κυκλοφοριακών μεγεθών με στόχο τη συνεχή ενημέρωση των συστημάτων πλοήγησης σχετικά με τις κυκλοφοριακές συνθήκες που επικρατούν στο δίκτυο. Αναλύονται οι κυριότερες μέθοδοι που μπορούν να χρησιμοποιηθούν από τα συστήματα πλοήγησης για την επιλογή διαδρομής σε δυναμικά δίκτυα υπό συνθήκες αβεβαιότητας. Εξετάζεται η χρησιμότητα των συστημάτων πλοήγησης στη βελτίωση του καταμερισμού της ζήτησης και προτείνονται μέθοδοι που μπορούν να χρησιμοποιηθούν από αυτά ή από κάποιον κεντρικό διαχειριστή ώστε να επιτευχθεί αυτός ο στόχος. Τέλος, αναλύονται οι σημαντικότερες ισορροπίες των δυναμικών δικτύων και οι δυσκολίες που προκύπτουν κατά τη διαδικασία καταμερισμού της ζήτησης μέσω των συστημάτων πλοήγησης.

8^{ο} ΚΕΦΑΛΑΙΟ : Πραγματοποιούνται εφαρμογές σε συγκοινωνιακά δίκτυα μέσω αλγορίθμων που έχουν προγραμματιστεί και βρίσκονται στο Παράρτημα. Μέσω των εφαρμογών αυτών συγκρίνεται η αποδοτικότητα των αλγορίθμων που μπορούν να χρησιμοποιηθούν από τα συστήματα πλοήγησης με σκοπό την καθοδήγηση του χρήστη. Πραγματοποιούνται επίσης εφαρμογές σε δυναμικά δίκτυα ώστε να ερευνηθεί η χρησιμότητα των συστημάτων πλοήγησης στον καταμερισμό της ζήτησης και το κατά πόσο μπορεί να βελτιωθεί εάν χρησιμοποιηθούν νέες μέθοδοι που έχουν αναπτυχθεί τα τελευταία χρόνια στο χώρο των ευφυών συστημάτων μεταφορών.

9^{ο} ΚΕΦΑΛΑΙΟ : Παρουσιάζονται τα συμπεράσματα και γίνονται εισηγήσεις για μελλοντική έρευνα.

ΚΕΦΑΛΑΙΟ 2^{ο}

ΔΙΚΤΥΑ ΜΕΤΑΦΟΡΩΝ

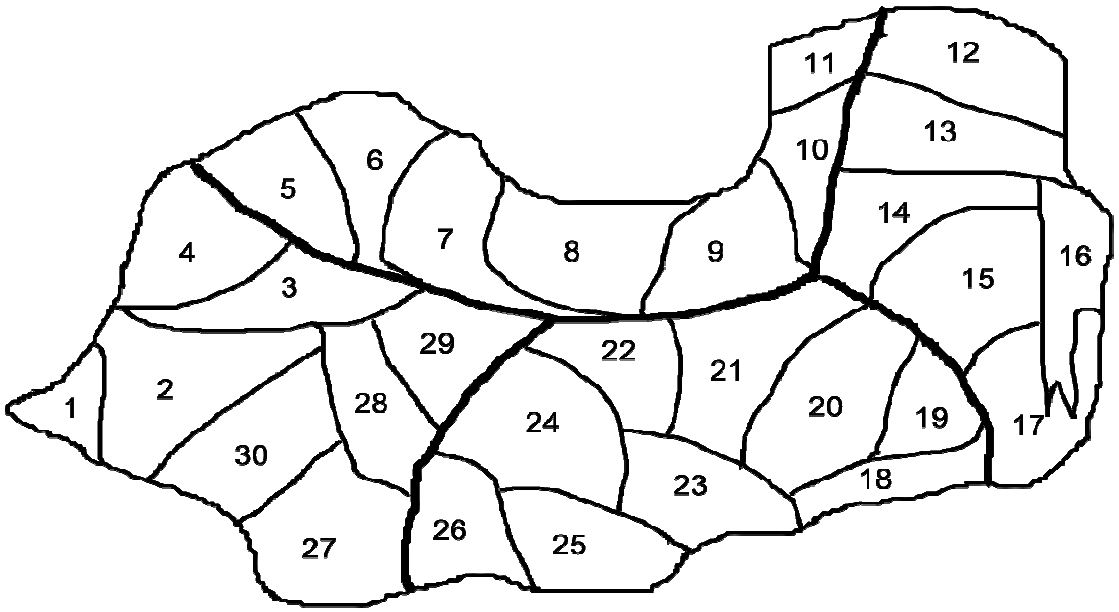
2.1.ΓΕΝΙΚΕΣ ΕΝΝΟΙΕΣ ΓΙΑ ΤΟ ΣΧΕΔΙΑΣΜΟ ΜΕΤΑΦΟΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Η ανάγκη για μεταφορική υποδομή είναι συνυφασμένη με τις ανθρώπινες κοινωνίες. Η μεταφορική υποδομή μιας περιοχής δημιουργείται με σκοπό την εξυπηρέτηση των δραστηριοτήτων του πληθυσμού της. Οι δραστηριότητες όμως αυτές μεταβάλλονται συνεχώς και απρόβλεπτα στο χρόνο. Έτσι ο προσδιορισμός της μεταφορικής υποδομής που απαιτείται σε μία περιοχή αποτελεί βασικό συγκοινωνιακό πρόβλημα και προϋποθέτει την ανάπτυξη σύνθετων μοντέλων που επικεντρώνονται στη γένεση των μετακινήσεων αλλά και τον καταμερισμό τους.

Ως μετακίνηση μπορεί να οριστεί η διαδικασία κατά την οποία ένα άτομο ή εμπόρευμα μετακινείται ή μεταφέρεται από ένα σημείο i του χώρου σε ένα άλλο σημείο j με συγκεκριμένο σκοπό. Για την εκτίμηση των παραγόμενων μετακινήσεων, που προσδιορίζουν έμμεσα και τη ζήτηση για μεταφορική υποδομή, γίνονται διάφορες παραδοχές. Οι παραδοχές αυτές έχουν ως στόχο την προσομοίωση της πραγματικότητας μέσω της χρήσης ενός απλουστευμένου μοντέλου ώστε να μπορούν να αναχθούν τα ανάλογα συμπεράσματα. Αρχικά η περιοχή μελέτης υποδιαιρείται σε μη επικαλυπτόμενες υποπεριοχές και αυτές με τη σειρά τους υποδιαιρούνται σε ακόμα μικρότερες υποπεριοχές οι οποίες καλούνται ζώνες. Θεωρείται ότι στο κέντρο κάθε ζώνης, που ονομάζεται κεντροειδές της ζώνης, συγκεντρώνονται όλες οι δραστηριότητες και η μονάδα ανάλυσης είναι η κυκλοφοριακή ζώνη.

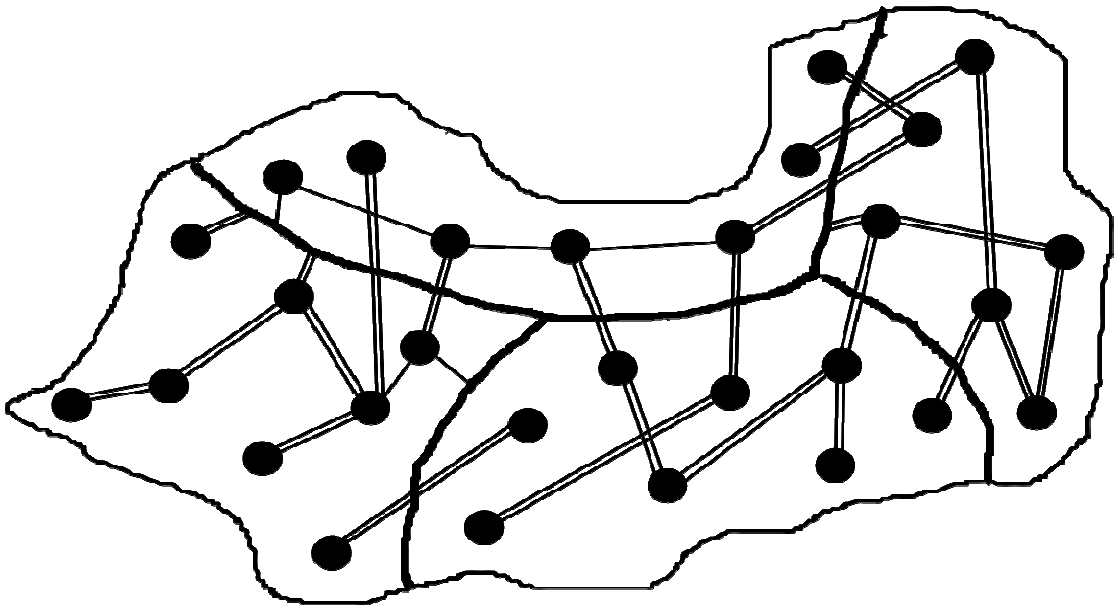
Για τη διαμόρφωση των ορίων της κυκλοφοριακής ζώνης λαμβάνονται υπόψη διάφοροι παράγοντες όπως :

α) Χρήσεις Γης
β) Όρια Διοικητικών Ενοτήτων
γ) Θέση Κέντρων Δραστηριότητας
δ) Δίκτυα Μεταφορών και Γεωγραφικά Χαρακτηριστικά



Σχήμα 2.1.Υποδιαίρεση Περιοχής Μελέτης σε Ζώνες

Στη συνέχεια θεωρείται ότι τα κεντροειδή των ζωνών του σχήματος 2.1., στα οποία συγκεντρώνεται το σύνολο των δραστηριοτήτων, συνδέονται μεταξύ τους μέσω του συγκοινωνιακού δικτύου όπως φαίνεται στο παρακάτω σχήμα :



Σχήμα 2.2.Σύνδεση Ζωνών μέσω συγκοινωνιακού δικτύου

Κάθε ζώνη παράγει και προσελκύει μετακινήσεις ανάλογα με τις δραστηριότητες που συγκεντρώνει. Έτσι ακολουθείται μία διαδικασία, γνωστή ως γένεση μετακινήσεων, με την οποία μεγέθη μιας δραστηριότητας (εργασία, αγορές, ψυχαγωγία εκπαίδευση, κλπ) μετατρέπονται σε αριθμό μετακινήσεων. Το αποτέλεσμα αυτής της διαδικασίας είναι η πρόβλεψη των μετακινήσεων που παράγονται ή έλκονται από κάθε ζώνη. Υπάρχουν τρεις μέθοδοι ανάλυσης της γένεσης των μετακινήσεων :

α) Η μέθοδος της ανάλυσης κατά κατηγορίες (Cross Classification – Category Analysis). Τα νοικοκυριά ταξινομούνται σε κατηγορίες ανάλογα με τα χαρακτηριστικά τους (εισόδημα, διαθεσιμότητα ΙΧ, μέγεθος, αριθμός εργαζόμενων, κ.α.) και στη συνέχεια για κάθε κατηγορία υπολογίζεται ο ρυθμός γένεσης των μετακινήσεων από μετρήσεις για την υπάρχουσα κατάσταση. Συνήθως γίνεται η παραδοχή ότι ο ρυθμός γένεσης μετακινήσεων σε κάθε κατηγορία παραμένει σταθερός για την περίοδο που γίνονται οι προβλέψεις. Έτσι ο αριθμός των μετακινήσεων που παράγονται από τη ζώνη *i* στον χρονικό ορίζοντα των προβλέψεων *t* είναι : $M^i(\mathrm{t}) = \Sigma_{\alpha,\beta,\gamma\ldots} R^i_{\alpha,\beta,\gamma,\ldots}(t) \times \mathrm{f}_{\alpha,\beta,\gamma,\ldots}$

όπου $R^i_{\alpha,\beta,\gamma,\ldots}(t)$ είναι ο αριθμός των νοικοκυριών της ζώνης i που προβλέπεται ότι θα ανήκουν στην κατηγορία α, β, γ,… στο χρονικό διάστημα t και $\mathrm{f}_{\alpha,\beta,\gamma,\ldots}$ είναι ο ρυθμός των μετακινήσεων που παράγονται από ένα νοικοκυριό που ανήκει στην κατηγορία α, β, γ, … και θεωρείται σταθερός. Ο ρυθμός των μετακινήσεων που παράγονται υπολογίζεται από έρευνες σε δείγματα από τα νοικοκυριά κάθε κατηγορίας.

β) Η μέθοδος της ανάλυσης παλινδρόμησης (Regression Analysis). Με τη μέθοδο αυτή μπορεί να ακολουθηθεί η ανάλυση κατά ζώνη (αθροιστικά μοντέλα-aggregate models), όπου εκφράζεται ο αριθμός των παραγόμενων ή ελκυόμενων μετακινήσεων ως συνάρτηση των κοινωνικο-οικονομικών και λοιπών χαρακτηριστικών κάθε ζώνης. Επίσης μπορεί να ακολουθηθεί και η ανάλυση κατά άτομο (εξατομικευμένα μοντέλα-disaggregate models) , όπου εκφράζεται ο αριθμός των παραγόμενων ή ελκυόμενων μετακινήσεων ως συνάρτηση των χαρακτηριστικών κάθε ατόμου. Η μορφή της συναρτησιακής σχέσης και οι τιμές των παραμέτρων υπολογίζονται χρησιμοποιώντας τη θεωρία της ανάλυσης παλινδρόμησης.

γ) Η μέθοδος του συντελεστή ανάπτυξης (Growth Factor). Με τη μέθοδο αυτή εκτιμώνται οι μελλοντικές μετακινήσεις βασιζόμενες στις μετακινήσεις του έτους βάσης ως εξής : $M_i = \mu_i \times \mathrm{F_i}$, όπου :

M_i : Οι μελλοντικές μετακινήσεις στη ζώνη *i*.

μ_i : Οι παρατηρούμενες μετακινήσεις στο έτος βάσης στη ζώνη *i*.

Fi : Ο συντελεστής ανάπτυξης που σχετίζεται με τον πληθυσμό, την ιδιοκτησία Ι.Χ., το εισόδημα κ.α. στην υπάρχουσα και τη μελλοντική κατάσταση.

Η μέθοδος αυτή χρησιμοποιείται όταν δεν αναμένονται δομικές αλλαγές του συγκοινωνιακού δικτύου και προσφέρει εκτιμήσεις χαμηλού κόστους αλλά και περιορισμένης ερμηνευτικότητας λόγω των απλοποιητικών παραδοχών που χρησιμοποιεί.

Έπειτα από το στάδιο της γένεσης των μετακινήσεων τα άκρα των μετακινήσεων που προκύπτουν μπορούν να συνδεθούν με γραμμές επιθυμίας μετακινήσεων από μία ζώνη σε μία άλλη. Θεωρώντας δηλαδή τη ζώνη γένεσης της μετακίνησης ως σημείο αναφοράς αναζητείται η ζώνη έλξης της μετακίνησης. Αυτή η διαδικασία είναι γνωστή και ως κατανομή των μετακινήσεων. Οι επιθυμίες μετακινήσεων

συναρτώνται με τη διατιθέμενη συγκοινωνιακή υποδομή και τα αποτελέσματα αποτυπώνονται ως ζήτηση μετακινήσεων μέσω μητρώων Προέλευσης-Προορισμού. Η κατανομή των μετακινήσεων προκύπτει μέσω αθροιστικών μοντέλων όπως το μοντέλο βαρύτητας. Μέσω του μοντέλου αυτού θεωρείται ότι ο αριθμός των μετακινήσεων που παράγονται από μία ζώνη i και έλκονται από μία ζώνη j εξαρτάται από διάφορους παράγοντες όπως τον αριθμό των μετακινήσεων που παράγονται από τη ζώνη i, τον αριθμό των μετακινήσεων που έλκονται από τη ζώνη j και μία συνάρτηση τριβής για τη μετακίνηση από τη ζώνη i στη ζώνη j.

Στη συνέχεια της διαδικασίας του σχεδιασμού μεταφορικών συστημάτων η μεταφορική ζήτηση που έχει προκύψει και αποτυπώνεται μέσω μητρώων Προέλευσης-Προορισμού καταμερίζεται στα διατιθέμενα μέσα μεταφοράς. Για το σκοπό αυτό χρησιμοποιούνται διάφορα αθροιστικά μοντέλα όπως :

α) Μοντέλα Καταμερισμού στα άκρα της μετακίνησης : Είναι ένα από τα πρώτα μοντέλα που χρησιμοποιήθηκαν και βασίζονται στην υπόθεση ότι τα χαρακτηριστικά του μετακινούμενου καθορίζουν και τις επιλογές του. Συσχετίζει την επιλογή του μέσου με διάφορα χαρακτηριστικά όπως το εισόδημα, η πυκνότητα δόμησης, η κατοχή ιδιωτικού μεταφορικού μέσου κ.α. Δεν είναι κατάλληλα για μελλοντικές προβλέψεις και δεν είναι ευαίσθητα σε αλλαγές στα χαρακτηριστικά των μεταφορικών συστημάτων.

β) Μοντέλα Καταμερισμού μετακινήσεων με καμπύλες καταμερισμού : Λαμβάνουν υπόψη τα χαρακτηριστικά των μετακινήσεων, όπως ο χρόνος εντός του μεταφορικού μέσου, αλλά όχι τα χαρακτηριστικά των μετακινούμενων. Χρησιμοποιούν καμπύλες που δίνουν το ποσοστό χρήσης κάθε μέσου ως συνάρτηση της διαφοράς (ή του λόγου) του χρόνου/κόστους μετακίνησης του συγκεκριμένου μέσου από το χρόνο/κόστος διαδρομής ενός ανταγωνιστικού μέσου. Οι καμπύλες προκύπτουν από την ανάλυση στοιχείων κυκλοφοριακών ερευνών και έχουν σιγμοειδή μορφή. Η αξιοπιστία των προβλέψεων ωστόσο είναι αμφίβολη καθώς δε βασίζονται σε κάποια θεωρεία ανάλυσης των επιλογών.

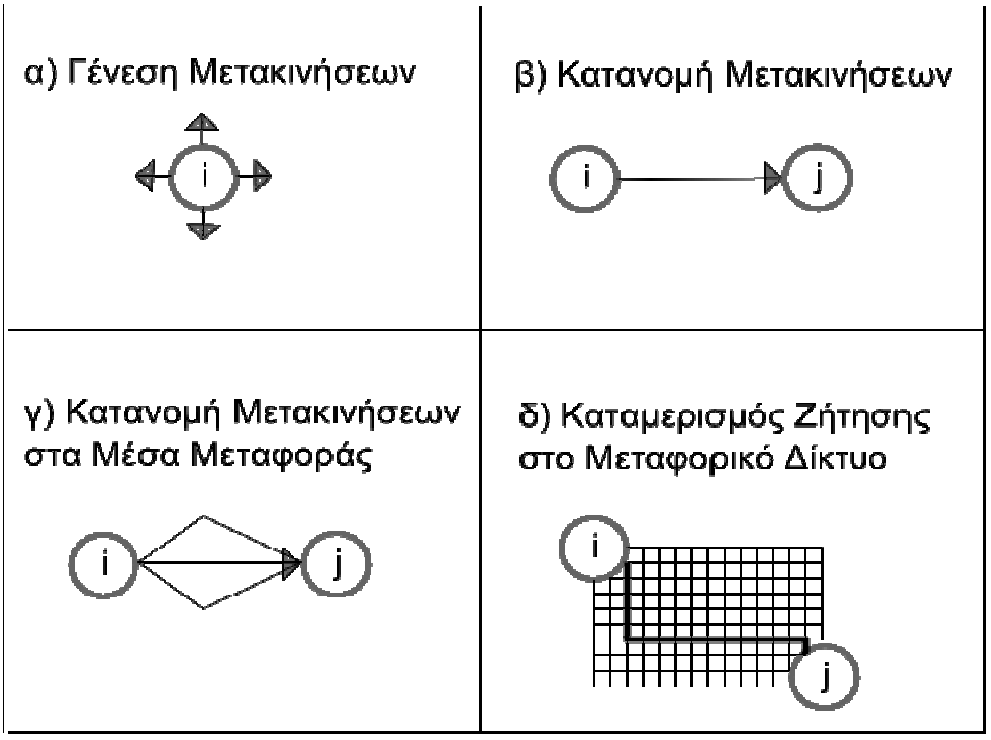
γ) Πολυωνυμικά Μοντέλα Λογιστικής συνάρτησης : Θεωρούν ότι η πιθανότητα ένας μετακινούμενος να επιλέξει το μεταφορικό μέσο i ισούται με την πιθανότητα η συστηματική χρησιμότητα που σχετίζεται με την εναλλακτική αυτή να είναι μεγαλύτερη από τη συστηματική χρησιμότητα όλων των υπόλοιπων εναλλακτικών. Για το σκοπό αυτό απαιτείται ο προσδιορισμός της συνάρτησης χρησιμότητας για κάθε μεταφορικό μέσο. Τα μοντέλα μπορούν να χρησιμοποιηθούν εφόσον ο λόγος επιλογής ενός μέσου έναντι οποιουδήποτε άλλου μέσου είναι ανεξάρτητος από το σύνολο των επιλογών που έχει στη διάθεσή του ο μετακινούμενος.

Στο τελικό στάδιο του σχεδιασμού μεταφορικών συστημάτων η ζήτηση για μετακινήσεις, που αποτυπώνεται στα μητρώα Προέλευσης-Προορισμού κάθε μεταφορικού μέσου, καταμερίζεται στο υπάρχον δίκτυο. Εάν εξετασθεί ένα

μεταφορικό μέσο με συγκεκριμένο μητρώο Π-Π ο καταμερισμός της ζήτησης στο υπάρχον δίκτυο ακολουθεί την παρακάτω διαδικασία :

Για τη μετακίνηση μεταξύ δύο ζωνών οι χρήστες ακολουθούν τη συντομότερη διαδρομή. Στη συνέχεια η διαδρομή αυτή λόγω αύξησης της κυκλοφορίας παύει να είναι η συντομότερη και χρησιμοποιείται μία άλλη διαδρομή και στη συνέχεια μία τρίτη κ.ο.κ. μέχρι να αυξηθεί η κυκλοφορία και στις εναλλακτικές διαδρομές και να αρχίσει ξανά να χρησιμοποιείται η αρχική διαδρομή. Ο καταμερισμός της ζήτησης στο μεταφορικό δίκτυο μπορεί να γίνει με τη βοήθεια αλγορίθμων. Μέσω αυτής της διαδικασίας το δίκτυο φορτίζεται με κυκλοφορία οχημάτων και μπορούν να γίνουν εκτιμήσεις για ανάγκη νέων υποδομών ή για άλλες πιθανές βελτιώσεις. Περισσότερες πληροφορίες σχετικά με το σχεδιασμό μεταφορικών συστημάτων υπάρχουν στο βιβλίο Σταθόπουλος, Καρλαύτης (2008).

Τα στάδια του σχεδιασμού μεταφορικών συστημάτων, όπως αναπτύχθηκαν παραπάνω, παρουσιάζονται στο ακόλουθο σχήμα :



Σχήμα 2.3.Διαδικασία Σχεδιασμού Μεταφορικών Συστημάτων

2.2.ΠΡΟΒΛΗΜΑΤΑ ΜΕΤΑΦΟΡΙΚΩΝ ΔΙΚΤΥΩΝ ΚΑΙ ΜΕΤΡΑ ΑΝΤΙΜΕΤΩΠΙΣΗΣ ΚΥΚΛΟΦΟΡΙΑΚΩΝ ΠΡΟΒΛΗΜΑΤΩΝ

Οι αστικές περιοχές έχουν υψηλή συγκέντρωση εμπορικών, οικονομικών κ.α. δραστηριοτήτων και αποτελούνται από πολύπλοκες δομές που υποστηρίζονται από συγκοινωνιακά δίκτυα. Τα περισσότερα συγκοινωνιακά προβλήματα εμφανίζονται σε αστικές περιοχές και συνήθως είναι αποτέλεσμα της μη ικανοποίησης των σύνθετων απαιτήσεων που έχουν αυτές οι περιοχές. Η ανάπτυξη της αστικής περιοχής είναι άρρηκτα συνδεδεμένη με την ποιότητα του συγκοινωνιακού δικτύου καθώς μέσα από αυτό επιτελείται το μεγαλύτερο μέρος των δραστηριοτήτων της. Για το λόγο αυτό είναι απαραίτητο να περιοριστούν στο ελάχιστο τα προβλήματα που αντιμετωπίζουν τα συγκοινωνιακά δίκτυα. Ορισμένα από τα πιο συνηθισμένα προβλήματα που αντιμετωπίζουν είναι τα ακόλουθα :

α) Κυκλοφοριακή συμφόρηση σε συγκεκριμένα τμήματα του μεταφορικού δικτύου. Αποτελεί ένα από τα πιο κοινά συγκοινωνιακά προβλήματα και μπορεί να προκύψει λόγω του λάθος σχεδιασμού του δικτύου, της ανεπάρκειας μεταφορικής υποδομής ώστε να καλυφθεί η απαιτούμενη ζήτηση, του λάθος καταμερισμού της ζήτησης στο σύνολο του μεταφορικού δικτύου κ.α.

β) Κυκλοφοριακή συμφόρηση κατά τη διάρκεια κάποιας χρονικής περιόδου μέσα στην ημέρα. Αποτελεί και αυτό σύνηθες πρόβλημα και οφείλεται κυρίως στη συγκέντρωση των δραστηριοτήτων στο χρονικό διάστημα αυτό.
γ) Κυκλοφοριακή συμφόρηση κατά τη διάρκεια κάποιων μηνών του έτους. Παρατηρείται κυρίως σε περιοχές με έντονη διακύμανση στην ένταση των δραστηριοτήτων τους όπως τουριστικά θέρετρα κ.α.
δ) Χαμηλή στάθμη συγκοινωνιακής εξυπηρέτησης. Αποτελεί πρόβλημα που προέρχεται από τον κορεσμό του δικτύου λόγω κυκλοφοριακής συμφόρησης.
ε) Αυξημένη χρήση Ι.Χ. τα οποία έχουν μικρό συντελεστή πλήρωσης και επιδεινώνουν την κυκλοφοριακή συμφόρηση.
στ) Γενικότερα οχήματα με μειωμένο συντελεστή πλήρωσης.
ζ) Ανάπτυξη αστικών περιοχών χωρίς πρόβλεψη για ενδεχόμενη αύξηση της κυκλοφορίας. Αποτελεί και αυτό σύνηθες πρόβλημα με αρνητικές επιπτώσεις καθώς δεν προβλέπεται ο απαραίτητος χώρος για την επέκταση της διατιθέμενης υποδομής σε μελλοντικό χρόνο.
η) Αλλαγές στις χρήσεις γης και μεταβολές δραστηριοτήτων. Αποτελούν μία από τις αβεβαιότητες του συστήματος το οποίο χρησιμοποιεί τη συγκοινωνιακή υποδομή. Λόγω αυτής της αβεβαιότητας υπάρχουν συχνά προβλήματα κορεσμού ή υπερδιαστασιολόγησης του μεταφορικού δικτύου ακόμα και σε μικρό διάστημα μετά την κατασκευή του. Μπορεί να προκληθούν και μέσω κυβερνητικών αποφάσεων, κάτι που δυσχεραίνει κι άλλο την πρόβλεψη για τη μελλοντική κατάσταση.
θ) Αυξημένη συχνότητα μετακινήσεων.
ι) Απότομη αύξηση μετακινήσεων και χρήσης Ι.Χ. λόγω κρατικών παρεμβάσεων.
κ) Αλόγιστη κατανάλωση ενέργειας και εξάρτηση από μη ανανεώσιμες πηγές.

λ) Περιβαλλοντικές επιπτώσεις. Περιλαμβάνουν συνήθως την ατμοσφαιρική ρύπανση και την ηχορύπανση.
μ) Αυξημένο μήκος και διάρκεια μετακινήσεων. Προκύπτει από λάθη στο σχεδιασμό του μεταφορικού δικτύου αλλά και έλλειψη ενημέρωσης του χρήστη σχετικά με το ποιά είναι η συντομότερη διαδρομή που πρέπει να ακολουθήσει για να φθάσει στον προορισμό του τη δεδομένη χρονική στιγμή.
ν) Αυξημένη κυκλοφοριακή ροή σε οικιστικές ζώνες. Αυτό έχει αντίκτυπο στην ποιότητα ζωής των κατοίκων μέσω διάφορων μορφών όπως η ηχορύπανση, τα ατυχήματα, ο περιορισμός των χώρων αναψυχής κ.α.
ξ) Περιορισμένες δυνατότητες επέκτασης μεταφορικού δικτύου, ιδίως σε περιοχές με πυκνή δόμηση. Παρατηρείται κυρίως στα κέντρα των πόλεων που αποτελούν τα κέντρα δραστηριοτήτων αλλά δε μπορούν να καλύψουν τη μεταφορική ζήτηση.
ο) Ατυχήματα και ασφάλεια. Με την αύξηση της ζήτησης για μετακινήσεις αυξάνεται και ο αριθμός των ατυχημάτων, τα οποία εκτός από τις αρνητικές τους συνέπειες σε κοινωνικό και οικονομικό επίπεδο προκαλούν και καθυστερήσεις στο μεταφορικό δίκτυο.

Τα παραπάνω προβλήματα μπορούν να περιοριστούν μέσω κατάλληλων δράσεων. Ορισμένα από τα μέτρα διαχείρισης της κυκλοφορίας για την αντιμετώπισή τους είναι τα ακόλουθα :

α) Βελτίωση Ροής Οχημάτων με σκοπό την αύξησης της κυκλοφοριακής ικανότητας, του χρόνου διαδρομής και τη μείωση των ατυχημάτων και περιλαμβάνει διάφορα μέτρα όπως :
Βελτίωση σηματοδότησης, μονοδρομήσεις, απαγόρευση κινήσεων, έλεγχος εισόδων ελεύθερων λεωφόρων, απαγόρευση στάθμευσης παρά το κράσπεδο, χρησιμοποίηση για κίνηση των ερεισμάτων των ελεύθερων λεωφόρων και λεωφορειολωρίδων.

β) Προνομιακή Μεταχείριση και Βελτίωση Μαζικών Μεταφορών με σκοπό τη μείωση της χρήσης των επιβατικών αυτοκινήτων με μέτρα όπως :
Λωρίδες λεωφορείων και λεωφορειόδρομοι, προτεραιότητα λεωφορείων στη σηματοδότηση, ειδική μεταχείριση λεωφορείων με σήμανση, αναμόρφωση δρομολογίων και επέκταση γραμμών, επικοινωνία και κατεύθυνση οχημάτων.

γ) Προαγωγή Μετακινήσεων με Ανθρώπινη Ενέργεια με σκοπό τη μείωση μετακινήσεων με μηχανοκίνητα μέσα και μείωση της κυκλοφοριακής συμφόρισης με μέτρα όπως :
Διευκόλυνση κυκλοφορίας πεζών, διευκόλυνση κυκλοφορίας ποδηλάτων, χώροι στάθμευσης ποδηλάτων, ποδηλατοδρόμοι.

δ) Διαχείριση Στάθμευσης με σκοπό την προώθηση των Μ.Μ.Μ. με μέτρα όπως :
Αύξηση τελών στάθμευσης, περιορισμό προσφοράς στάθμευσης, αύξηση αστυνόμευσης, απαγόρευση στάθμευσης σε περιοχές με προβλήματα κυκλοφοριακής συμφόρησης.

ε) Προνομιακή Μεταχείριση Επιβατικών Αυτοκινήτων με Υψηλή Πλήρωση με σκόπο την εξυπηρέτηση μεγαλύτερου αριθμού μετακινούμενων για τον ίδιο αριθμό οχημάτων με μέτρα όπως :
Ενίσχυση από εργοδότες της ομαδικής χρήσης αυτοκινήτων, ειδικές λεωφορειολωρίδες, διόδια, ταξί πολλαπλής μίσθωσης, προνομιακή είσοδος σε αυτοκινητόδρομους.

στ) Περιορισμοί Κυκλοφορίας Οχημάτων με στόχο την απαγόρευση της κυκλοφορίας σε κρίσιμες περιοχές για να περιοριστεί η κυκλοφοριακή συμφόρηση με κυριότερα μέτρα όπως :
Παρεμπόδιση διαμπερούς κυκλοφορίας μέσα από γειτονιές, περιοδική ή μόνιμη απαγόρευση της κυκλοφορίας σε κρίσιμες περιοχές, περιορισμοί κυκλοφορίας φορτηγών αυτοκινήτων, περιορισμοί κυκλοφορίας σε μία περιοχή με χρέωση.

ζ) Μείωση Μετακινήσεων σε Περιόδους Αιχμής με σκοπό τη βελτίωση των συνθηκών κυκλοφορίας κατά τις κρίσιμες περιόδους αυξημένης κίνησης και την αποφυγή κατασκευής έργων απαραίτητων μόνο για τις περιόδους αυτές με μέτρα όπως :
Αλλαγή ωραρίων εργασίας σε ορισμένες ομάδες εργαζομένων, χρέωση κατά τις περιόδους αιχμής ή αύξηση των υφιστάμενων διοδίων.

η) Διασπορά των Δραστηριοτήτων στον Αστικό Χώρο με σκοπό την κατανομή των μετακινήσεων στο χώρο και τη μείωση της κυκλοφοριακής συμφόρησης σε τμήματα συγκέντρωσης των δραστηριοτήτων με μέτρα όπως :
Αλλαγή χρήσεων γης, δημιουργία θέσεων εργασίας και μεταφορά βασικών υπηρεσιών σε αποκεντρωμένες ζώνες.

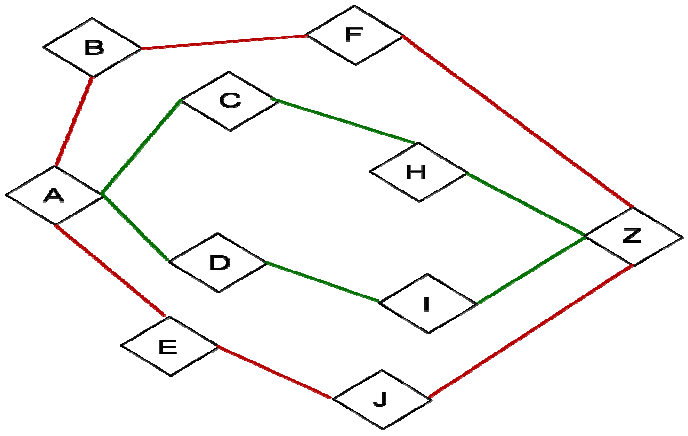
θ) Μείωση Συχνότητας Μετακινήσεων με σκοπό τη μείωση της κυκλοφοριακής συμφόρησης με μέτρα όπως :
Χρησιμοποίηση των Η/Υ για αγορές προϊόντων και πληρωμή λογαριασμών, πληρωμές λογαριασμών μέσω τραπεζών ή ταχυδρομείου κ.α.

ι) Εφαρμογή Νέων Τεχνολογιών με εφαρμογή εξελιγμένων μεθόδων πληροφορικής και τηλεπικοινωνιών που αναπτύχθηκαν τα τελευταία χρόνια με στόχο την πιο αποδοτική χρήση της μεταφορικής υποδομής και τη μείωση της κυκλοφοριακής συμφόρησης μέσω της ενημέρωσης του χρήστη με εφαρμογές όπως :
Συστήματα Πλοήγησης (Navigation Systems), ηλεκτρονικοί πίνακες μεταβαλλόμενων μηνυμάτων (VMS-Variable Message Signs), ψηφιακό ραδιόφωνο κ.α. Η αποδοτική χρήση της μεταφορικής υποδομής μέσω των νέων τεχνολογιών αποτελεί μεγάλη πρόκληση και έχει απασχολήσει το χώρο των συγκοινωνιών τα τελευταία χρόνια. Ήδη από το 1984 η Ευρωπαϊκή Επιτροπή ξεκίνησε το 1^{ο} Πρόγραμμα-Πλαίσιο για την έρευνα και την τεχνολογία (ESPIRIT και RACE) και έγιναν έρευνες σχετικά με το πως θα μπορούσαν τα συστήματα πληροφορικής και

τηλεπικοινωνιών να χρησιμοποιηθούν για την παροχή ενημέρωσης στους χρήστες των μεταφορικών δικτύων. Συνέχεια αυτού του προγράμματος αποτέλεσε το πρόγραμμα DRIVE (Dedicated Road Infrastructure for Vehicle Efficiency in Europe) το 1988 και το ADVANCED TRANSPORT TELEMATICS που έληξε το 1995. Παρόμοια προγράμματα συνεχίζονται ακόμα και σήμερα και αποδεικνύουν τη χρησιμότητα των νέων τεχνολογιών στην αντιμετώπιση προβλημάτων στα μεταφορικά δίκτυα. Μία από αυτές τις νέες τεχνολογίες είναι και τα συστήματα πλοήγησης που έχουν και αυτά με τη σειρά τους ως στόχο την παροχή πληροφοριών στο χρήστη της μεταφορικής υποδομής. Φραντζεσκάκης και συν. (1997)

2.3.ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΚΑΤΑΜΕΡΙΣΜΟΥ ΤΗΣ ΖΗΤΗΣΗΣ – ΕΞΙΣΟΡΡΟΠΗΣΗ ΔΙΚΤΥΩΝ

Για τη βελτίωση της χρήσης ενός συγκοινωνιακού δικτύου με προβλήματα κορεσμού απαιτείται να βελτιωθεί ο καταμερισμός της φόρτισής του. Μέχρι σήμερα έχουν πραγματοποιηθεί πολλές έρευνες με σκοπό τον προσδιορισμό της συνθήκης υπό την οποία μεγιστοποιείται η αποδοτικότητα των δικτύων. Ο J.G.Wardrop (1952) ήταν ο πρώτος που έδωσε δύο ορισμούς σχετικά με την ισορροπία των συγκοινωνιακών δικτύων. Η πρώτη αρχή του Wardrop ορίζει ότι οι χρόνοι διάνυσης όλων των διαδρομών που χρησιμοποιήθηκαν για τη μετακίνηση από το κεντροειδές μίας ζώνης προς το κεντροειδές μίας άλλης ζώνης θα πρέπει να είναι ίσοι και μάλιστα μικρότεροι από οποιονδήποτε άλλο χρόνο οποιασδήποτε άλλης μη χρησιμοποιούμενης διαδρομής. Αυτή η αρχή είναι γνωστή και ως ισορροπία του χρήστη (User's Equilibrium) καθώς κάθε χρήστης επιλέγει τη βέλτιστη διαδρομή γι'αυτόν με βάση το κόστος διάνυσης κάθε συνδέσμου του δικτύου. Οι ροές που διαμορφώνονται στο δίκτυο βάσει αυτής της αρχής ονομάζονται ροές ισορροπίας χρήστη. Για να ικανοποιείται ο ορισμός αυτός στο παρακάτω σχήμα όπου οι χρησιμοποιούμενες διαδρομές για τη μετακίνηση από τη ζώνη Α προς τη ζώνη Z συμβολίζονται με πράσινο χρώμα και οι μη χρησιμοποιούμενες με κόκκινο θα πρέπει ο χρόνος μετακίνησης από τη ζώνη Α στη Z μέσω των διαδρομών A→C→H→Z και A→D→I→Z να είναι ίσος και μικρότερος από το χρόνο μετακίνησης μέσω των διαδρομών A→B→F→Z και A→E→J→Z.



Σχήμα 2.4.Εξισορρόπηση Συγκοινωνιακού Δικτύου ως προς το χρήστη

Μία διαφοροποίηση αυτής της αρχής αποτελεί η στοχαστική ισορροπία του χρήστη (Stochastic User's Equilibrium). Ο όρος αυτός οφείλεται στους C.F.Daganzo and Y.Sheffi (1977). Σύμφωνα με αυτήν κανένας χρήστης δε μπορεί να βελτιώσει τον αντιληπτό χρόνο μετακίνησής του ενεργώντας μονομερώς (επιλέγοντας άλλη διαδρομή). Η ισορροπία αυτή υπονοεί ότι οι χρήστες δεν ακολουθούν τις συντομότερες διαδρομές, διότι δε μπορούν να απαριθμήσουν και να συγκρίνουν τις εναλλακτικές τους, αλλά τις διαδρομές που θεωρούν συντομότερες.

Η δεύτερη αρχή του Wardrop, γνωστή ως ισορροπία του συστήματος (System Optimum) ορίζει ότι στην κατάσταση ισορροπίας του δικτύου το μέσο κόστος μετακίνησης είναι και το ελάχιστο. Από τον ορισμό αυτό συνεπάγεται ότι οι χρήστες δεν επιλέγουν πάντα τη συντομότερη διαδρομή προς τον προορισμό τους. Οι κυκλοφοριακές ροές που διαμορφώνονται με βάση αυτή την αρχή ονομάζονται βέλτιστες ροές για το σύστημα καθώς αυξάνουν τη χωρητικότητά του. Η αρχή αυτή θεωρεί ότι τα κορεσμένα δίκτυα μπορούν να επιλυθούν ως προς ένα ολικό βέλτιστο. Για να συμβεί αυτό θα πρέπει οι αποφάσεις να λαμβάνονται από έναν κεντρικό διαχειριστή και οι χρήστες του δικτύου να λαμβάνουν την ανάλογη πληροφόρηση. Απαιτείται δηλαδή η συνεργασία τους ακόμα και εάν χρειαστεί να ακολουθήσουν μη αποδοτικές διαδρομές.

Η εύρεση της ισορροπίας του συστήματος μπορεί να εκφραστεί μαθηματικά ως εξής : Εάν σε συγκοινωνιακό δίκτυο οριστεί ζώνη προέλευσης μετακίνησης A και ζώνη προορισμού Z, τότε υπάρχουν R διαδρομές μέσω των οποίων μπορεί να επιτευχθεί η σύνδεσή τους, όπου $R \geqslant 0$. Η ροή σε κάθε διαδρομή R που εξυπηρετεί μία ζήτηση είτε είναι μηδέν είτε έχει κόστος ίσο με το ελάχιστο κόστος μετακίνησης (π) γι'αυτό το ζεύγος προέλευσης-προορισμού. Ταυτόχρονα το κόστος κάθε διαδρομής ισούται με το ελάχιστο κόστος κάθε ζεύγους προέλευσης-προορισμού. Εάν q_{ri} ο φόρτος κάθε διαδρομής r_i με i=1,R τότε ισχύει το σύστημα :

$q_{ri}(c_{ri}-\pi)=0$, $i\epsilon\{1,R\}$ *//Ο φόρτος κατανέμεται σε διαδρομές r_i με κόστος ίσο με το ελάχιστο κόστος.*

$c_{ri} \geqslant \pi$, i=1,R *//Το κόστος κάθε διαδρομής από το A στο Z είναι τουλάχιστον ίσο με το ελάχιστο κόστος.*

$\sum q_{ri}=d$, i=1,R *//Ο φόρτος του δικτύου που κατανέμεται σε διαδρομές r_i με κόστος ίσο με το ελάχιστο δυνατό κόστος ισούται με τη συνολική ζήτηση για μετακίνηση A→Z.*

Εάν το δίκτυο έχει N κόμβους (κεντροειδή ζωνών) και M συνδέσμους (οδικοί άξονες), τότε θεωρώντας ότι το κόστος μετακίνησης μέσω κάθε συνδέσμου $j\epsilon M$ είναι συνάρτηση μόνο της φόρτισής του μπορεί να συμβολιστεί ως $c_j(q_j)$, δηλαδή ο σύνδεσμος j για φόρτιση q_j έχει κόστος μετακίνησης $c_j(q_j)$. Ο καταμερισμός της φόρτισης στο δίκτυο ώστε να έχουμε εξισορρόπηση του δικτύου ως προς το σύστημα επιλύεται από το παρακάτω πρόβλημα βελτιστοποίησης :

Εάν L οι σύνδεσμοι της διαδρομής r_i , i=1,R

Minimize $f(q)=\sum_{L\epsilon M} \int_0^{q_L} c_L(q_L)\ \partial q$

Με επιπλέον περιορισμό $\sum_{i=1}^{M} q\ \delta_i = q_L$, όπου η μεταβλητή δ_i παίρνει την τιμή 1 εάν ο σύνδεσμος i ανήκει στη διαδρομή r_i , αλλιώς παίρνει την τιμή 0.

2.4.ΜΗ ΑΠΟΔΟΤΙΚΟΣ ΚΑΤΑΜΕΡΙΣΜΟΣ ΤΗΣ ΜΕΤΑΦΟΡΙΚΗΣ ΖΗΤΗΣΗΣ-ΕΠΙΠΤΩΣΕΙΣ ΣΤΗΝ ΟΙΚΟΝΟΜΙΑ ΚΑΙ ΣΤΗΝ ΚΛΙΜΑΤΙΚΗ ΑΛΛΑΓΗ

Το πρόβλημα του καταμερισμού της ζήτησης συνδέεται άμεσα με την αύξηση της ίδιας της μεταφορικής ζήτησης και την αδυναμία ικανοποίησής της από τις διαθέσιμες υποδομές. Σε αυτήν τη θεματική ενότητα έχουν δημοσιευθεί μέχρι σήμερα διάφορες έρευνες που σχετίζονται με τα οικονομικά των μεταφορών. Στη συνέχεια παρουσιάζεται ένας πίνακας με την αύξηση των χρόνων καθυστέρησης ανά ημέρα από το 1982 έως το 1995 έπειτα από μελέτες που έγιναν από το Texas Transportation Institute σε 50 αστικές περιοχές των Η.Π.Α.

1982	1986	1990	1992	1993
7.26	10.344	12.581	13.572	14.15

Πίνακας 2.1.Ώρες καθυστέρησης σε εκατομμύρια κατά τη μελέτη 50 αστικών περιοχών. P.S.McCarthy (2001)

Όπως φαίνεται από τα παραπάνω στοιχεία οι χρόνοι καθυστέρησης διπλασιάστηκαν σε αυτές τις περιοχές κατά τη δεκαετία (1982-1993). Σύμφωνα πάλι με έρευνα του Texas Transport Institute (2002) στις 75 μεγαλύτερες μητροπολιτικές περιοχές των Η.Π.Α., λόγω της κυκλοφοριακής συμφόρησης προέκυψαν ζημιές 67,5 δισεκατομμυρίων δολαρίων στον τομέα της παραγωγικότητας. Επιπρόσθετα καταναλώθηκαν άσκοπα 21,6 δισεκατομμύρια γαλόνια βενζίνης και προέκυψαν 3,6 δισεκατομμύρια ώρες καθυστερήσεων ενώ το μέσο ετήσιο κόστος λόγω της κυκλοφοριακής συμφόρησης ανέρχεται σε 1000 δολάρια για κάθε χρήστη σε μεγάλες πόλεις και 200 δολάρια σε μικρότερες πόλεις.

Εκτός όμως από τα στοιχεία της οικονομίας επηρεάζεται και το περιβάλλον. Από περιβαλλοντική άποψη η κυκλοφοριακή συμφόρηση ευθύνεται σε ένα βαθμό για το φαινόμενο του θερμοκηπίου που οδηγεί στην κλιματική αλλαγή του πλανήτη και αποτελεί θέμα μείζονος σημασίας στην παγκόσμια περιβαλλοντική ατζέντα. Το κατά πόσο ευθύνεται η κυκλοφοριακή συμφόρηση για το φαινόμενο της κλιματικής αλλαγής είναι δύσκολο να υπολογιστεί με ακρίβεια λόγω της περιπλοκότητας του φαινομένου. Τα τελευταία χρόνια ωστόσο έχουν γίνει προσπάθειες προσμέτρησης της επιρροής κάθε διαδικασίας στο φαινόμενο του θερμοκηπίου με υπολογιστικούς τρόπους. Μία από αυτές τις προσπάθειες είναι η καθιέρωση μίας νέας μονάδας μέτρησης, του αποτυπώματος άνθρακα (carbon footprint). Το αποτύπωμα άνθρακα μπορεί να χρησιμοποιηθεί ευθέως και δίνει μία εικόνα για τις εκπομπές CO_2 που προκύπτουν από μία διαδικασία. Το αποτύπωμα άνθρακα όμως είναι ένα ενδεικτικό μέγεθος και υπό την περιβαλλοντική σκοπιά πρέπει να μελετηθεί συνολικά ο κύκλος του άνθρακα. Ωστόσο ο εύκολος υπολογισμός του, καθώς είναι ένα μέγεθος που εξαρτάται μόνο από τις εκπομπές CO_2, έχει βοηθήσει στην ευρεία διάδοσή του ειδικά στο Ηνωμένο Βασίλειο. Επειδή αυτός ο όρος έχει μελετηθεί τα τελευταία χρόνια ο ορισμός του δεν είναι σαφής και οι τρόποι υπολογισμού του έχουν μεγάλο εύρος, από

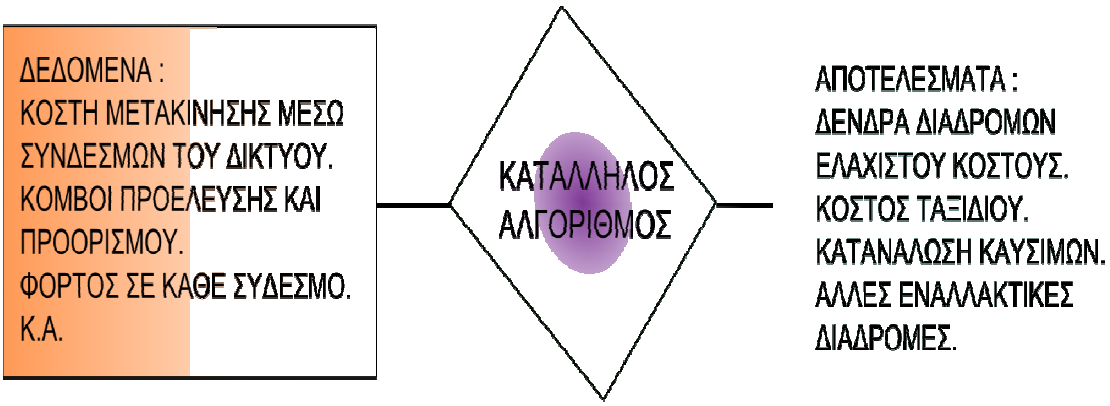
βασικούς υπολογισμούς που διατίθενται μέσω του διαδικτύου μέχρι πιο αναλυτικές μεθόδους που περιέχουν στοιχεία από τον κύκλο του άνθρακα. Οι διάφοροι τρόποι υπολογισμού του οφείλονται στο ότι επικρατεί σύγχυση σχετικά με αυτό που υπολογίζει στην πραγματικότητα. Πολλές μελέτες σχετικά με αυτό το θέμα παρουσιάστηκαν από το 2007 και έπειτα. Στην πλειοψηφία τους αναφέρουν ότι το αποτύπωμα άνθρακα είναι ένας όρος που χρησιμοποιείται ως συνώνυμο των εκπομπών CO_2 και των άλλων αερίων που προκαλούν το φαινόμενο του θερμοκηπίου τα οποία μετατρέπονται με τη σειρά τους σε ισοδύναμα CO_2. Οι Wiedmann and Minx (2007) όρισαν το αποτύπωμα άνθρακα ως μία μονάδα μέτρησης των εκπομπών CO_2 που προκαλούνται άμεσα ή έμμεσα από μία δραστηριότητα. Με τον ορισμό αυτό δε λαμβάνουν υπόψην τις εκπομπές άλλων αερίων που προκαλούν το φαινόμενο του θερμοκηπίου και πρότειναν έναν άλλο μέγεθος ¨climate footprint¨ στο οποίο περιλαμβάνεται και ο υπολογισμός αυτών των εκπομπών. Ένας άλλος τρόπος υπολογισμού του αποτυπώματος άνθρακα ενός προϊόντος που υιοθετήθηκε από την Carbon Trust, η οποία είναι μία ιδιωτική εταιρεία που ιδρύθηκε από τη Βρετανική κυβέρνηση, περιλαμβάνει τις εκπομπές CO_2 που οφείλονται σε άμεσες διαδικασίες που σχετίζονται με το προϊόν και εξαιρεί τις έμμεσες διαδικασίες (για παράδειγμα μεταφορά των υπαλλήλων στο εργοστάσιο παραγωγής του). Το αποτύπωμα άνθρακα μετριέται σε τόνους ισοδυνάμων διοξειδίου του άνθρακα (t CO_2 - eq) και αποτελεί ένα εύχρηστο μέγεθος που μπορεί να χρησιμοποιηθεί ώστε να πραγματοποιηθεί μία προσεγγιστική σύγκριση μεταξύ διάφορων διαδικασιών σχετικά πάντα με το πόσο ευθύνονται για το φαινόμενο του θερμοκηπίου.

Πάνω σε αυτές τις βάσεις πραγματοποιήθηκαν διάφορες έρευνες στη Μεγάλη Βρετανία σχετικά με το αποτύπωμα άνθρακα. Από την έρευνα των Gap et al. (2006) σχετικά με το αποτύπωμα άνθρακα σε σχολεία, προέκυψε ότι όλα τα σχολεία της Μεγάλης Βρετανίας είχαν αποτύπωμα άνθρακα 9,2 εκατομμυρίων (t CO_2 - eq) κατά το έτος 2001. Από αυτό μόνο το 26% οφειλόταν σε εκπομπές CO_2 για λόγους θέρμανσης και το 22% για λόγους ηλεκτροδότησης, ενώ το 20% οφειλόταν στις μεταφορές. Παρόμοια αποτελέσματα προέκυψαν και από την έρευνα των SEI et al. (2006). Η έρευνα αυτή υπολόγισε το αποτύπωμα άνθρακα των νοικοκυριών της Μεγάλης Βρετανίας κατά το έτος 2001. Κάθε νοικοκυριό είχε άμεσες και έμμεσες εκπομπές CO_2 της τάξης των 20,7 (t CO_2 - eq). Από αυτό οι άμεσες εκπομπές που περιλαμβάνουν τη θέρμανση και τις μεταφορές έφταναν το 70% με τις μεταφορές από μόνες τους να ευθύνονται για το 28% των εκπομπών CO_2.

Σύμφωνα με τα παραπάνω προκύπτουν κάποια αρχικά στοιχεία σχετικά με την επίδραση των μεταφορών στο φαινόμενο της κλιματικής αλλαγής. Οι μεταφορές αποτελούν μία από τις κύριες δραστηριότητες που οδηγούν σε μεγάλες εκπομπές CO_2 και μαζί με τον ηλεκτρισμό και τη θέρμανση ευθύνονται σχεδόν εξολοκλήρου για τις εκπομπές άνθρακα από ένα μέσο νοικοκυριό. Ένα σημαντικό μέρος όμως των εκπομπών CO_2 στις μεταφορές οφείλεται στην κυκλοφοριακή συμφόρηση. Το παραπάνω πρόβλημα είναι τόσο έντονο που ακόμα και με μικρές βελτιώσεις στον καταμερισμό της ζήτησης μπορεί να προκύψουν μεγάλα οφέλη.

2.5.Η ΧΡΗΣΙΜΟΤΗΤΑ ΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΣΤΟΝ ΚΑΤΑΜΕΡΙΣΜΟ ΤΗΣ ΖΗΤΗΣΗΣ ΔΙΚΤΥΩΝ

Πολλές εφαρμογές, μεταξύ των οποίων και τα συστήματα πλοήγησης χρησιμοποιούν αλγορίθμους για τον υπολογισμό μεγεθών σε συγκοινωνιακά δίκτυα. Οι αλγόριθμοι αυτοί υλοποιούνται μέσω Η/Υ και επιλύουν προβλήματα μέσα σε μικρό χρονικό διάστημα που θα ήταν σχεδόν αδύνατο να αντιμετωπιστούν με οποιονδήποτε άλλο τρόπο. Μία πολύ ενδιαφέρουσα κατηγορία αλγορίθμων για το συγκοινωνιολόγο μηχανικό είναι οι αλγόριθμοι που επιλύουν προβλήματα εύρεσης των συντομότερων διαδρομών μεταξύ δύο κόμβων (κεντροειδή ζωνών) προέλευσης-προορισμού. Σε ένα πυκνό δίκτυο όπου ο αριθμός των συνδέσμων είναι μεγάλος (εκατοντάδες ή χιλιάδες), ο αριθμός των εναλλακτικών διαδρομών γίνεται εξαιρετικά μεγάλος και η απαρίθμηση όλων των διαδρομών πρακτικά αδύνατη. Το γεγονός αυτό αποκλείει την εύρεση των συντομότερων διαδρομών χωρίς τη χρήση Η/Υ. Αυτή η παρατήρηση έχει επισημανθεί και από τον Oppenheim (1955). Από πολύ νωρίς αναζητήθηκαν διαφορετικές προσεγγίσεις για την επίλυση αυτών των προβλημάτων. Έτσι από τα πρώτα χρόνια χρήσης του Η/Υ πραγματοποιήθηκαν πολλές μελέτες από συγκοινωνιολόγους μηχανικούς αλλά και μηχανικούς ηλεκτρονικών υπολογιστών με στόχο την εύρεση αλγορίθμων οι οποίοι μπορούν να υλοποιηθούν μέσω Η/Υ και επιλύουν τέτοιου είδους προβλήματα σε πολύ μικρό χρόνο. Οι αλγόριθμοι αυτοί δέχονται δεδομένα τα οποία μπορεί να είναι φόρτοι, κόστη μετακίνησης, αποστάσεις και επιστρέφουν αποτελέσματα, όπως συντομότερες διαδρομές, χρόνοι ταξιδιού κ.α. πολλές φορές μέσα σε λίγα δευτερόλεπτα. Τα δεδομένα που δέχονται μπορεί να είναι είτε στατικά, είτε δυναμικά, δηλαδή να αλλάζουν συνεχώς τιμές μέσω ενός συστήματος πληροφοριών. Στο παρακάτω σχήμα παρουσιάζεται αυτή η διαδικασία :



Σχήμα 2.5.Γενικά στοιχεία λειτουργίας αλγορίθμων επιλογής διαδρομών

Οι αλγόριθμοι έχουν συμβάλει στον τομέα της επιλογής διαδρομής κατά την μετακίνηση, παρέχοντας στο χρήστη του δικτύου πληροφόρηση μέσα σε πολύ μικρό χρονικό διάστημα. Επίσης αποτελούν βοήθημα και για τον διαχειριστή του δικτύου που πρέπει να γνωρίζει τις συντομότερες διαδρομές του και πότε αυτές μεταβάλλονται, ώστε να μπορεί καθοδηγήσει αναλόγως τους χρήστες του δικτύου με διάφορα μέσα.

2.6.ΛΕΙΤΟΥΡΓΙΑ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

Σήμερα όλο και περισσότεροι χρήστες χρησιμοποιούν συστήματα πλοήγησης, τα οποία δεν αποτελούν πλέον είδος πολυτελείας. Τα συστήματα πλοήγησης προσφέρονται πλέον ως αξεσουάρ ακόμα και για οχήματα μεσαίας κατηγορίας. Τα συστήματα αυτά δίνουν τη δυνατότητα καθοδήγησης στο χρήστη μέσω ακουστικών και οπτικών μηνυμάτων με στόχο να τον οδηγήσουν στον προορισμό του. Ο χρήστης δεν έχει παρά να εισάγει μία πληροφορία σχετικά με τον προορισμό του σε αυτά. Οι πληροφορίες για κάθε δυνατό προορισμό, δηλαδή για τη μορφή του συγκοινωνιακού δικτύου, περιλαμβάνονται σε μία βάση δεδομένων που είναι αποθηκευμένη σε ένα DVD το οποίο χρησιμοποιείται από το σύστημα πλοήγησης. Σε αυτό το DVD περιέχονται και πληροφορίες σχετικά με σημεία ενδιαφέροντος (μνημεία, μουσεία, ξενοδοχεία, εμπορικά κέντρα κ.α.). Έτσι ο χρήστης μπορεί να ενημερωθεί για παράδειγμα για όλα τα ξενοδοχεία που βρίσκονται σε μία περιοχή. Τα μέρη από τα οποία αποτελείται ένα σύστημα πλοήγησης είναι :

α) Δύο δέκτες-αισθητήρες (gyro, tacho) που μέσω ενός δορυφόρου GPS και με τη χρήση ειδικών αλγορίθμων χρησιμοποιούνται για τον προσδιορισμό της θέσης του μεταφορικού μέσου.

β) Ο υπολογιστής που αποτελείται από ένα DVD player, ένα σκληρό δίσκο, έναν επεξεργαστή με συνήθη απόδοση 700MHz και μία οθόνη στην οποία απεικονίζεται ο ψηφιακός χάρτης του μεταφορικού δικτύου και μέσω αυτής παρέχεται οπτική πληροφορία και καθοδήγηση στο χρήστη.

γ) Μία σύνδεση GSM-phone ή ραδιοφωνική σύνδεση TMC-RDS που χρησιμοποιούνται για να λαμβάνονται ενημερώσεις σχετικά με τις αλλαγές στην κυκλοφορική ροή του μεταφορικού δικτύου.

Οι κύριες χρήσεις του συστήματος πλοήγησης είναι ο εντοπισμός της θέσης του μεταφορικού μέσου και η δρομολόγησή του στον τελικό προορισμό. Ο εντοπισμός της θέσης του γίνεται όπως αναφέρθηκε ήδη μέσω ενός δορυφόρου. Μετά τον εντοπισμό της θέσης του και την αποτύπωσή της στον ψηφιακό χάρτη ακολουθεί η επιλογή της διαδρομής που προτείνεται στον χρήστη. Η διαδρομή αυτή επιλέγεται μέσω ειδικών αλγορίθμων. Ο αλγόριθμος που χρησιμοποιείται συνήθως είναι ο αλγόριθμος Dijkstra (1959). Για να γίνει όσο το δυνατό πιο σύντομα η επιλογή της διαδρομής αυτής χρησιμοποιούνται πολλές τεχνικές. Για παράδειγμα εάν είναι γνωστές οι θέσεις προέλευσης και προορισμού γίνεται αναζήτηση στο δίκτυο της εγγύτερης περιοχής και όχι σε όλο το δίκτυο που είναι αποθηκευμένο στο ψηφιακό χάρτη. Όταν επιλεγεί η διαδρομή ο χρήστης ενημερώνεται μέσω οπτικής και ακουστικής καθοδήγησης. Φυσικά οι απαραίτητη πληροφόρηση πρέπει να παρέχεται στο χρήστη την κατάλληλη στιγμή ώστε να έχει το περιθώριο να αντιδράσει. Πολλές φορές μάλιστα ο χρήστης δεν ακολουθεί τη διαδρομή που προτείνεται και για το λόγο αυτό το σύστημα πλοήγησης αναγνωρίζει την πορεία που ακολουθεί και του

προτείνει νέα διαδρομή προς τον προορισμό του. Συνοψίζοντας, η λειτουργία του συστήματος πλοήγησης ακολουθεί τα στάδια :

α) Άνοιγμα της συσκευής
β) Αναμονή για τη λήψη σήματος μέσω δορυφόρου
γ) Επιλογή προορισμού διαδρομής
δ) Υπολογισμός διαδρομής
ε) Καθοδήγηση με εικονικές και φωνητικές οδηγίες

Κατά τη διαδικασία επιλογής διαδρομής μπορούν να χρησιμοποιηθούν διάφορα κριτήρια όπως ο χρόνος διάνυσης, η κυκλοφοριακή ροή, η ποιότητα του οδοστρώματος, το είδος των οδικών τμημάτων κ.α., τα οποία μπορούν να εκφραστούν μέσω μίας γενικευμένης συνάρτησης κόστους. Έτσι κάθε σύνδεσμος του δικτύου θα έχει ένα βάρος που προσδιορίζεται από την παραπάνω συνάρτηση. Με αυτό τον τρόπο κάθε χρήστης μπορεί να επιλέγει ανάμεσα σε μία ποικιλία διαδρομών όπως η συντομότερη, η πιο οικονομική, αυτή που θα έχει τη μικρότερη κυκλοφοριακή ροή κ.α. όπου σε κάθε περίπτωση μεταβάλλονται τα βάρη των συνδέσμων που λαμβάνονται υπόψη από τον αλγόριθμο κατά τον υπολογισμό της. Αλγόριθμοι για τη διαδικασία αυτή παρουσιάζονται στο Κεφάλαιο 5. Επίσης πρέπει να δίνεται η δυνατότητα στο χρήστη να επιλέγει μεταξύ εναλλακτικών διαδρομών εάν η διαδρομή που προτείνεται δεν τον ικανοποιεί. Αλγόριθμοι για τη διαδικασία αυτή παρουσιάζονται στην Ενότητα 5.6.

Η λειτουργία των συστημάτων πλοήγησης όμως δεν πρέπει να εξετάζεται μόνο από την πλευρά της βελτιστοποίησης της επιλεγόμενης διαδρομής για κάθε χρήστη προσωπικά. Αντίθετα πρέπει να διερευνηθεί η χρησιμότητά τους στις μεταφορές και το κατά πόσο μπορούν να συμβάλλουν στη μείωση των καθυστερήσεων, στην πιο αποδοτική χρήση της μεταφορικής υποδομής κ.α. Το γεγονός αυτό έχει γίνει αντιληπτό στο χώρο των συγκοινωνιολόγων και ήδη τα συστήματα πλοήγησης μαζί με άλλα ευφυή συστήματα μεταφορών (Intelligent Transportation Systems) έχουν ενταχθεί στην κοινή συγκοινωνιακή πολιτική των κρατών μελών της Ευρωπαϊκής Ένωσης. Προς αυτή την κατεύθυνση έχει βοηθήσει και το νέο δορυφορικό σύστημα που προσδιορίζει τη θέση ενός μέσου και έχει αναπτυχθεί υπό την αιγίδα της Ευρωπαϊκής Ένωσης, γνωστό ως Galileo. Επίσης έχουν πραγματοποιηθεί και διάφορες έρευνες σε αυτό τον τομέα όπως ένα σχετικό πείραμα στο Τορίνο στο οποίο με πληροφόρηση κάθε οδηγού από ευφυή συστήματα μεταφορών μειώθηκε ο χρόνος ταξιδίου κατά 20% και αυτό είχε θετικά αποτελέσματα και για τις δημόσιες συγκοινωνίες όπου ο χρόνος ταξιδιού μειώθηκε κατά 3% αντίστοιχα. Τα αποτελέσματα αυτά παρουσιάστηκαν στο : "EXTRA consortium for DG Energy and Transport, Results from the Transport Research Programme, European Commission, 2010".

ΚΕΦΑΛΑΙΟ 3ο

ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

3.1.ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό πραγματοποιείται βιβλιογραφική ανασκόπηση των κυριότερων μεθόδων που έχουν προταθεί και μπορούν να χρησιμοποιηθούν έμμεσα ή άμεσα από τα συστήματα πλοήγησης με στόχο τη μείωση των χρόνων μετακίνησης και την προστασία του περιβάλλοντος.

Στη δεύτερη ενότητα του κεφαλαίου παρουσιάζονται μέθοδοι που μπορούν να χρησιμοποιηθούν άμεσα από ένα σύστημα πλοήγησης. Όπως έχει αναφερθεί ήδη τα συστήματα πλοήγησης χρησιμοποιούν συνήθως κάποια τροποποιημένη έκδοση του αλγορίθμου Dijkstra (1959) για τον υπολογισμό της συντομότερης διαδρομής, η οποία στη συνέχεια προτείνεται στο χρήστη. Εκτός από αυτό τον αλγόριθμο όμως έχουν προταθεί και άλλοι οι οποίοι είναι εξίσου σημαντικοί και παρουσιάζονται συνοπτικά. Επίσης πραγματοποιείται ανασκόπηση και σε αλγόριθμους που μπορούν να χρησιμοποιηθούν από τα συστήματα πλοήγησης για άλλες λειτουργίες όπως :
α) Την εύρεση όχι μόνο της συντομότερης διαδρομής αλλά και των k πρώτων συντομότερων διαδρομών μεταξύ των κεντροειδών.
β) Την επιλογή της συντομότερης διαδρομής σε συγκοινωνιακά δίκτυα όπου οι χρόνοι διάνυσης των συνδέσμων δεν είναι γνωστοί ή μεταβάλλονται με το χρόνο.
γ) Την επιλογή της συντομότερης διαδρομής η οποία ικανοποιεί ταυτόχρονα και κάποιους περιορισμούς. Οι περιορισμοί αυτοί μπορεί να είναι η μειωμένη κατανάλωση καυσίμου, ο περιορισμός της ατμοσφαιρικής ρύπανσης κατά τη μετακίνηση κ.α.

Στην τρίτη ενότητα πραγματοποιείται ανασκόπηση των ερευνών που σχετίζονται με το βέλτιστο καταμερισμό της ζήτησης σε στατικά συγκοινωνιακά δίκτυα. Επίσης αναφέρονται έρευνες που προτείνουν τρόπους δρομολόγησης των μεταφορικών μέσων ώστε να περιοριστούν οι οικονομικοί και ενεργειακοί πόροι που καταναλώνονται άσκοπα.

Στην τέταρτη ενότητα πραγματοποιείται ανασκόπηση των μελετών που σχετίζονται με το βέλτιστο καταμερισμό της ζήτησης σε δυναμικά δίκτυα. Επίσης πραγματοποιείται ανασκόπηση των μεθόδων που μπορούν να χρησιμοποιηθούν άμεσα από ένα σύστημα πλοήγησης ή έμμεσα από κάποιο κέντρο διαχείρισης με στόχο την αποδοτική δρομολόγηση των μεταφορικών μέσων σε δυναμικά δίκτυα, στα οποία επικρατούν συνθήκες κυκλοφοριακής συμφόρησης. Οι έρευνες σε αυτό τον τομέα ξεκίνησαν τα τελευταία χρόνια και η χρησιμότητα των συστημάτων πλοήγησης σε αυτά τα δίκτυα αποτελεί θέμα προς διερεύνηση.

3.2.ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΤΗ ΔΡΟΜΟΛΟΓΗΣΗ ΤΟΥ ΧΡΗΣΤΗ ΣΕ ΣΥΓΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ

Leonhard Euler (1736) στην εργασία του *Solutio problematis ad geometriam situs pertinentis,* αναζητούσε μονοπάτι ώστε να μπορέσει κάποιος να διασχίσει και τις επτά γέφυρες της πόλης Königsberg μόνο μία φορά και να επιστρέψει στο σημείο εκκίνησης (Euler's circuit). Στο ερώτημα αυτό απέδειξε ότι δεν υπάρχει τέτοιο μονοπάτι. Το σκεπτικό του ήταν ότι όλη η πόλη μπορούσε να μοντελοποιηθεί σε ένα γράφημα και κάθε γέφυρα να αποτελεί έναν σύνδεσμο. Η μοντελοποίηση της πόλης σε γράφημα και η θεώρηση κάθε γέφυρας ως σύνδεσμο τον βοήθησε στην επίλυση του προβλήματος. Η μέθοδός του έθεσε τα θεμέλια στη θεωρία των γράφων (graph theory) και πάνω σε αυτή βασίστηκαν οι πρώτοι αλγόριθμοι εύρεσης ελάχιστης διαδρομής.

E.F.Moore (1957) στην εργασία του *The shortest path throw a maze*, πρότεινε έναν αλγόριθμο που υπολογίζει τις ελάχιστες διαδρομές από ένα συγκεκριμένο κόμβο του γραφήματος προς όλους τους υπόλοιπους κόμβους (Single Source Shortest Path Problem) και χρησιμοποιείται ακόμα και αν οι σύνδεσμοι έχουν αρνητικά βάρη. Ο χρόνος εκτέλεσής του είναι $O(\mathrm{M}\times\mathrm{N})$ όπου M ο αριθμός των συνδέσμων και N ο αριθμός των κόμβων στο δίκτυο.

R.Bellman (1958) και L.R.Ford, Fulkerson (1962) στις εργασίες τους *On a Routing Problem* και *Flows in Networks*, πρότειναν έναν αλγόριθμο παρόμοιο με αυτόν του Moore. Πολλές φορές στη διεθνή βιβλιογραφία μάλιστα αναφέρονται ως ο ίδιος αλγόριθμος.

E.W.Dijkstra (1959) στην εργασία του *A Note on Two Problems in Connexion with Graphs,* πρότεινε έναν αλγόριθμο εύρεσης ελάχιστων διαδρομών από έναν κόμβο προς όλους τους υπόλοιπους κόμβους του δικτύου (SSSPP), που ανήκει στην κατηγορία των άπληστων αλγορίθμων και αποτελεί έναν από τους βασικότερους αλγόριθμους αυτής της κατηγορίας με ευρεία εφαρμογή μέχρι σήμερα. Ο χρόνος εκτέλεσής του στη βασική του μορφή είναι $O(N^2)$ ή $O(\mathrm{N}\times\mathrm{M})$ ανάλογα με τη μέθοδο αναπαράστασης του δικτύου.

W.Hoffman, R. Pavley (1959) στην εργασία τους *A Method for the Solution of the Nth Best Path Problem,* πρότειναν έναν αλγόριθμο για την εύρεση της $2^{\eta\varsigma}$ συντομότερης διαδρομής μεταξύ ενός κόμβου προέλευσης και ενός κόμβου προορισμού. Ο αλγόριθμος αυτός χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Dijkstra και ο χρόνος εκτέλεσης του είναι $O(\ \mathrm{k}\times(\mathrm{N}+\mathrm{M})logN))$, όπου k ο αριθμός των κόμβων από τους οποίους αποτελείται η συντομότερη διαδρομή.

B.Roy (1959) και S.Warshall (1962) και R.Floyd (1962) στις εργασίες τους *Transitivité et connexité* και *A theorem on Boolean matrices* και *Algorithm 97: Shortest Path,* πρότειναν έναν αλγόριθμο βασιζόμενο στο δυναμικό προγραμματισμό με εφαρμογή στα προβλήματα εύρεσης των ελάχιστων διαδρομών από όλους τους

κόμβους του δικτύου προς όλους τους κόμβους του δικτύου (All Pairs Shortest Path Problems) με χρόνο εκτέλεσης O(N^3).

M.Pollack, W.Wiebenson (1960) στην εργασία τους *Solution of the shortest route problem-a review*, αναφέρθηκαν στις μεθόδους υπολογισμού ελάχιστων διαδρομών από έναν κόμβο προέλευσης προς τους υπόλοιπους κόμβους του δικτύου.

G.B. Dantzig, P.Wolfe (1960) στην εργασία τους *Decomposition Principle for Linear Programs*, πρότειναν μία τεχνική τον πιο αποδοτικό υπολογισμό γραμμικών προβλημάτων που ικανοποιούν μία συγκεκριμένη ιδιότητα. Η ιδιότητα αυτή επιτρέπει στο κύριο πρόβλημα να διασπαστεί σε επιμέρους γραμμικά υποπροβλήματα και η επίλυση των υποπροβλημάτων αυτών οδηγεί στην επίλυση του βασικού προβλήματος.

R.Bellman, R.Kalaba (1960) στην εργασία τους *On Kth best policies,* πρότειναν έναν αλγόριθμο με εφαρμογή στο πρόβλημα εύρεσης των *k* συντομότερων διαδρομών από έναν κόμβο προέλευσης προς ένα κόμβο προορισμού.

G.B.Dantzig (1960) στην εργασία του, *On the shortest route through a network,* πρότεινε έναν αλγόριθμο για την εύρεση των συντομότερων διαδρομών από έναν κόμβο προς όλους τους υπόλοιπους κόμβους ενός δικτύου, ο οποίος ολοκληρώνεται μετά από το πολύ N(N-1)/2 συγκρίσεις. Ο αλγόριθμος αυτός αποτελεί παραλλαγή του αλγορίθμου του E.F.Moore (1957).

M.Pollack (1961) στην εργασία του *the Kth best route through a network,* παρουσίασε την δομή των προβλημάτων σχετικά με την εύρεση των *k* συντομότερων διαδρομών από έναν κόμβο προέλευσης σε ένα κόμβο προορισμού και έδωσε έναν αλγόριθμο.

P.E.Hart, N.J.Nilsson, B.Raphael (1968) στην εργασία τους *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, πρότειναν ένα αλγόριθμο γνωστό ως A* ή Astar ο οποίος εφαρμόζεται σε προβλήματα εύρεσης της ελάχιστης διαδρομής μεταξύ ενός κόμβου προέλευσης και ενός κόμβου προορισμού (Point to Point Shortest Path Problems). Η δομή του είναι παρόμοια με αυτή του αλγορίθμου Dijkstra μόνο που στα δεδομένα του δίνεται και η αναμενόμενη απόσταση από κάθε κόμβο προς τον κόμβο προορισμού ώστε ο αλγόριθμος να κατευθύνεται συνεχώς προς τον προορισμό και να περιορίζει την αναζήτησή του στο δίκτυο. Ο χρόνος εκτέλεσής του εξαρτάται από το κατά πόσο ισχύουν οι αναμενόμενες αποστάσεις από κάθε κόμβο προς τον κόμβο προορισμού που έχουν δοθεί στα δεδομένα.

H.Frank (1969) στην εργασία του *Shortest paths in probabilistic graphs,* μελετά το πρόβλημα εύρεσης των συντομότερων διαδρομών σε δίκτυα όπου τα βάρη των συνδέσμων μεταβάλλονται, ακολουθώντας κάποια πιθανοτική κατανομή. Η μελέτη γίνεται μέσω στατιστικών μεθόδων και προτείνεται ένας αλγόριθμος για τον υπολογισμό των διαδρομών.

S.Dreyfus (1969) στην εργασία του *An appraisal of some shortest paths algorithms,USA Air Force project RAND*, πρότεινε έναν αλγόριθμο με εφαρμογή στο πρόβλημα εύρεσης των *k* συντομότερων διαδρομών από όλους τους κόμβους του δικτύου προς έναν κόμβο προορισμού. Απέδειξε επίσης ότι ο χρόνος εκτέλεσής του ήταν αρκετά μικρότερος από τον αντίστοιχο των R.Bellman, R.Kalaba. Ο χρόνος αυτός είναι $O(\mathrm{k} \times N^2)$, όπου k ο αριθμός των ελάχιστων διαδρομών που υπολογίζονται. Στην εργασία αυτή επίσης ασχολήθηκε με όλα τα θέματα εύρεσης διαδρομών και παρουσίασε τις κυριότερες μεθόδους που είχαν αναπτυχθεί μέχρι τότε.

J.Y.Yen (1971) και (1972) στις εργασίες του *Finding the K shortest loopless paths in a network* και *Another algorithm for finding the K shortest-loopless network paths*, πρότεινε αλγορίθμους για την επίλυση στο πρόβλημα εύρεσης των *k* συντομότερων διαδρομών από έναν κόμβο προέλευσης προς έναν κόμβο προορισμού που χρησιμοποιούνται ακόμα και σήμερα. Οι διαδρομές αυτές δεν περιέχουν κυκλικό βρόγχο, δηλαδή δε μπορεί να υπάρχει δύο φορές ο ίδιος κόμβος σε μία διαδρομή.

E.L.Lawler (1972) στην εργασία του *A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem*, βελτίωσε το χρόνο εκτέλεσης του αλγορίθμου του Yen χρησιμοποιώντας βελτιωμένες δομές δεδομένων σε $O(\mathrm{k} \times \mathrm{N}(\mathrm{M}+\mathrm{N}(\mathrm{N} \times \log \mathrm{N}))$.

U.Pape (1974) στην εργασία του *Implementation and Efficiency of Moore - Algorithms for the Shortest Path Problem*, πρότεινε μία νέα δομή δεδομένων για την υλοποίηση του αλγορίθμου του Moore. Θεωρητικά ο χρόνος υλοποίησης του νέου αλγορίθμου ήταν εκθετικός και πολύ μεγαλύτερος από αντίστοιχο του αλγόριθμου του Moore αλλά σε πραγματικές εφαρμογές ο χρόνος αυτός ήταν πολύ μικρός.

B.L.Fox (1975) στην εργασία του *k-th shortest paths and applications to the probabilistic networks,* ανέπτυξε έναν αλγόριθμος με εφαρμογή στο πρόβλημα εύρεσης των *k* συντομότερων διαδρομών από έναν κόμβο προέλευσης προς ένα κόμβο προορισμού ο οποίος χρησιμοποιεί τον αλγόριθμο Dijkstra με βελτιωμένες δομές δεδομένων στην ουρά αναμονής των κόμβων. Ο χρόνος εκτέλεσής του είναι $O(\mathrm{M}+\mathrm{k} \times \mathrm{N} \times \log \mathrm{N})$.

P.B.Mirchandani (1976) στην εργασία του *Shortest distance and reliability of probabilistic networks,* πρότεινε έναν αλγόριθμο που επιλύει το πρόβλημα εύρεσης της διαδρομής ελαχίστου κόστους από έναν κόμβο σε έναν άλλο κόμβο του δικτύου (PPSPP), όταν το βάρος του κάθε συνδέσμου ακολουθεί μία ανεξάρτητη πιθανοτική μεταβολή. Ο αλγόριθμος αυτός υποθέτει ότι είναι γνωστές όλες οι εναλλακτικές διαδρομές μεταξύ δύο κόμβων.

D.B.Johnson (1977) στην εργασία του *Efficient algorithms for shortest paths in sparse networks,* πρότεινε έναν αλγόριθμο που χρησιμοποιείται στα προβλήματα εύρεσης ελάχιστων διαδρομών από όλους τους κόμβους του δικτύου προς όλους τους κόμβους του δικτύου (All Pairs Shortest Path Problems). Δέχεται αρνητικά βάρη στους συνδέσμους, αρκεί βέβαια οι σύνδεσμοι με αρνητικά βάρη να μη σχηματίζουν βρόγχο που δεσμεύει την επίλυση του προβλήματος. Ο χρόνος εκτέλεσής του μπορεί

να είναι μικρότερος από τον αντίστοιχο του Floyd-Warshall σε ειδικές περιπτώσεις αραιών δικτύων.

J.S.Croucher (1978) στην εργασία του *A note of stochastic shortest-route problem*, ανέπτυξε έναν αλγόριθμο εύρεσης των ελάχιστων διαδρομών σε στοχαστικά δίκτυα όπου τα βάρη των συνδέσμων δεν είναι γνωστά μέχρι τη στιγμή που ο χρήστης φθάνει σε έναν κόμβο που συνδέεται άμεσα μαζί τους (Stochastic Shortest Route Problems). Η κύρια ιδέα του αλγορίθμου είναι ότι λόγω της αβεβαιότητας δεν πρέπει να αναζητείται από την αρχή το ολικό βέλτιστο, αλλά να αναζητείται μία στρατηγική που οδηγεί σταδιακά σε αυτό.

R.B.Dial, F. Glover, D.Karney, D.Klingman (1979) στην εργασία τους *A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees,* μελέτησαν μέσα από εφαρμογές και συγκρίσεις τις μεθόδους προσδιορισμού των δένδρων ελαχίστων διαδρομών.

G.Handler, I.Zang (1980) στην εργασία τους *A Dual Algorithm for the Constrained Shortest Path Problem,* πρότειναν την τεχνική των πολλαπλασιαστών Lagrange που απλοποίησε την επίλυση των προβλημάτων εύρεσης της συντομότερης διαδρομής η οποία ταυτόχρονα πρέπει να πληροί και ορισμένους περιορισμούς διαθέσιμων πόρων. Η μέθοδος που πρότειναν περιοριζόταν σε εφαρμογές που υπήρχε ένας περιορισμός διαθέσιμων πόρων.

N.Katoh, T.Ibaraki, H.Mine (1982) στην εργασία τους *An efficient algorithm for k shortest simple paths,* πρότειναν έναν αλγόριθμο που βελτιώνει τον αλγόριθμο του Yen (1971) σε δίκτυα στα οποία οι σύνδεσμοι δεν έχουν κατεύθυνση (undirected graphs) με χρόνο υλοποίησης $O(k(M+N \times \log N))$.

M.Fredman, R. Tarjan (1987) στην εργασία τους *Fibonacci heaps and their uses in improved network optimization algorithms,* πρότειναν μία νέα δομή δεδομένων, γνωστή και ως σωρός Fibonacci, για την υλοποίηση της ουράς προτεραιότητας στον αλγόριθμο Dijkstra μειώνοντας το χρόνο εκτέλεσής του σε $O(N+M \times \log N)$.

A.Mofat, T.Takaoka (1987) στην εργασία τους *An all pairs shortest path algorithm with expected time O(n2log n)*, πρότειναν έναν νέο αλγόριθμο επίλυσης του προβλήματος εύρεσης των συντομότερων διαδρομών από όλους τους κόμβους του δικτύου προς όλους τους κόμβους του δικτύου (APSPP) με βελτιωμένο χρόνο εκτέλεσης $O(N^2 \times \log N)$.

G.Gallo (1986) και S.Pallottino (1988) στις εργασίες τους *Shortest Path Methods, A Unified Approach* και *Shortest Path Algorithms*, πρότειναν ένα νέο αλγόριθμο επίλυσης του (SSSPP) που αποτελεί βελτίωση του αλγορίθμου Pape χρησιμοποιώντας δύο ουρές προτεραιότητας με θεωρητικό χρόνο υλοποίησης $O(M \times N^2)$ που αποδεικνύεται πολύ μικρότερος στην πράξη. Στις ουρές προτεραιότητας αποθηκεύονται οι κόμβοι του δικτύου, οι οποίοι επιλέγονται σε κάθε επανάληψη σύμφωνα με κάποιους κανόνες.

G.Andreatta, L.Romeo (1988) στην εργασία τους *Stochastic shortest path problems with recource,* ασχολήθηκαν με την επίλυση του προβλήματος εύρεσης της διαδρομής ελαχίστου κόστους μεταξύ δύο κόμβων του δικτύου (PPSPP), όπου το

βάρος των συνδέσμων είναι άγνωστο μέχρι το μεταφορικό μέσο να φθάσει σε έναν κόμβο που συνδέεται άμεσα με αυτούς.

M.Desrochers (1988) στην εργασία *An algorithm for the shortest path problem with resource constraints*, πρότεινε έναν αλγόριθμο δυναμικού προγραμματισμού για την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής με περιορισμούς διαθέσιμων πόρων.

J.E.Beasly, N.Christofides (1989) στην εργασία *An algorithm for the resource constrained shortest path problem*, πρότειναν έναν ευρετικό αλγόριθμο για την εύρεση της συντομότερης διαδρομής σε δίκτυα με περιορισμούς διαθέσιμων πόρων που βασίζεται στην τεχνική των πολλαπλασιαστών Lagrange με την οποία όλοι οι περιορισμοί μεταφέρονται στη βασική συνάρτηση και πολλαπλασιάζονται με τους αντίστοιχους πολλαπλασιαστές. Ο αλγόριθμος αυτός επέκτεινε την τεχνική των G.Handler, I.Zang (1980) και σε εφαρμογές με πολλούς περιορισμούς διαθέσιμων πόρων. Έτσι μειώνεται σημαντικά ο χρόνο εκτέλεσης αλλά η διαδρομή που προτείνεται τελικά δεν είναι η βέλτιστη καθώς η διαδικασία που ακολουθείται είναι προσεγγιστική.

C.H.Papadimitriou, M.Yannakakis (1991) στην εργασία τους *Shortest paths without a map*, πρότειναν μία μέθοδο για την εύρεση της συντομότερης διαδρομής σε δίκτυα όπου οι χρόνοι διάνυσης των συνδέσμων δεν είναι γνωστοί αρχικά. Το πρόβλημα αναφέρεται σε δίκτυα με αβεβαιότητες που αίρονται καθώς το μεταφορικό μέσο περιηγείται σε αυτά και αναζητούνται οι κανόνες που πρέπει να τηρούνται δυναμικά, δηλαδή σε κάθε επιμέρους βήμα, με στόχο τη δημιουργία στρατηγικής για την επιλογή της βελτιστης διαδρομής.

M.Desrochers, J.Desrosiers, M.Solomon (1992) στην εργασία τους *A new optimization algorithm for the vehicle routing problem with time windows*, πρότειναν ένα νέο αλγόριθμο για την εύρεση της διαδρομής που για τη δρομολόγηση μεταφορικών μέσων με περιορισμούς στο χρόνο έναρξης της διαδρομής και στο χρόνο τερματισμού της.

D.Bertsekas (1993) στην εργασία του *A Simple and Fast Label Correcting Algorithm for Shortest Paths*, πρότεινε έναν αλγόριθμο εύρεση της συντομότερης διαδρομής από έναν κόμβο προέλευσης σε έναν κόμβο προορισμού, βασιζόμενο στις μεθόδους δυναμικού προγραμματισμού. Επίσης παρουσίασε τις κυριότερες μεθόδους δυναμικού προγραμματισμού που είχαν προταθεί για την επίλυση του (SSSPP) μέχρι τότε.

M.Fredman, D.Willard (1993) στην εργασία τους *Surpassing the information theoretic bound with fusion trees*, πρότειναν μία βελτιωμένη δομή δεδομένων για τον αλγόριθμο Dijkstra. Με τη δομή αυτή ο χρόνος εκτέλεσής του μειώθηκε σε $O(\mathrm{M}\sqrt{logN})$.

M.Fredman, D.Willard (1994) στην εργασιά τους *Trans-Dichotomous algorithms for minimum spanning trees and shortest paths*, πρότειναν μία ακόμα δομή δεδομένων που μειώνει και άλλο το χρόνο εκτέλεσης του αλγορίθμου Dijkstra σε $O(\mathrm{M+N} \times \mathrm{logN/log(logN)})$.

J.Desrosiers, Y.Dumas, M.Solomon, F.Soumis (1995) στην εργασία τους *Time Constrained Routing and Scheduling,* μελέτησαν το πρόβλημα της δρομολόγησης υπό χρονικούς περιορισμούς. Η έρευνά τους χρησιμοποιήθηκε και για το πρόβλημα της επιλογής βέλτιστης διαδρομής σε δίκτυα με περιορισμούς διαθέσιμων πόρων.

M.Thorup (1996) στην εργασία του *On RAM priority queues*, πρότεινε μία νέα δομή δεδομένων για την επίλυση του (SSSPP) σε χρόνο $O(\mathrm{M} \times \log(\log \mathrm{N}))$.

G.H.Polychronopoulos, J.N.Tsitsiklis (1996) στην εργασία τους *Stochastic shortest path problems with recource*, ασχολήθηκαν με την επίλυση του προβλήματος εύρεσης της ελάχιστης διαδρομής από έναν κόμβο σε έναν άλλο κόμβο του δικτύου (PPSPP), όταν τα βάρη των συνδέσμων δεν είναι γνωστά αρχικά, αλλά στη συνέχεια καθώς το μεταφορικό μέσο διασχίζει το δίκτυο και προτείνουν έναν αλγόριθμο δυναμικού προγραμματισμού για την επιλογή διαδρομής.

G.Ramalingam, T.Reps (1996) στην εργασιά τους *An Incremental Algorithm for the generalization of the Shortest Path Problem*, πρότειναν έναν αλγόριθμο δυναμικού προγραμματισμού που χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Dijkstra για την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής σε δυναμικά δίκτυα όπου τα βάρη των συνδέσμων μεταβάλλοται συχνά και απαιτούνται πολλοί επανυπολογισμοί.

D.Frigioni, A.Marchetti-Spaccamela, U.Nani (1996) στην εργασία τους *Fully Dynamic Output Bounded Single Source Shortest Path Problem*, πρότειναν έναν αλγόριθμο για την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής σε δυναμικά δίκτυα με βελτιωμένους χρόνους εκτέλεσης σε σύγκριση με τη μέθοδο των G.Ramalingam, T.Reps (1996).

R.Raman (1997) στην εργασία του *Recent results on the single-source shortest paths problem*, πρότεινε έναν νέο αλγόριθμο για την επίλυση του προβλήματος εύρεσης των ελάχιστων διαδρομών από έναν κόμβο του δικτύου προς όλους τους υπόλοιπους κόμβους. Ο χρόνος εκτέλεσής του είναι $O(\mathrm{M+N} \times log^{1/3+e}N)$.

L.Fu, L.R.Rillet (1997) στην εργασία τους *Expected shortest paths in dynamic and stochastic traffic networks,* μελετούν το πρόβλημα εύρεσης διαδρομών ελαχίστου κόστους σε συγκοινωνιακά δίκτυα όπου τα βάρη των συνδέσμων δίνονται με πιθανοτικό τρόπο(στοχαστικά) και μεταβάλλονται στο χρόνο(δυναμικά) που ανήκει στην κατηγορία Dynamic and Stochastic Shortest Path Problem. Προτείνουν έναν ευρετικό αλγόριθμο από την κατηγορία εύρεσης των *k* συντομότερων διαδρομών μεταξύ δύο κόμβων σε δίκτυο ο οποίος μπορεί να χρησιμοποιηθεί για τις απαιτήσεις των συστημάτων πλοήγησης.

N.Alon, Z.Galil, O.Margalit (1997) στην εργασία τους *On the Exponent of the All Pairs Shortest Path Problem,* πρότειναν ένα νέο αλγόριθμο για τον υπολογισμό των συντομότερων διαδρομών από κάθε κόμβο του δικτύου προς κάθε κόμβο του δικτύου με χρόνο εκτέλεσης $O((\mathrm{Mn})^{\nu} log^3 n)$.

B.Cherkassky, A.Goldberg, C.Silverstein (1997) στην εργασία τους *Buckets, Heaps, Lists and monotone priority queues*, πρότειναν έναν αλγόριθμο για την επίλυση του (SSSPP). Ο θεωρητικός χρόνος εκτέλεσής του είναι $O(N \times (\log C)^{\frac{1}{3}+\varepsilon}+M)$, όπου C : το κόστος του συνδέσμου με το μεγαλύτερο βάρος.

D.Eppstein (1998) στην εργασία του *Finding the k shortest paths*, πρότεινε έναν αλγόριθμο με εφαρμογή στο πρόβλημα εύρεσης των *k* συντομότερων διαδρομών από έναν κόμβο προέλευσης προς έναν κόμβο προορισμού. Ο χρόνος εκτέλεσής του είναι $O(M+N \times \log N+k)$. Ο αλγόριθμος αυτός έχει ευρεία χρήση και είναι ο ταχύτερος της κατηγορίας του, βελτιώνοντας το χρόνο εκτέλεσης του αλγορίθμου B.L.Fox (1985).

E.Q.V.Martins, M.M.B.Pascoal, J.L.E. DosSantos (1998) στην εργασία τους *Deviation Algorithms for ranking shortest paths*, αναφέρθηκαν στις μεθόδους εύρεσης των k συντομότερων διαδρομών και των k συντομότερων διαδρομών που δεν περιέχουν κυκλικούς βρόγχους. Πρότειναν επίσης και μία γενικευμένη μέθοδο που στηρίζεται στον αλγόριθμο του Yen (1971) με εφαρμογές και στην εύρεση διαδρομών που περιέχουν κυκλικούς βρόγχους.

I.Chabini (1998) στην εργασία του, *Discrete Dynamic Shortest Path Problems in Transportation Applications.Complexity and Algorithms with Optimal Run Time,* πρότεινε έναν αλγόριθμο για την επίλυση του προβλήματος (APSPP) θεωρώντας ότι τα βάρη των συνδέσμων μεταβάλλονται δυναμικά σε διακεκριμένους χρόνους. Στην περίπτωση δηλαδή που η υπολογιστική μηχανή αναβαθμίζει ταυτόχρονα όλα τα δεδομένα της ανά κάποιο συγκεκριμένο χρονικό διάστημα.

K.Mehlhorn, M.Ziegelmann (2000) στην εργασία *Resource Constrained Shortest Paths*, έκαναν νέες προτάσεις για τον υπολογισμό της συντομότερης διαδρομής υπό περιορισμούς διαθέσιμων πόρων. Στις προτάσεις τους περιλαμβάνεται και η χρήση της ελλειψοειδούς μεθόδου για τον προσδιορισμό των πολλαπλασιαστών Lagrange ώστε ο χρόνος εκτέλεσης να μετατραπεί σε πολυωνυμικό.

K.Mehlhorn, M.Ziegelmann (2001) στην εργασία τους *CNOP – A Package for Constrained Network Optimization,* ανέπτυξαν ένα πρόγραμμα σε γλώσσα C++ για την επιλογή της συντομότερης διαδρομής σε δίκτυα με περιορισμούς διαθέσιμων πόρων βασιζόμενοι στην εργασία των Beasly and Christofides (1989).

Y.Han, M.Thorup (2002) στην εργασία τους *Integer sorting in $O(n\sqrt{\log\log N})$ expected time and linear space,* πρότειναν ένα νέο αλγόριθμο για την επίλυση του προβλήματος εύρεσης των ελάχιστων διαδρομών από έναν κόμβο του δικτύου προς όλους τους υπόλοιπους κόμβους που θεωρητικά έχει γραμμικό χρόνο εκτέλεσης.

E.Q.V.Martins, M.M.B.Pascoal (2003) στην εργασία τους *A new implementation of Yen's ranking loopless paths algorithm*, πρότειναν ένα νέο αλγόριθμο για την επίλυση του προβλήματος εύρεσης των k συντομότερων διαδρομών μεταξύ δύο κόμβων προέλευσης προορισμού που ανήκει στην κατηγορία των αλγορίθμων διαχωρισμού των εναλλακτικών διαδρομών από τη συντομότερη διαδρομή.

I.Dumitresu, N.Boland (2003) στην εργασία *Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem*, έκαναν έναν απολογισμό των μεθόδων εύρεσης της συντομότερης διαδρομής διαδρομής υπό περιορισμούς διαθέσιμων πόρων μέσω της χρήσης των πολλαπλασιαστών Lagrange.

P.Avela, M.Boccia, A.Sforza (2004) στην εργασία τους *Resource Constrained Shortest Path Problems in Path Planning for Fleet Management,* πρότειναν ένα νέο τύπο προβλημάτων για την εύρεση ελάχιστων διαδρομών σε δίκτυα με περιορισμούς διαθέσιμων πόρων.

D.Feillet, P.Dejax, M.Gendreau, C.Gueguen (2004) στην εργασία τους *An exact algorithm for the elementary resource constrained shortest path problem*, πρότειναν έναν αλγόριθμο δυναμικού προγραμματισμού για την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής υπό περιορισμούς διαθέσιμων πόρων.

S.Irnich, G.Desaulniers (2004) στην εργασία τους *Shortest path problems with resource constraints*, έκαναν έναν απολογισμό των μεθόδων επίλυσης στο πρόβλημα εύρεσης της συντομότερης διαδρομής υπό περιορισμούς διαθέσιμων πόρων με ιδιαίτερη αναφορά στις μεθόδους δυναμικού προγραμματισμού.

T.Takaoka (2005) στην εργασία του *An $O(n^3 loglogn/logn)$ time algorithm for the all-pairs shortest path problem,* πρότεινε έναν νέο αλγόριθμο επίλυσης του προβλήματος εύρεσης των συντομότερων διαδρομών από όλους τους κόμβους του δικτύου προς όλους τους κόμβους του δικτύου (APSPP) με χρόνο εκτέλεσης $O((N^3 \times \text{loglogN})/\text{logN})$.

P.Sanders, D.Schultes (2005) στην εργασία τους *Fast and Exact Shortest Path Queries Using Highway Hierarchies*, πρότειναν μία νέα μέθοδο εύρεσης της βέλτιστης διαδρομής μεταξύ δύο κόμβων βασιζόμενοι στην ιεράρχηση των δικτύων και πιο συγκεκριμένα ακολουθώντας την παραδοχή ότι οι χρόνοι διάνυσης μεγάλων οδικών αξόνων είναι μειωμένοι σε σύγκριση με το τοπικό δίκτυο.

G.Righini, M.Salani (2005) στην εργασία τους *New dynamic programming algorithms for the resource-constrained elementary shortest path problem,* πρότειναν ένα νέο αλγόριθμο που βασίζεται στη μέθοδο των χρονοπαράθυρων η οποία παρουσιάστηκε από τους Desrochers et al. (1992).

E.Nikolova, M.Brand, D.R.Karger (2006) στην εργασία τους *Optimal root planning under uncertainty,* πρότειναν έναν αλγόριθμο για την εύρεση της διαδρομής ελαχίστου κόστους μεταξύ δύο κόμβων (PPSPP) σε δίκτυα όπου το βάρος των συνδέσμων είναι μία μεταβλητή η οποία εξαρτάται μη γραμμικά από το χρόνο. Ο αλγόριθμός τους χρησιμοποιεί το δυναμικό προγραμματισμό και έχει πολυωνυμικό χρόνο εκτέλεσης μόνο στην περίπτωση που επιλέγει και το χρόνο εκκίνησης της διαδρομής.

H.J.Cho, C.L.Lan (2009) στην εργασία τους *Hybrid shortest path algorithm for vehicle navigation*, πρότειναν ένα νέο αλγόριθμο εύρεσης της συντομότερης διαδρομής μεταξύ δύο κόμβων (PPSPP) χρησιμοποιώντας τους αλγορίθμους Dijkstra, A* και διπλής κατεύθυνσης (bidirectional) για χρήση στα συστήματα πλοήγησης. Υποθέτουν επίσης ότι οι χρήστες επιθυμούν να μετακινηθούν μέσω κεντρικών οδικών αρτηριών αποφεύγοντας το τοπικό δίκτυο.

3.3.ΕΡΕΥΝΕΣ ΓΙΑ ΤΟΝ ΚΑΤΑΜΕΡΙΣΜΟ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΣΤΑΤΙΚΑ ΔΙΚΤΥΑ

J.G.Wardrop (1952) στην εργασία του *Some theoretical aspects of road traffic research,* διατύπωσε τις δύο αρχές ισορροπίας των δικτύων (ως προς το χρήστη και ως προς το σύστημα) που αποτελούν τη βάση για την εξισορρόπηση των συγκοινωνιακών δικτύων μέχρι και σήμερα.

M.Beckmann, C.B.McGuire, C.B.Winsten (1956) στην εργασία τους *Studies in the Economics of Transportation,* ήταν οι πρώτοι που διατύπωσαν την ισορροπία ως προς το χρήστη που δόθηκε από την πρώτη αρχή του Wardrop σε μορφή μαθηματικού προβλήματος που μπορεί να προγραμματιστεί.

M.Frank, P.Wolfe (1956) στην εργασία τους *An algorithm for quadratic programming*, πρότειναν έναν αλγόριθμο για την επίλυση προβλημάτων τετραγωνικού προγραμματισμού υπό γραμμικούς περιορισμούς. Η μέθοδος αυτή είναι μία επέκταση του αλγορίθμου Simplex που χρησιμοποιείται στο γραμμικό προγραμματισμό και χρησιμοποιήθηκε αργότερα για τον καταμερισμό της ζήτησης στα δίκτυα.

D.Braess (1968) στην εργασία του *Uber ein paradoxon der verkehrsplanung*, διατύπωσε το πρώτο παράδοξο στην ισορροπία των δικτύων. Σύμφωνα με αυτή την εργασία ακόμα και η προσθήκη νέων συνδέσμων στο δίκτυο μπορεί να οδηγήσει σε επιδείνωση της υπάρχουσας κατάστασης. Πολλές μελέτες ακολούθησαν μετά από αυτό και διατυπώθηκε το ερώτημα του κατά πόσο και πότε χρησιμεύει η κατασκευή νέων οδικών αρτηριών σε δίκτυα με κυκλοφοριακή συμφόρηση.

S.C.Dafermos, F.T.Sparrow (1969) στην εργασία τους *The traffic assignment problem for a general network,* ασχολήθηκαν με την ισορροπία ως προς το χρήστη σε στατικά δίκτυα με ντετερμινιστικά βάρη συνδέσμων. Στην εργασία αυτή πρότειναν ένα νέο ορισμό για την ισορροπία ως προς το χρήστη που διαφέρει από την πρώτη αρχή του Wardrop. Η ισορροπία αυτή είναι γνωστή ως [User-optimized flow (UO)].

J.D.Murchland (1970) στην εργασία *Braess's paradox of traffic flow*, επικαιροποίησε το παράδοξο του Braess μέσω παραδειγμάτων και επεσήμανε και άλλες τέτοιες περιπτώσεις που προκύπτουν κατά τον καταμερισμό της ζήτησης.

R.B.Potts, R.M.Oliver (1972) στο βιβλίο τους *Flows in Transportation Networks,* ασχολήθηκαν με όλα τα στάδια σχεδιασμού μεταφορικών συστημάτων και με το πρόβλημα της ισορροπίας ως προς το χρήστη σε στατικά δίκτυα.

L.J.LeBlanc, E.K.Morlok, W.P.Pierskalla (1975) στην εργασία τους *An Efficient Approach to solving the Network Equilibrium Traffic Assignment Problem*, εφάρμοσαν πρώτη φορά τον αλγόριθμο των Frank, Wolfe (1956) για τον καταμερισμό της ζήτησης σε ένα μικρό στατικό δίκτυο.

C.F.Daganzo, Y.Sheffi (1977) στην εργασία τους *On Stochastic Model of Traffic Assignment,* διατύπωσαν μία από τις σημαντικότερες ισορροπίες στα συγκοινωνιακά δίκτυα. Τη στοχαστική ισορροπία ως προς το χρήστη, που σε πολλές εφαρμογές έχει αντικαταστήσει την πρώτη αρχή του Wardrop.

R.E.Allsop, J.A.Charlesworth (1977) στην εργασία τους *Traffic in a Signal-Controlled Road Network: An Example of Different Signal Timings Inducing Different Routeings*, πρότειναν μία μέθοδο υπολογισμού ώστε να βελτιστοποιείται ο έλεγχος της κυκλοφορίας και να διατηρείται η ισορροπία στο δίκτυο.

C.Fisk (1979) στην εργασία του *More paradoxes in the equilibrium assignment problem*, επεσήμανε διάφορες περιπτώσεις που οδηγούν σε παράδοξα αποτελέσματα κατά τον καταμερισμό της ζήτησης στο δίκτυο.

S.C.Dafermos (1980) στην εργασία του *Traffic Equilibrium and variational inequalities,* μελέτησε την ισορροπία ως προς το χρήστη σε στατικά δίκτυα και τις μεθόδους καταμερισμού της ζήτησης.

Y.Sheffi (1985) στο βιβλίο του *Urban Transportation Networks, Equilibrium Analysis with Mathematical Programming Methods,* κατηγοριοποίησε τις υπάρχουσες και πρότεινε νέες μεθόδους στο πρόβλημα της ισορροπίας των δικτύων ως προς το χρήστη σε στατικά δίκτυα. Αναφέρθηκε κυρίως στη μέθοδο των πολλαπλασιαστών Lagrange μέσω της οποίας επαναδιατυπώνεται το μαθηματικό μοντέλο των Beckmann, McGuire, Winsten (1956) και μπορεί να χρησιμοποιηθεί ο αλγόριθμος των Frank, Wolfe (1956).

B.G.Heydecker (1986) στην εργασία του *On the definition of traffic equilibrium*, πρότεινε ένα νέο ορισμό για την ισορροπία ως προς το χρήστη, γνωστό ως [Equilibrated flow (EQ)].

R.LeBlanc, D.E.Boyce (1986) στην εργασία τους *A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows*, πρότειναν μία νέα μέθοδο για τον καταμερισμό της ζήτησης σε στατικά δίκτυα.

M.Patriksson (1994) στην εργασία του *The Traffic Assignment Problem: Models and Methods,* ασχολήθηκε με το πρόβλημα της εξισορρόπησης των δικτύων και κατέγραψε τις μεθόδους που χρησιμοποιήθηκαν για την επίλυσή του.

Y.A.Korilis, A.A.Lazar, A.Orda (1997) και Y. A.Korilis, A.A.Lazar, A.Orda (1999) στις εργασίες τους *Capacity allocation under noncooperative routing* και *Avoiding the Braess paradox in noncooperative network*, διερεύνησαν τις απαιτούμενες συνθήκες ώστε η προσθήκη ενός συνδέσμου να βελτιώσει την υπάρχουσα κατάσταση στο δίκτυο αποφεύγοντας το παράδοξο του Braess (1968).

3.4.ΕΡΕΥΝΕΣ ΓΙΑ ΤΟΝ ΚΑΤΑΜΕΡΙΣΜΟ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ

C.F.Daganzo (1978) στην εργασία του *On the traffic assignment problem with flow dependent costs-II,* διατύπωσε έναν αλγόριθμο που φορτίζει το δίκτυο ακολουθώντας τις αρχές της εξισορρόπησης ως προς το χρήστη με κόστη συνδέσμων που εξαρτώνται από την κυκλοφοριακή συμφόρηση.

C.Fisk (1980) στην εργασία του *Some Developments in equilibrium traffic assignment*, πρότεινε μία μαθηματική έκφραση βασιζόμενη στο λογιστικό μοντέλο για τη στοχαστική ισορροπία του χρήστη.

D.W.Hearn, S.Lawphongpanich, J.A.Ventura (1987) στην εργασία τους *Restricted simplicial decomposition: computation and extensions*, πρότειναν μία μέθοδο για τον σταδιακό καταμερισμό της ζήτησης σε δυναμικά δίκτυα.

D.E.Boyce (1989) στην εργασία του *Route Guidance Systems for managing urban transportation networks: Review and prospects*, ανέλυσε τις αδυναμίες των συστημάτων πλοήγησης που ακολουθούν το στατικό καταμερισμό της ζήτησης όταν εφαρμόζονται σε δυναμικά δίκτυα και εστίασε στα προβλήματα κυκλοφοριακής συμφόρησης που προκαλούνται.

B.N.Janson (1991) στην εργασία του *Dynamic traffic assignment for urban road networks,* πρότεινε ένα μοντέλο που περιελάμβανε τις επιλογές διαδρομών ως μεταβλητές που χρησιμοποιούνται για την πρόβλεψη του χρόνου διάνυσης των συνδέσμων.

D.E.Kaufman, R.L.Smith, K.E.Wunderlich (1991) στην εργασία τους *An Iterative Routing/Assignment Method for Anticipatory Real-Time Route Guidance*, πρότειναν ένα λογισμικό πακέτο μέσω του οποίου επεξεργάζονται τα δεδομένα σχετικά με τις κυκλοφοριακές συνθήκες που επικρατούν και δίνονται τιμές στους χρόνους διάνυσης των συνδέσμων σχετικά με το άμεσο μέλλον. Οι τιμές αυτές δίνονται μέσω προβλέψεων χρησιμοποιώντας και ιστορικά στοιχεία.

T.L.Friesz, D.Bernstein, T.E.Smith, R.L.Tobin, B.W.Wie (1993) στην εργασία τους *A Variational inequality formulation of the dynamic network user equilibrium problem,* ανέπτυξαν ένα μοντέλο για την επιλογή διαδρομής και τη λήψη απόφασης σχετικά με το χρόνο αναχώρησης του μεταφορικού μέσου σε δυναμικά δίκτυα. Ανέλυσαν επίσης τα κριτήρια που πρέπει να πληροί η δυναμική πλέον ισορροπία ως προς το χρήστη.

B.Ran, D.E.Boyce, L.J.LeBlanc (1993) στην εργασία τους *A New class of instantaneous dynamic useroptimal traffic assignment models,* πρότειναν μία μέθοδο καταμερισμού της ζήτησης συ δυναμικά δίκτυα υποθέτοντας ότι οι χρήστες μπορεί να ακολουθήσουν μία διαδρομή που δεν είναι βέλτιστη για το κοινό καλό με την προϋπόθεση ότι δε θα διαφέρει σημαντικά από τη βέλτιστη.

D.E.Kaufman, R.L.Smith (1993) στην εργασία τους *Fastest paths in time dependent networks for intelligent vehicle-highway systems application,* πρότειναν έναν

αλγόριθμο για τον υπολογισμό της συντομότερης διαδρομής σε δυναμικά δίκτυα. Ο αλγόριθμος αυτός στηρίζεται στη συνεχή ροή κυκλοφοριακών δεδομένων και χρησιμοποιεί ως υπορουτίνα έναν αλγόριθμο επίλυσης του (SSSPP). Ο χρόνος εκτέλεσής του δε διαφέρει από το χρόνο που απαιτείται για την εύρεση της συντομότερης διαδρομής σε στατικά δίκτυα και στο σύνολό του εξαρτάται από το πόσο συχνά αναβαθμίζονται τα δεδομένα.

B.Ran, D.E.Boyce (1994) στην εργασία τους *Dynamic Urban Transportation Network Models-Theory and Implication for Intelligent Vehicle-Highway Systems*, εφάρμοσαν για πρώτη φορά τον αλγόριθμο των Frank, Wolfe (1956) σε δυναμικά δίκτυα για τον καταμερισμό της ζήτησης.

M.Ghali, M.Smith (1995) στην εργασία τους *A model for the dynamic system optimum traffic assignment problem*, περιέγραψαν ένα μοντέλο καταμερισμού της ζήτησης με στόχο τη μείωση των καθυστερήσεων σε συγκοινωνιακά δίκτυα επικεντρώνοντας στις καθυστερήσεις που προκύπτουν σε κάθε επιμέρους σύνδεσμο.

B.W.Wie, R.Tobin, D.Bernstein, T.Friesz (1995) στην εργασία τους *A Comparison of system optimal and user optimal equilibrium dynamic traffic assignments with scheduled delays,* συνέκριναν την ισορροπία ως προς το χρήστη και την ισοροοπία ως προς το σύστημα σε ένα δυναμικό δίκτυο με δεκαοκτώ συνδέσμους συγκρίνοντας τους συνολικούς χρόνους διάνυσης σε κάθε περίπτωση.

O.Damberg, J.T.Lundgren, M.Patriksson (1996) στην εργασία τους *An algorithm for the stochastic user equilibrium problem*, πρότειναν έναν αλγόριθμο για τον καταμερισμό της ζήτησης σε δυναμικά δίκτυα με στόχο τη στοχαστική ισορροπία του χρήστη.

K.Wunderlich, D.Kaufman, R.L.Smith (1997) στην εργασία τους *Link travel time prediction techniques for convergent iterative anticipatory route guidance methods*, πρότειναν μία μέθοδο που βασίζεται στη συλλογή κυκλοφοριακών δεδομένων και την τροποποίησή τους λαμβάνοντας υπόψη τις βραχυπρόθεσμες μεταβολές στις κυκλοφοριακές συνθήκες. Έτσι τα κυκλοφοριακά δεδομένα τροποποιούνται και διανέμονται στους χρήστες οδηγώντας τους στην επιλογή διαδρομής μέσω μίας δρομολόγησης που περιέχει στοιχεία πρόβλεψης για το άμεσο μέλλον.

M.E.Ben-Akiva, M.Bierlaine, J.Bottom, H.N.Koutsopoulos, R.Mishalani, R. (1997) στην εργασία τους *Development of a Route Guidance Generation System for Real-Time Application*, πρότειναν ένα λογισμικό πακέτο, γνωστό ως DynaMIT το οποίο επεξεργάζεται τα τρέχοντα κυκλοφοριακά δεδομένα μαζί με ιστορικά στοιχεία και προτείνει χρόνους διάνυσης για τους συνδέσμους βραχυπρόθεσμα.

E.Koutsoupias, C.Papadimitriou (1999) στην εργασία *Worst-case equilibria,* μελέτησαν τα αρνητικά αποτελέσματα που έχει η δρομολόγηση με στόχο της ισορροπία του χρήστη σε συγκοινωνιακά δίκτυα.

M.Mavronicolas, P.Spirakis (2001) στην εργασία τους *The price of selfish routing*, διερεύνησαν το κόστος της δρομολόγησης με στόχο την επιλογή της συντομότερης διαδρομής από κάθε χρήστη σε δυναμικά δίκτυα.

ΚΕΦΑΛΑΙΟ 4ο

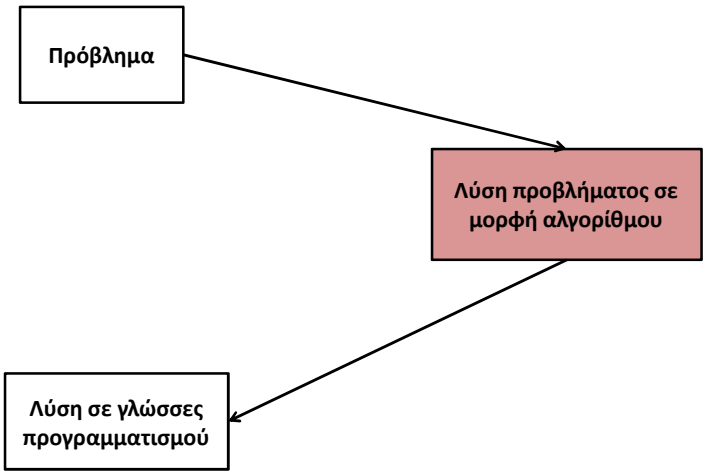
ΣΤΟΙΧΕΙΑ ΑΛΓΟΡΙΘΜΩΝ

4.1.ΑΛΓΟΡΙΘΜΟΙ

Ο όρος αλγόριθμος (algorithm), προέρχεται από τη λέξη (algorism) και αρχικά (aljabr). Η ιδέα του αλγορίθμου αναπτύχθηκε πρώτη φορά από τον Άραβα μαθηματικό Abu Ja'far Mohammed ibn Mussa al-Khowarizmi, στις αρχές του 9ου αιώνα. Το βιβλίο του On Calculation with Hindu–Arabic numeral system μεταφράστηκε αρκετά αργότερα στα λατινικά με τίτλο Algoritmi de numero Indorum που σήμαινε Algoritmi on the numbers of the Indians όπου Algoritmi ήταν το όνομα του συγγραφέα σε λατινική μετάφραση. Η λέξη Algoritmi είναι και η αιτία της σύγχρονης ονομασίας των αλγορίθμων, καθώς οι αναγνώστες του βιβλίου θεώρησαν ότι η λέξη algoritmi αναφερόταν στη λατινική λέξη algorismus που σήμαινε υπολογιστική μέθοδος.

Ένας αλγόριθμος μπορεί να οριστεί ως μία διατεταγμένη ακολουθία βημάτων που οδηγεί στην επίλυση ενός προβλήματος μέσω εντολών, έτσι κάθε αλγόριθμος στην ουσία αποτελεί μία ακολουθία καλά ορισμένων εντολών. Ένα από τα πρώτα παράδειγμα ανάπτυξης μίας ακολουθίας σαφώς ορισμένων εντολών που οδηγεί στην επίλυση ενός προβλήματος είναι ο αλγόριθμος εύρεσης του μέγιστου κοινού διαιρέτη δύο ακεραίων που διατυπώθηκε από τον Ευκλείδη. Τα χαρακτηριστικά των καλών αλγορίθμων είναι η γενικότητα που αναφέρεται στην ποικιλία των δεδομένων που μπορεί να επεξεργαστεί ο αλγόριθμος και η οριστικότητα που αναφέρεται στη σαφήνεια του αλγορίθμου όταν επιλύει κάποιο πρόβλημα. Συνήθως οι αλγόριθμοι υλοποιούνται μέσω υπολογιστών όπου μία βάση δεδομένων δέχεται επεξεργασία από μία σειρά εντολών που αποτελούν το σώμα του αλγορίθμου και παράγεται ένα αρχείο αποτελεσμάτων. Βέβαια ένας αλγόριθμος μπορεί να υλοποιηθεί και από άλλα μέσα όπως ο ανθρώπινος εγκέφαλος, κάποια μηχανική συσκευή κ.α.

Μία από τις μεγαλύτερες δυσκολίες κατά την υλοποίηση ενός αλγορίθμου σε Η/Υ είναι η δυσχέρεια στην επικοινωνία μεταξύ χρήστη και Η/Υ. Ο χρήστης προσπαθεί να ορίσει μία ακολουθία εντολών που θα χρησιμοποιηθούν από τον Η/Υ για την εκτέλεση του αλγορίθμου. Το σημαντικότερο στοιχείο που πρέπει να έχουν οι εντολές αυτές ώστε να υπάρχει σωστή επικοινωνία μεταξύ χρήστη και Η/Υ είναι η σαφήνειά, δηλαδή οι εντολές να μην είναι σε καμία περίπτωση διφορούμενες και να μη μπορούν να ερμηνευτούν με περισσότερους από έναν τρόπο. Για να επιτευχθεί αυτό είναι σημαντικό να διαχωριστούν οι φάσεις της δημιουργίας του αλγορίθμου και της υλοποίησης του σε Η/Υ.



Σχήμα 4.1.Φάσεις επίλυσης προβλήματος μέσω Η/Υ

Τέλος, οι αλγόριθμοι είναι χρήσιμοι όταν πληρούν κάποιο σκοπό. Αυτό σημαίνει ότι μπορούν να υλοποιηθούν από κάποιο μέσο είτε αυτό είναι ο ανθρώπινος εγκέφαλος ή ένας Η/Υ ή κάποιο άλλο και να παράγουν αποτέλεσμα σε πραγματικό χρόνο. Περισσότερα στοιχεία σχετικά με το χώρο των αλγορίθμων υπάρχουν στο βιβλίο : Κάβουρας (2003).

4.2.ΣΥΝΤΟΜΗ ΑΝΑΦΟΡΑ ΣΤΙΣ ΚΑΤΗΓΟΡΙΕΣ ΜΕΘΟΔΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

4.2.1.Γραμμικός Προγραμματισμός (Linear Programming)

Ιστορικά, ο γραμμικός προγραμματισμός αναπτύχθηκε αρχικά από το Ρώσο μαθηματικό Leonid Kantorovich (1939) ο οποίος ανέπτυξε μία σειρά από γραμμικά προβλήματα και τον G.B.Dantzig (1949) που δημοσίευσε τον αλγόριθμο simplex. Ο γραμμικός προγραμματισμός είναι μία τεχνική που στοχεύει στη μεγιστοποίηση ή ελαχιστοποίηση μίας συνάρτησης, η οποία καλείται αντικειμενική και αποτελείται από μεταβλητές που παίρνουν τιμές στο σύνολο των πραγματικών αριθμών πληρώντας συγκεκριμένους περιορισμούς. Η λύση που μεγιστοποιεί ή ελαχιστοποιεί την αντικειμενική συνάρτηση ικανοποιώντας και τους επιμέρους περιορισμούς καλείται βέλτιστη (optimal solution), ενώ οι λύσεις που ικανοποιούν μόνο τους περιορισμούς καλούνται δυνατές. Θεωρώντας αντικειμενική συνάρτηση : $c_1x_1 + c_2x_2 + \ldots\ldots + c_mx_m$

Η γενική μορφή ενός προβλήματος γραμμικού προγραμματισμού είναι :

Μεγιστοποίησε ή Ελαχιστοποίησε την ποσότητα :

$\sum_{i=1}^{m} c_i x_i \quad , x_i \in R$

Ώστε να ισχύουν οι περιορισμοί :

$\{ \sum_{i=1}^{m} a_{j,i} x_i \geq$ ή $\leq b_j \}$ για $\forall$ jϵ1,k

Όπου k ο αριθμός των περιορισμών και $b_j, a_{j,i} \in R$

Η παραπάνω έκφραση αυτή μπορεί να δοθεί και σε μορφή πινάκων ως :

Μεγιστοποίησε ή Ελαχιστοποίησε : $\mathbf{c}^T\mathbf{x}$ **:** Αντικειμενική Συνάρτηση
Υπό τους περιορισμούς : $A\mathbf{x} \leq$ ή $\geq \mathbf{b}$.

4.2.2.Ακέραιος Γραμμικός Προγραμματισμός (Integer Linear Programming)

Στον γραμμικό ακέραιο προγραμματισμό οι μεταβλητές που πρέπει να υπολογιστούν παίρνουν τιμές στο σύνολο των ακεραίων. Αυτό δημιουργεί προβλήματα κατά την επίλυση καθώς δε μπορεί να βρεθεί το πραγματικό βέλτιστο της αντικειμενικής συνάρτησης όπως στον γραμμικό προγραμματισμό. Σε αντίθεση με το γραμμικό προγραμματισμό όπου μπορεί να χρησιμοποιηθεί ο αλγόριθμος simplex, δεν έχει βρεθεί αντίστοιχος αποδοτικός αλγόριθμος για την επίλυση προβλημάτων αυτής της κατηγορίας. Η γενική μορφή των προβλημάτων ακέραιου προγραμματισμού είναι :

Μεγιστοποίησε ή Ελαχιστοποίησε την ποσότητα :

$\sum_{i=1}^{m} c_i x_i \quad , x_i \in Z$

Ώστε να ισχύουν οι περιορισμοί :

$\{ \sum_{i=1}^{m} a_{j,i} x_i \geq ή \leq b_j \}$ για $\forall$ jϵ1,k

Όπου k ο αριθμός των περιορισμών και b_j, $a_{j,i} \epsilon R$

Ένα παράδειγμα μπορεί να είναι :

Μεγιστοποίησε : c_1X1 + c_2X2 +…….+c_iXi Xi ϵ Z.

Περιορισμοί : a_{11} X1 + a_{12} X2+…….+a_{1i} Xi ≤ b1 $a_{1i,}$ b1 ϵ R.

a_{21} X1 + a_{22} X2+…….+a_{2i} Xi ≤ b2 $a_{2i,}$ b2 ϵ R.

…….

…….

a_{j1} X1 + a_{j2} X2+…….+a_{ji} Xi ≤ bj $a_{ji,}$ bj ϵ R.

Μία ακόμα περίπτωση γραμμικού προγραμματισμού αποτελεί και ο Μικτός Ακέραιος Γραμμικός Προγραμματισμός (Mixed Integer Linear Programming), στον οποίο κάποιες μεταβλητές επιτρέπεται να πάρουν τιμές στο σύνολο των πραγματικών και κάποιες άλλες στο σύνολο των ακεραίων. Περισσότερες πληροφορίες σχετικά με το γραμμικό και τον ακέραιο γραμμικό προγραμματισμό υπάρχουν στο βιβλίο : Καρλαύτης, Λαγαρός (2010).

4.2.3.Μη Γραμμικός Προγραμματισμός (Non Linear Programming)

Στο μη γραμμικό προγραμματισμό η αντικειμενική συνάρτηση που πρέπει να μεγιστοποιηθεί ή να ελαχιστοποιηθεί αντίστοιχα δεν είναι γραμμική. Μπορεί να είναι για παράδειγμα δευτεροβάθμια. Επίσης οι μεταβλητές που βρίσκονται στην αντικειμενική συνάρτηση ικανοποιούν κάποιους περιορισμούς που εκφράζονται μέσω ανισοτήτων και αυτές οι ανισώσεις μπορεί επίσης να μην είναι γραμμικές. Ένα τέτοιο παράδειγμα μη γραμμικού προβλήματος είναι το ακόλουθο.

Μεγιστοποίησε : $x_1x_2 + x_2x_3$

Υπό τους περιορισμούς : $x_1^2 - x_2^2 + x_3^2 \leq 2$

$x_1^2 + x_2^2 + x_3^2 \leq 10$

4.2.4.Δυναμικός Προγραμματισμός (Dynamic Programming)

Η πρώτη αναφορά στο δυναμικό προγραμματισμό έγινε το 1940 από τον μαθηματικό και καθηγητή στο πανεπιστήμιο της ανατολικής Καλιφόρνια, Richard Bellman για να περιγράψει τη διαδικασία επίλυσης ενός προβλήματος στο οποίο απαιτείται να λαμβάνονται συνεχώς οι σωστές αποφάσεις η μία μετά την άλλη. Ο Δυναμικός Προγραμματισμός (ΔΠ) είναι μία υπολογιστική μέθοδος η οποία εφαρμόζεται όταν πρόκειται να ληφθεί μία σύνθετη απόφαση η οποία προκύπτει από τη σύνθεση επιμέρους αποφάσεων που αλληλοεξαρτώνται. Συνήθως η αλληλεξάρτηση μεταξύ των αποφάσεων προκύπτει επειδή μεσολαβεί κάποια χρονική διαδοχή, όπως συμβαίνει και στην περίπτωση αναζήτησης της συντομότερης διαδρομής μεταξύ δύο

σημείων σε ένα γράφημα. Οι αποφάσεις αυτές μπορεί να λαμβάνονται σε ένα περιβάλλον γνωστών συνθηκών (ντετερμινιστικός δυναμικός προγραμματισμός) ή ακόμα και σε ένα περιβάλλον αβεβαιότητας (στοχαστικός δυναμικός προγραμματισμός). Η μέθοδος επίλυσης προβλημάτων αυτής της κατηγορίας βασίζεται στη διασύνδεση των επιμέρους αποφάσεων με κατάλληλη αναδρομική σχέση ώστε η σύνθεση των επιμέρους αποφάσεων να δίνει την τελικά βέλτιστη επιλογή. Το αρχικό πρόβλημα διασπάται σε επιμέρους υποπροβλήματα τα οποία συνδέονται με τη βοήθεια κατάλληλων αναδρομικών σχέσεων. Για να καλυφθούν όλες οι εκδοχές από τη διασύνδεση των επιμέρους υποπροβλημάτων, τα υποπροβλήματα αυτά λύνονται παραμετρικά, δηλαδή για όλες τις δυνατές τιμές ορισμένων παραμέτρων. Αυτό αποτελεί και το κυρίως υπολογιστικό κόστος της μεθόδου, το οποίο, αν και είναι σημαντικό, είναι πάντως πολύ μικρότερο από το κόστος της πλήρους απαρίθμησης και αξιολόγησης όλων των δυνατών λύσεων. Λόγω αυτού του κόστους η μέθοδος χρησιμοποιείται για προβλήματα που δεν είναι δυνατό να αντιμετωπισθούν με μεθόδους Γραμμικού ή Γραμμικού Ακέραιου Προγραμματισμού. Χαρακτηριστικό του ΔΠ είναι ότι δεν υπάρχει γενικευμένη διατύπωση της μεθόδου που να έχει άμεση λειτουργική ισχύ. Οι αναδρομικές σχέσεις που συνεπάγεται η μέθοδος διαφοροποιούνται ριζικά από πρόβλημα σε πρόβλημα. Η αρχή πάνω στην οποία στηρίχθηκε ο δυναμικός προγραμματισμός, γνωστή και ως (Principle of Optimality) όπως διατυπώθηκε από τον Bellman, υποστηρίζει ότι η βέλτιστη ακολουθία αποφάσεων που απαιτείται για την επίλυση του προβλήματος προέρχεται από τη βέλτιστες ακολουθίες αποφάσεων κατά την επίλυση τοπικών υποπροβλημάτων. Η αρχή αυτή περιγράφεται ακολούθως.

Έστω ένα συγκοινωνιακό δίκτυο G={N,E}, όπου N ο αριθμός των κόμβων του και E ο αριθμός των συνδέσμων του. Κάθε σύνδεσμος του δικτύου e ϵ E έχει ένα βάρος που αντιπροσωπεύει τη δυσκολία μετακίνησης μέσω αυτού w(e). Τότε για τη μετακίνηση μεταξύ δύο κόμβων του δικτύου r, s μέσω μίας διαδρομής p υπάρχει συνολική αντίσταση μετακίνησης D(s) = $\sum_{e \epsilon p} w(e)$. Αν στόχος του προβλήματος είναι η εύρεση της διαδρομής με τη μικρότερη αντίσταση μεταξύ των κόμβων r, s η αντικειμενική συνάρτηση που πρέπει να ελαχιστοποιηθεί είναι η D(s). Για την επίλυση του προβλήματος απαιτείται η λήψη μίας βέλτιστης ακολουθίας αποφάσεων. Εν προκειμένω οι αποφάσεις αυτές αφορούν τους διαδοχικούς συνδέσμους που θα ακολουθήσει η διαδρομή από το r στο s. Εάν δε χρησιμοποιηθεί ο δυναμικός προγραμματισμός απαιτείται ο προσδιορισμός όλων των δυνατών διαδρομών από το r στο s και η επιλογή της διαδρομής για την οποία η συνάρτηση D(s) παίρνει τη μικρότερη τιμή. Με χρήση του δυναμικού προγραμματισμού όμως η επίλυση του προβλήματος έχει πολύ μικρότερο υπολογιστικό κόστος χρησιμοποιώντας της αρχή ότι οι βέλτιστες ακολουθίες αποφάσεων κατά την επίλυση τοπικών υποπροβλημάτων οδηγούν και στο ολικό βέλτιστο. Πιο συγκεκριμένα εάν η διαδρομή με τη μικρότερη αντίσταση από το r στο s αποτελείται από τους κόμβους p = {r, x_1, …, x_k, …, x_n, s} τότε και η διαδρομή με τη μικρότερη αντίσταση από το r στον τυχαίο κόμβο x_k, ο οποίος ανήκει στη διαδρομή p, θα είναι q_1 = {r, x_1, …, x_k}. Αυτό είναι εμφανές καθώς εάν η διαδρομή αυτή ήταν διαφορετική π.χ. q_2 = {r, x_1', …, x_k} τότε η

αντίσταση της διαδρομής q_2 είναι μικρότερη από την αντίσταση της διαδρομής q_1. Η διαδρομή p όμως προκύπτει ως άθροισμα των διαδρομών {r, x_1, …, x_k} + {x_{k+1}, …, x_n, s}. Έτσι εάν η διαδρομή q_2 = {r, x_1', …, x_k} έχει μικρότερη αντίσταση από την q_1 και η διαδρομή {r, x_1', …, x_k} + {x_{k+1}, …, x_n, s} θα έχει μικρότερη αντίσταση από τη διαδρομή p. Κάτι τέτοιο όμως δε μπορεί να ισχύει γιατί η p είναι η διαδρομή με τη μικρότερη αντίσταση από το r στο s. Οπότε ισχύει η υπόθεση ότι για κάθε κόμβο που ανήκει στη διαδρομή p η συντομότερη διαδρομή από τον κόμβο r στον κόμβο αυτόν αποτελεί τμήμα της διαδρομής p. Το πρόβλημα δηλαδή μπορεί να επιλυθεί αναδρομικά με συνεχείς επαναλήψεις μέσω της διάσπασής του σε μικρότερα υποπροβλήματα. Έτσι η ελαχιστοποίηση της συνάρτησης D(s) προϋποθέτει την επίλυση πιο εύκολων προβλημάτων όπως την ελαχιστοποίηση των συναρτήσεων D(i), iϵN, όπου η ελαχιστοποίηση μίας συνάρτησης οδηγεί στην ελαχιστοποίηση μίας άλλης συνάρτησης και αυτό συνεχίζεται μέχρι να φθάσουμε στην D(s).

Τα προβλήματα που επιλύονται μέσω του Δυναμικού Προγραμματισμού παρουσιάζουν τα ακόλουθα χαρακτηριστικά :
1) Οι αποφάσεις λαμβάνονται διαδοχικά.
2) Το πρόβλημα μπορεί να διαιρεθεί σε βήματα (φάσεις) και σε κάθε βήμα απαιτείται να ληφθεί μια "στρατηγική" απόφαση.
3) Κάθε βήμα έχει ένα ορισμένο αριθμό "καταστάσεων" που συνδέονται με αυτό.
4) Το αποτέλεσμα μιας στρατηγικής απόφασης που λαμβάνεται σε κάθε βήμα είναι να μετατρέπει την παρούσα κατάσταση σε μια κατάσταση που συνδέεται με το επόμενο βήμα.
5) Με κάθε απόφαση συνδέεται ένα κέρδος ή μία ζημία (κόστος).
6) Ο αντικειμενικός σκοπός, που εκφράζεται από την αντικειμενική συνάρτηση, είναι να μεγιστοποιηθεί το συνολικό κέρδος ή να ελαχιστοποιηθεί η συνολική ζημία, ή γενικότερα να επιτευχθεί το καλύτερο δυνατό αποτέλεσμα.
7) Τέλος, ο τρόπος με τον οποίο βρεθήκαμε σε μια κατάσταση ενός βήματος είναι άσχετος με τις αποφάσεις που θα επακολουθήσουν. Δηλαδή οι αποφάσεις που θα επακολουθήσουν εξαρτώνται μόνο από την κατάσταση στην οποία βρισκόμαστε και όχι από τον τρόπο με τον οποίο βρεθήκαμε σε αυτή την κατάσταση. Περισσότερες πληροφορίες σχετικά με το δυναμικό προγραμματισμό υπάρχουν στο βιβλίο του Bellman (2003).

4.2.5.Άπληστοι Αλγόριθμοι (Greedy Algorithms)

Χρησιμοποιούνται σε προβλήματα που ο Δυναμικός Προγραμματισμός είναι απαραίτητος αλλά έχει μεγάλο κόστος κατά την εφαρμογή του. Με τον όρο άπληστοι χαρακτηρίζονται οι αλγόριθμοι που σε κάθε στάδιο επιλέγουν την τοπική βέλτιστη λύση και οδηγούν στην εύρεση της καθολικής βέλτιστης λύσης. Οι αλγόριθμοι αυτοί αφού βρουν την τοπική βέλτιστη λύση προχωρούν στην εύρεση της επόμενης βέλτιστης λύσης χωρίς να εξετάσουν ξανά τη λύση που έχει ήδη βρεθεί. Έτσι σε κάθε επανάληψη το μέγεθος του προβλήματος μειώνεται. Αυτή είναι και η βασική τους διαφορά με το δυναμικό προγραμματισμό στον οποίο σε κάθε επόμενο βήμα λαμβάνονται αποφάσεις με βάση τα νέα δεδομένα που προκύπτουν και μπορούν να

γίνουν αλλαγές ακόμα και σε επιλογές που είχαν γίνει σε προηγούμενα βήματα, κάτι που δε συμβαίνει πλέον καθώς οι επιλογές είναι αμετάκλητες. Οι αλγόριθμοι αυτοί μπορεί να χρησιμοποιηθούν σε προβλήματα στα οποία ισχύει η αρχή ότι η επόμενη βέλτιστη επιλογή οδηγεί στο τελικό βέλτιστο αποτέλεσμα.

4.2.6.Η μέθοδος 'Διαίρει και Βασίλευε' (Divide and Conquer)

Η μέθοδος προγραμματισμού με το όνομα διαίρει και βασίλευε αποτελείται από τρία βασικά βήματα όπως φανερώνουν και οι λέξεις της ονομασίας της :
–Διαίρει: Κατάτμηση του αρχικού προβλήματος σε n υποπροβλήματα, όσο το δυνατόν ιδίου μεγέθους.
–Βασίλευε: Επίλυση υποπροβλημάτων με αναδρομικό τρόπο.
–Συνδύασε: Συνδυασμός των επιμέρους λύσεων, εντός πολυωνυμικού χρόνου, για την επίτευξη της ολικής λύσης.
Πιο αναλυτικά το αρχικό προς επίλυση πρόβλημα τεμαχίζεται σε άλλα μικρότερα μέχρι να προκύψουν πολύ μικρά προβλήματα η επίλυση των οποίων είναι εύκολη. Μετά οι λύσεις αυτών των προβλημάτων συνδυάζονται για να προκύψει η λύση του βασικού προβλήματος.
Η ομοιότητα αυτής της μεθόδου με το Δυναμικό προγραμματισμό είναι η διάσπαση του αρχικού προβλήματος σε μικρότερου μεγέθους υποπροβλήματα τα οποία επιλύονται αναδρομικά.

4.3.ΣΤΟΙΧΕΙΑ ΑΝΑΛΥΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

Η ανάλυση των αλγορίθμων αναφέρεται στον προσδιορισμό των βασικών τους χαρακτηριστικών όπως ο χρόνος που απαιτείται για την εκτέλεσή τους, ο χώρος σε θέσεις μνήμης και η πιθανότητα να μην προκύψει αποδεκτό αποτέλεσμα κατά την εκτέλεση. Πιο συγκεκριμένα, κάθε αλγόριθμος μπορεί να επεξεργαστεί κάποια δεδομένα και να παράγει αποτελέσματα σε πραγματικό χρόνο. Τυπικά ο χρόνος που απαιτείται για την ολοκλήρωση αυτής της διαδικασίας αυξάνεται όταν υπάρχει αύξηση και του όγκου των δεδομένων. Ο μέσος χρόνος υλοποίησης ενός αλγορίθμου (average running time) είναι δύσκολο να υπολογιστεί θεωρητικά, γι'αυτό συνήθως λαμβάνεται υπόψη το χειρότερο σενάριο που μπορεί να προκύψει κατά την εκτέλεσή του (worst case running time). Κατά το θεωρητικό υπολογισμό του χρόνου υλοποίησης ενός αλγορίθμου γίνεται αρχικά η υπόθεση ότι ο χρόνος αυτός είναι μία συνάρτηση T() του αριθμού των δεδομένων n. Στη συνέχεια υπολογίζεται ο αριθμός των βασικών πράξεων που εκτελείται στη χειρότερη περίπτωση πάνω στα δεδομένα n από τον αλγόριθμο. Ο αριθμός αυτός μπορεί να προσεγγίσει και το χρόνο ο οποίος απαιτείται για την υλοποίησή του. Το πλεονέκτημα αυτής της μεθόδου είναι ότι μπορεί να προσεγγίσει το χρόνο εκτέλεσής του συναρτήσει μόνο του όγκου των δεδομένων χωρίς να λαμβάνονται υπόψη οι δυνατότητες του Η/Υ ή οποιουδήποτε άλλου μέσου υλοποίησης του αλγορίθμου. Μέσω του θεωρητικού υπολογισμού του χρόνου υλοποίησης μπορεί να προσδιοριστεί ο ρυθμός αύξησης του χρόνου αυτού (growth rate) κατά την αύξηση του όγκου των δεδομένων. Αυτός ο ρυθμός αύξησης είναι χαρακτηριστικό του αλγορίθμου.

Στη συνέχεια εξετάζεται ο χρόνος εκτέλεσης ενός συγκεκριμένου αλγορίθμου που δίνεται σε μορφή ψευδοκώδικα.

Δεδομένα : Μήτρα X με n ακεραίους
Αποτελέσματα : Μήτρα A με n ακεραίους #Βασικές Πράξεις

A←νέα μήτρα με ***n*** ακεραίους	***n***
for ***i***←0 **to** ***n*** − 1	***n***
s←***X***[0]	***1***
for ***j***←1 **to** ***i***	***1+2+…+(n-1)***
s← ***s*** + ***X***[***j***]	***1+2+…+(n-1)***
A[***i***] ←***s*** / (***i*** + 1)	***n***
return ***A***	***1***

Το σύνολο των πράξεων που απαιτούνται για την υλοποίηση του αλγορίθμου είναι:
$2 \times [1+2+….+(n-1)] + 3 \times n + 2 = 2 \times [1+2+….+(n-1)+n] + n + 2 =$
$2 \times n(n+1)/2 + n + 2.$

Όσο αυξάνεται ο όγκος των δεδομένων n, η συνάρτηση υπολογισμού του χρόνου υλοποίησης του αλγορίθμου $T(n) = 2 \times n(n+1) / 2 + n + 2$ δίνει μεγαλύτερα αποτελέσματα.

Σύμφωνα με στοιχεία από το πεδίο της ασυμπτωτικής θεωρίας στα μαθηματικά, μπορεί να περιγραφεί η συμπεριφορά μίας συνάρτησης όταν οι μεταβλητές τις τείνουν στο άπειρο. Αν και η θεωρία αυτή αρχικά είχε καθαρά μαθηματικό περιεχόμενο η κύρια εφαρμογή της σήμερα είναι στην πρόβλεψη της συμπεριφοράς των αλγορίθμων σχετικά με το χρόνο εκτέλεσής τους και τις απαιτήσεις τους σε θέσεις μνήμης. Με βάση τη θεωρία αυτή μπορεί να βρεθεί ένα ανώτατο όριο του ρυθμού αύξησης της συνάρτησης καθώς οι μεταβλητές της τείνουν στο άπειρο. Παράγωγα αυτής της θεωρίας είναι διάφορα εργαλεία εκτίμησης της πολυπλοκότητας των αλγορίθμων όπως :

Big-O Notation :
Δεδομένων των συναρτήσεων *T*(*n*) και g(n) μπορεί να γραφεί ότι
T(n) = O(g(n)) για n→ ∞ , αν και μόνο αν υπάρχει ένας αριθμός c ⩾ 0 ϵ R
και ένας αριθμός και ένας αριθμός $n_0 \geqslant 0 \in N$ τέτοιοι ώστε :
T(n) ≤ c g(n) για κάθε n > n_0.
Τότε η T(n) είναι ασυμπτωτικά μικρότερη ή ίση με τη g(n).

Στην εφαρμογή του παραδείγματος εάν υποθέσουμε ότι g(n) = n θα πρέπει :
| 2 × n(n+1)/2 + n + 2 | ≤ c | n | για κάθε n > n_0 →
2 × n(n+1)/2 + n + 2 ≤ c × n για κάθε n > n_0 > 0 →
n^2 + (2-c) × n + 2 ≤ 0 για κάθε n > n_0 > 0 →
n ≤ c - 2 - 2/n για κάθε n > n_0 > 0, κάτι το οποίο δε μπορεί να ισχύει αφού ο αριθμός c πρέπει να έχει συγκεκριμένη τιμή. Αντίθετα αν g(n) = n^2 αυτό ισχύει και μπορεί να γραφεί ότι T(n) = O(n^2), κάτι που σημαίνει ότι ο χρόνος υλοποίησης του προβλήματος ανταποκρίνεται με τετραγωνικό ρυθμό αύξησης κατά την αύξηση του όγκου των δεδομένων.

Η σημαντικότερη ιδιότητα αυτής της θεώρησης είναι ότι εάν μία συνάρτηση T(n) είναι πολυωνυμικού βαθμού d μπορεί να γραφεί ότι T(n) = O(n^d) χωρίς να λαμβάνονται υπόψη οι συντελεστές της συνάρτησης αυτής ή οι όροι της οι οποίοι έχουν βαθμό μικρότερο του d.
Έτσι εάν T(n) = $\sum_{i=1}^{d} a_i n^i$, τότε T(n) = O(n^d).

Big-Omega Notation :
Δεδομένων των συναρτήσεων *T*(*n*) και g(n) μπορεί να γραφεί ότι
T(n) = Ω(g(n)) για n→ ∞ , αν και μόνο αν υπάρχει ένας αριθμός c ⩾ 0 ϵ R
και ένας αριθμός $n_0 \geqslant 0 \in Z$ τέτοιοι ώστε :
T(n) ⩾ c g(n) για κάθε n > n_0.
Τότε, η T(n) είναι ασυμπτωτικά μεγαλύτερη ή ίση με τη g(n).

Big-Theta Notation :
Δεδομένων των συναρτήσεων *T*(*n*) και g(n) μπορεί να γραφεί ότι
T(n) = Θ(g(n)) για n→ ∞ , αν και μόνο αν υπάρχουν δύο αριθμοί c', $c'' \geqslant 0 \in R$
και ένας αριθμός $n_0 \geqslant 1 \in Z$ τέτοιοι ώστε :
c'' × g(n) ⩾ T(n) ⩾ c' × g(n) για κάθε n > n_0.
Τότε, η T(n) είναι ασυμπτωτικά ίση με τη g(n).

Oh Notation :
Δεδομένων των συναρτήσεων $T(n)$ και g(n) μπορεί να γραφεί ότι
T(n) = o(g(n)) για $n \to \infty$, αν και μόνο αν υπάρχει ένας αριθμός $c \geq 0 \in R$
και ένας αριθμός και ένας αριθμός $n_0 \geq 0 \in Z$ τέτοιοι ώστε :
$T(n) < c\ g(n)$ για κάθε $n > n_0$.
Τότε η T(n) είναι ασυμπτωτικά μικρότερη από τη g(n).

Omega Notation :
Δεδομένων των συναρτήσεων $T(n)$ και g(n) μπορεί να γραφεί ότι
T(n) = Ω(g(n)) για $n \to \infty$, αν και μόνο αν υπάρχει ένας αριθμός $c \geq 0 \in R$
και ένας αριθμός $n_0 \geq 0 \in Z$ τέτοιοι ώστε :
$T(n) > c\ g(n)$ για κάθε $n > n_0$.
Τότε, η T(n) είναι ασυμπτωτικά μεγαλύτερη από τη g(n).

Οι χρόνοι επίλυσης προβλημάτων εξαρτώνται από τη μορφή του αλγορίθμου και τον όγκο των δεδομένων. Η γενικότερη επιδίωξη είναι ο χρόνος υλοποίησης του αλγορίθμου να εξαρτάται όσο το δυνατόν λιγότερο από τον όγκο των δεδομένων ώστε να αποφεύγονται οι καθυστερήσεις κατά την εκτέλεση όταν ο όγκος των δεδομένων είναι μεγάλος. Εάν ο όγκος δεδομένων είναι n, τότε οι χρόνοι υλοποίησης ενός αλγορίθμου από το συντομότερο στον πιο αργό είναι :

Χρόνος Εκτέλεσης T(n)	
Σταθερός	**1**
Λογαριθμικός	$\boldsymbol{log_2 n}$
Γραμμικός	$\boldsymbol{n}$
Τετραγωνικός	$\boldsymbol{n^2}$
Πολυωνυμικός	$\boldsymbol{n^b}$**, όπου b** $\boldsymbol{\in Z}$
Εκθετικός	$\boldsymbol{n^n}$

Πίνακας 4.1.Ταξινόμηση χρόνων εκτέλεσης αλγορίθμων. Cormen et al. (2001)

4.4.ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΕΥΡΕΣΗΣ ΒΕΛΤΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ

Το πρόβλημα εύρεσης της ελάχιστης διαδρομής σε συγκοινωνιακά δίκτυα άρχισε να εξετάζεται από τη δεκαετία του 1960 και πάνω σε αυτό έχει δημοσιευθεί πλήθος ερευνών. Οι περισσότερες έρευνες που έχουν δημοσιευθεί έχουν ως αντικείμενό τους τη δημιουργία νέων αλγόριθμων εύρεσης της ελάχιστης διαδρομής σε στατικά δίκτυα, στα οποία ο χρόνος διάνυσης των συνδέσμων δε μεταβάλλεται. Τα τελευταία χρόνια βέβαια γίνεται προσπάθεια για δυναμική διαχείριση των συγκοινωνιακών συστημάτων με νέο στοιχείο ότι οι χρόνοι διάνυσης των συνδέσμων δεν είναι πάντα σταθεροί αλλά μεταβάλλονται με το χρόνο. Σε αυτή την κατηγορία ανήκουν τα δυναμικά προβλήματα εύρεσης ελάχιστης διαδρομής. Τα προβλήματα αυτά μπορούν να κατηγοριοποιηθούν ανάλογα με τη φύση τους ως εξής, όπως προτάθηκε από τον Chabini (1997) :

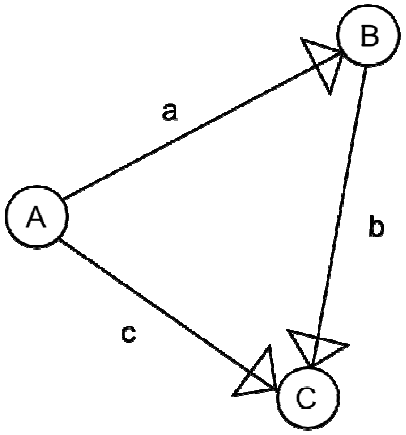
- Προβλήματα εύρεσης της ελάχιστης διαδρομής στα οποία οι χρόνοι διάνυσης ή τα κόστη διάνυσης των συνδέσμων θεωρούνται σταθερά και δε μεταβάλλονται στο χρόνο «Στατικά προβλήματα» ή «Δυναμικά προβλήματα» στα οποία υπάρχουν μεταβολές στα κόστη και στους χρόνους διάνυσης των συνδέσμων με την πάροδο του χρόνου.
- Προβλήματα που επικεντρώνονται στην εύρεση της ελάχιστης διαδρομής ή προβλήματα που επικεντρώνονται στην εύρεση της οικονομικότερης διαδρομής ή της διαδρομής που πληροί κάποια άλλα κριτήρια.
- Δυναμικά προβλήματα στα οποία θεωρείται ότι οι χρόνοι διάνυσης των συνδέσμων μεταβάλλονται ανά διακεκριμένα χρονικά διαστήματα ή μεταβάλλονται συνεχώς με το χρόνο.
- Δίκτυα που ικανοποιούν τη συνθήκη FIFO=(First In First Out) που δηλώνει ότι μεταφορικό μέσο που ξεκινά νωρίτερα από έναν κόμβο i με κατεύθυνση έναν κόμβο j θα φτάσει σίγουρα νωρίτερα στον κόμβο j από ότι αν ξεκινούσε αργότερα.
- Δίκτυα στα οποία οι καθυστερήσεις στους κόμβους επιτρέπονται : Waits are allowed (WA) και δίκτυα στα οποία οι καθυστερήσεις στους κόμβους δεν επιτρέπονται : Waits are not allowed (WN).
- Προβλήματα στα οποία δεν προτείνεται μόνο μία διαδρομή στο χρήστη αλλά όλες οι δυνατές διαδρομές με μία σειρά ιεραρχίας (για παράδειγμα από τη συντομότερη προς τις υπόλοιπες).
- Προβλήματα με ακέραιες και προβλήματα με πραγματικές τιμές στους χρόνους διάνυσης των συνδέσμων.

4.5.ΜΕΘΟΔΟΙ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΣΥΓΚΟΙΝΩΝΙΑΚΩΝ ΔΙΚΤΥΩΝ ΣΕ Η/Υ

Το δίκτυο μελέτης μπορεί να είναι οποιοδήποτε συγκοινωνιακό δίκτυο, όπως για παράδειγμα οδικό ή αερομεταφορών κ.τ.λ. Αρχικά στο δίκτυο γίνεται επιλογή κάποιων θέσεων οι οποίες καλούνται κόμβοι του δικτύου και μπορεί να είναι τα κέντρα των ζωνών σε αστικές περιοχές, οι διασταυρώσεις, οι πόλεις, τα λιμάνια, τα αεροδρόμια κ.α. ανάλογα με τη μορφή του δικτύου και την κλίμακα προσομοίωσης. Όταν μεταξύ δύο καθορισμένων θέσεων του δικτύου (κόμβων) υπάρχει οποιασδήποτε μορφής σύνδεση, τότε η σύνδεση αυτή θα καλείται ως σύνδεσμος των δύο κόμβων. Έτσι το πραγματικό δίκτυο μπορεί να προσομοιωθεί με ένα γράφημα το οποίο θα περιέχει κόμβους που αντιπροσωπεύουν επιλεγμένες θέσεις του δικτύου και συνδέσμους που συνδέουν αυτούς τους κόμβους όπου υπάρχει σύνδεση. Οι κόμβοι συμβολίζονται με ακέραιους αριθμούς όπως και οι σύνδεσμοι. Κάθε σύνδεσμος έχει ένα βάρος το οποίο αντιπροσωπεύει τη δυσκολία μετακίνησης από έναν κόμβο σε έναν άλλο και μπορεί να αντιστοιχεί σε οικονομικό κόστος, χρόνο μετακίνησης κ.α. Το βάρος μπορεί να πάρει τιμές στο σύνολο των πραγματικών αριθμών.

Το γράφημα που δημιουργείται μέσω της προσομοίωσης του πραγματικού δικτύου αποτελεί την εικόνα του προβλήματος που πρέπει να επιλυθεί. Πάνω στο γράφημα μπορεί να αναπτυχθούν διάφορα προβλήματα, όπως η εύρεση της ελάχιστης διαδρομής μεταξύ δύο κόμβων κ.α. Τα περισσότερα προβλήματα επιλύονται με χρήση αλγορίθμων. Εάν το μέσο υλοποίησης των αλγορίθμων είναι ο ανθρώπινος εγκέφαλος το γράφημα αποτελεί από μόνο του το πεδίο επίλυσης του προβλήματος. Βέβαια οι περιπτώσεις που μπορεί να συμβεί αυτό είναι πολύ λίγες γιατί κάτι τέτοιο είναι ιδιαίτερα χρονοβόρο και σε πολύπλοκα δίκτυα μπορεί να είναι και ακατόρθωτο. Έτσι γίνεται χρήση του Η/Υ ως μέσο υλοποίησης των αλγορίθμων. Για να γίνει αυτό βέβαια πρέπει το γράφημα να μεταφραστεί σε γλώσσα μηχανής. Υπάρχουν διάφοροι τρόποι για να γίνει αυτό μερικοί από τους οποίους είναι :

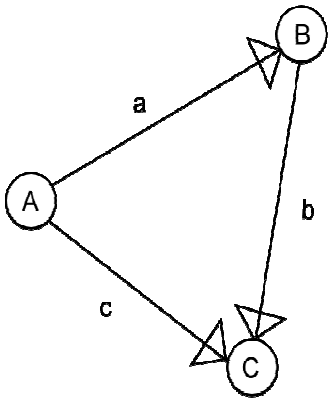
1) Αναπαράσταση γραφήματος με Πίνακα Γειτνίασης (adjacency matrix) : με αυτήν τη μέθοδο θεωρείται αρχικά ότι υπάρχουν σύνδεσμοι μεταξύ όλων των κόμβων του δικτύου. Εάν κάποιος σύνδεσμος στην πραγματικότητα δεν υπάρχει παίρνει, τότε το βάρος του ισούται με άπειρο. Η μέθοδος αυτή χρησιμοποιείται σε πυκνά δίκτυα (dense networks) με μεγάλο αριθμό συνδέσμων. Ο χώρος αποθήκευσης του πίνακα είναι $\Theta(N^2)$ όπου N ο αριθμός των κόμβων. Παρουσιάζει το πλεονέκτημα ότι μπορεί να ελεγθεί αμέσως η σύνδεση μεταξύ δύο κόμβων αλλά έχει και σοβαρό μειονέκτημα τη χρήση επιπλέον αποθηκευτικού χώρου για συνδέσμους που στην πραγματικότητα δεν υπάρχουν. Η γενική μορφή ενός τέτοιου πίνακα είναι :



	A	B	C
A	-	a	c
B	+∞	-	b
C	+∞	+∞	-

Σχήμα 4.2.Αναπαράσταση γραφήματος σε μορφή Πίνακα γειτνίασης

2) Αναπαράσταση γραφήματος με Λίστα Γειτνίασης (adjacency list) : με αυτήν τη μέθοδο αποθηκεύονται μόνο οι πραγματικοί σύνδεσμοι και ο χώρος που απαιτείται για την αποθήκευσή τους είναι Θ(Μ), όπου Μ ο αριθμός των συνδέσμων. Χρησιμοποιείται σε αραιά δίκτυα (sparse networks) με μικρό αριθμό συνδέσμων αντί του πίνακα γειτνίασης που έχει μεγάλο άσκοπο κόστος σε τέτοιες περιπτώσεις. Οι λίστες γειτνίασης που χρησιμοποιούνται για την αναπαράσταση ενός γραφήματος είναι ίσες με τον αριθμό των κόμβων που συνδέονται με άλλους κόμβους. Για κάθε κόμβο από τον οποίο υπάρχουν σύνδεσμοι προς άλλους κόμβους αντιστοιχεί μία λίστα γειτνίασης που είναι ένας μονοδιάστατος πίνακας (λίστα) στην οποία περιέχονται οι κόμβοι προορισμού και τα βάρη σύνδεσης των αντίστοιχων συνδέσμων.



Κόμβος Προέλευσης	Κόμβος Προορισμού	Αριθμός Συνδέσμου	Βάρος Συνδέσμου
A	B	1	a
A	C	2	c
B	C	3	b

Σχήμα 4.3.Αναπαράσταση γραφήματος σε μορφή Λίστας γειτνίασης

ΚΕΦΑΛΑΙΟ 5^{ο}

ΑΝΑΠΤΥΞΗ ΒΑΣΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ

5.1.ΣΚΟΠΟΣ ΚΕΦΑΛΑΙΟΥ

Πολλές φορές ο συγκοινωνιολόγος μηχανικός καλείται να επιλύσει σύνθετα προβλήματα λαμβάνοντας χιλιάδες στοιχεία από βάσεις δεδομένων στις οποίες είναι αποθηκευμένα. Τα προβλήματα αυτά μπορούν να επιλυθούν με έξυπνες μεθόδους με χρήση του Η/Υ. Ειδικά στο χώρο των συγκοινωνιακών δικτύων η ανάγκη για την εύρεση τέτοιων μεθόδων είναι μεγάλη. Ο όγκος της έρευνας πάνω σε αυτό τον τομέα που ξεκινά από τα μέσα του 20^{ου} αιώνα και συνεχίζεται με αμείωτο ενδιαφέρον μέχρι και σήμερα είναι ενδεικτικός της σπουδαιότητας αυτών των μεθόδων. Ένα από τα μεγαλύτερα προβλήματα σε αυτό το χώρο είναι ότι έχουν γίνει τόσες μελέτες που συχνά ένας ερευνητής μπορεί να επιλύσει κάποιο πρόβλημα και να προτείνει κάποια μέθοδο χωρίς να γνωρίζει ότι κάτι παρόμοιο έχει προταθεί από κάποιον άλλο ή ότι αυτό που προτείνει έχει ήδη ξεπεραστεί. Ο S.Dreyfus (1969) είχε αναφέρει ότι μέσα σε λιγότερο από 20 χρόνια απ'όταν που ξεκίνησε η έρευνα σε αυτό τον τομέα πολλοί ερευνητές δεν αξιοποιούσαν τα αποτελέσματα από άλλες έρευνες και πρότειναν ξεπερασμένες μεθόδους. Το πρόβλημα αυτό έχει αυξηθεί ακόμη περισσότερο στις μέρες μας όπου η ανάγκη για την εξεύρεση νέων μεθόδων είναι πιο έντονη από ποτέ. Στο κεφάλαιο αυτό γίνεται μία προσπάθεια παρουσίασης μερικών από τους βασικότερους αλγορίθμους που εφαρμόζονται σε συγκοινωνιακά προβλήματα όπως :

α) Εύρεση των συντομότερων διαδρομών από έναν κόμβο προς όλους τους υπόλοιπους κόμβους ενός συγκοινωνιακού δικτύου (Single Source Shortest Path Problem) – Ενότητες 5.2.1., 5.2.2., 5.2.3, 5.2.4.

β) Εύρεση των συντομότερων διαδρομών από κάθε κόμβο ενός συγκοινωνιακού δικτύου προς τον κόμβο τελικού προορισμού – Ενότητα 5.2.5.

γ) Εύρεση της συντομότερης διαδρομής από έναν κόμβο προέλευσης προς έναν κόμβο προορισμού σε ένα συγκοινωνιακό δίκτυο (Point to Point Shortest Path Problem) – Ενότητες 5.4.1, 5.4.2, 5.4.3

δ) Εύρεση των συντομότερων διαδρομών από όλους τους κόμβους προς όλους τους κόμβους σε ένα συγκοινωνιακό δίκτυο (All Pairs Shortest Path Problem) – Ενότητες 5.5.1., 5.5.2.

ε) Εύρεση των k συντομότερων διαδρομών από έναν κόμβο προέλευσης προς έναν κόμβο προορισμού σε ένα συγκοινωνιακό δίκτυο (kth Shortest Paths Problem) – Ενότητες 5.6.1., 5.6.2., 5.6.3.

στ) Εύρεση της συντομότερης διαδρομής από έναν κόμβο προέλευσης προς έναν κόμβο προορισμού η οποία ικανοποιεί ταυτόχρονα ορισμένους περιορισμούς (Resource Constrained Shortest Path Problem) – Ενότητα 5.7.

Επίσης, στο κεφάλαιο αυτό πραγματοποιείται μία σύντομη περιγραφή στα στοιχεία και στη μορφή ορισμένων αλγορίθμων που αναφέρονται στα παραπάνω προβλήματα. Έτσι γίνεται μία ουσιαστική αναδρομή σε αυτόν το χώρο. Εκτός από την περιγραφή των βασικότερων αλγορίθμων, παρουσιάζεται και η υλοποίησή τους μέσω κάποιας γλώσσας προγραμματισμού (Fortran 95, Java, C++). Τα προγράμματα που δημιουργούνται παρουσιάζονται στο Παράρτημα της εργασίας. Μέσω των προγραμμάτων αυτών γίνονται και κάποιες συγκρίσεις μεταξύ των αλγορίθμων σχετικά με το χρόνο εκτέλεσής τους και τις απαιτούμενες θέσεις μνήμης κατά την εφαρμογή τους σε συγκοινωνιακά δίκτυα. Αξίζει να σημειωθεί ότι πολλοί από αυτούς χρησιμοποιούνται σήμερα σε διάφορες εφαρμογές όπως σε συστήματα πλοήγησης, ευφυή συστήματα μεταφορών, εξισορρόπηση των συγκοινωνιακών δικτύων κ.α.

Πριν γίνει όμως η ανάλυσή τους θα παρουσιαστεί μία μορφή συμβολισμού των συγκοινωνιακών δικτύων. Ένα συγκοινωνιακό δίκτυο αποτελείται από κόμβους και συνδέσμους. Ως κόμβοι συνήθως θεωρούνται οι διασταυρώσεις των οδών και ως σύνδεσμοι οι οδικές αρτηρίες. Έτσι παρουσιάζονται ορισμένοι συμβολισμοί που θα χρησιμοποιηθούν στις επόμενες ενότητες αυτού του κεφαλαίου κατά την παρουσίαση διάφορων αλγορίθμων. Ένα δίκτυο G = {N,E} αποτελείται από N κόμβους και E συνδέσμους. Κάθε σύνδεσμος eϵE έχει κόμβο προέλευσης S(e) και κόμβο προορισμού E(e). Σύμφωνα με αυτά κάθε διαδρομή p μεταξύ δύο κόμβων αποτελείται από ένα σύνολο συνδέσμων p = $(e_x, \dots\dots, e_z)$ όπου $e_i \in$E για τους οποίους ισχύει : E(e_i) = S(e_{i+1}) $\forall$i.

Το κόστος των συνδέσμων του δικτύου G = {N,E} είναι μία συνάρτηση w(e) : E$\rightarrow R^+$ όπου το w(e) είναι το μη αρνητικό κόστος του συνδέσμου e$\in$E. Η συνάρτηση αυτή έχει κόστος ανάλογα με διάφορα κριτήρια όπως ο χρόνος διάνυσης του συνδέσμου, το κόστος διάνυσης κ.α. Σύμφωνα με αυτόν το συμβολισμό το κόστος διάνυσης μίας διαδρομής p = $(e_x, \dots\dots, e_z)$ είναι :

D(p) = $\sum_{e \in p} w(e)$

5.2.ΕΥΡΕΣΗ ΤΩΝ ΒΕΛΤΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ ΑΠΟ ΕΝΑΝ ΚΟΜΒΟ ΠΡΟΕΛΕΥΣΗΣ ΠΡΟΣ ΟΛΟΥΣ ΤΟΥΣ ΥΠΟΛΟΙΠΟΥΣ ΚΟΜΒΟΥΣ

Το πρόβλημα αυτό έχει απασχολήσει πολλούς μελετητές και υπάρχουν πολλές εργασίες πάνω σε αυτό όπως : Moore (1957), Bellman (1958), Ford and Fulkerson (1968), Dijkstra (1959), Pollack and Wiebenson (1960), Dantzig (1960), Pape (1974), Fredman and Tarjan (1987), Gallo and Pallottino (1988), Raman (1997), Han and Thorup (2002). Η γενικότερη μορφή του προβλήματος μπορεί να δοθεί ως :

Κάθε κόμβος i ϵ N-{r} προσεγγίζεται από τον κόμβο προέλευσης r με κόστος D(i) : N-{r}→R^+. Εάν τυχαία διαδρομή από το r στο i αποτελείται από τους συνδέσμους p = (e_x, e_y ... ., e_z), θεωρώντας σύνολο P = {x, y, ……, z}, πρέπει :
$S(e_x)$ = r
E(e_z) = i
∀ k ϵ P-{z} : E(e_k) = S(e_{k+1})
Έτσι το κόστος της τυχαίας διαδρομής r→i δίνεται από τη σχέση :
D(i) = $\sum_{e \epsilon p} w(e)$.
Το ζητούμενο είναι να βρεθεί η διαδρομή p_* για την οποία,
D_* = min{ D(i) } = min{ $\sum_{e \epsilon p} w(e)$ }, ∀p.
Αυτή η διαδικασία επαναλαμβάνεται για ∀ i ϵ N-{r}, δηλαδή για κάθε κόμβο.

Στη συνέχεια της ενότητας παρουσιάζονται ορισμένοι από τους πιο γνωστούς αλγόριθμους επίλυσης αυτού του προβλήματος.

5.2.1.Ο ΑΛΓΟΡΙΘΜΟΣ MOORE

Ένας από τους πρώτους αλγόριθμους, επίκαιρος ακόμα και σήμερα λόγω της απλότητάς του δόθηκε από την εργασία του Moore (1957). Ο αλγόριθμος αυτός βρήκε ευρεία εφαρμογή στα συστήματα μεταφορών και τις τηλεπικοινωνίες. Αξίζει να σημειωθεί ότι όλες οι εργασίες για την ανάπτυξη παρόμοιων αλγορίθμων ξεκινούν μόλις λίγα χρόνια πριν. Αυτό δεν είναι τυχαίο καθώς η χρησιμότητά τους συνδέεται με την εμφάνιση των ηλεκτρονικών υπολογιστών. Ειδικότερα, ο αλγόριθμος Moore χρησιμοποιείται για την εύρεση των ελαχίστων διαδρομών από έναν κόμβο *i* προς τους υπόλοιπους κόμβους ενός δικτύου. Ανήκει δηλαδή στην κατηγορία των αλγορίθμων επίλυσης των Single Source Shortest Path Problems. Με μικρές τροποποιήσεις μπορεί και να επιλύσει προβλήματα εύρεσης της ελάχιστης διαδρομής μεταξύ δύο κόμβων (Point to Point Shortest Path Problems).

Η μορφή του αλγορίθμου Moore, ο οποίος ανήκει στην κατηγορία των αλγορίθμων δυναμικού προγραμματισμού είναι :

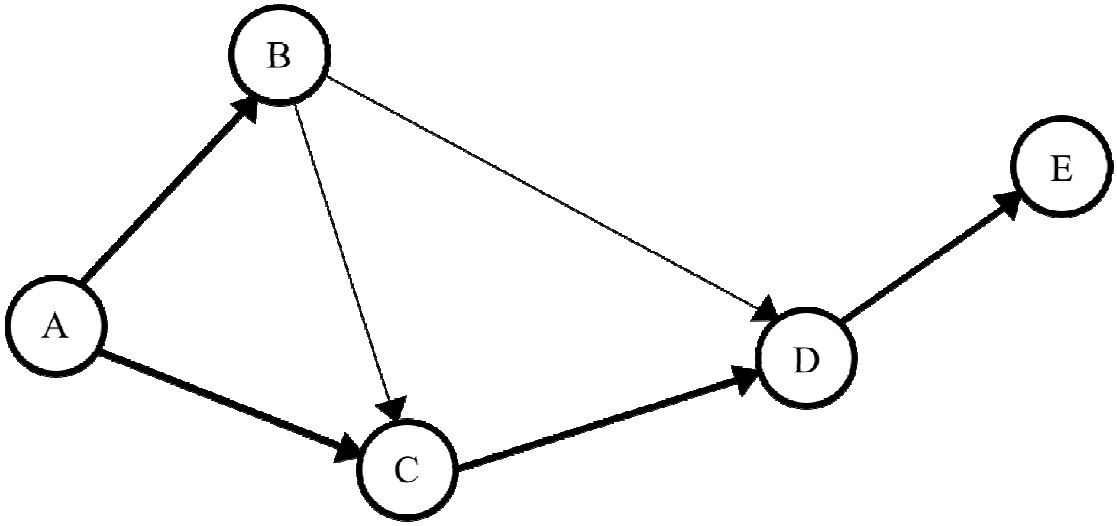
ΑΛΓΟΡΙΘΜΟΣ MOORE

Βήμα 1(Αρχικοποιήσεις) #Βασικές Πράξεις
Για Κάθε i = 1,N
D (i) ← +∞ // Όπου D (i) το ελάχιστο κόστος μιάς διαδρομής από τον αρχικό κόμβο r στον τελικό κόμβο i N
Τέλος Για Κάθε
D (r) ← 0 // Η μετακίνηση r→r έχει ελάχιστο κόστος ίσο με το μηδέν
Βήμα 2(Αλγόριθμος Moore)
Για Κάθε k = 1,N
Για Κάθε e = 1,E $N \times E$
Εάν D(E(e)) > D(S(e)) + w(e)
D(E(e)) ← D (S(e)) + w(e)
Τέλος Εάν
Τέλος Για Κάθε
Τέλος Για Κάθε

Στο τέλος της παραπάνω διαδικασίας το κόστος D(i) που προκύπτει για κάθε κόμβο iϵN είναι το μικρότερο δυνατό. Ο χρόνος εκτέλεσης αυτού του αλγορίθμου είναι $O(\mathrm{N} \times \mathrm{E})$. Σε πολλά γραφήματα όμως ο χρόνος αυτός αποδεικνύεται πολύ μικρότερος καθώς ο αλγόριθμος μπορεί να μην τερματίσει στη Νιοστή επανάληψη, αλλά σε κάποια μικρότερη επανάληψη (**φ**), εάν κατά την επανάληψη αυτή δεν αναβαθμιστεί καμία τιμή του κόστους D(i), ∀iϵN. Έτσι προκύπτει χρόνος εκτέλεσης $O(\boldsymbol{\varphi} \times \mathrm{E})$. Ο αλγόριθμος του Moore είναι παρόμοιος με τον αλγόριθμο των Bellman-Ford που διατυπώθηκε εκείνη την περίοδο και πολλές φορές εμφανίζονται στη βιβλιογραφία ως ο ίδιος αλγόριθμος.

5.2.2.Ο ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA

Ο αλγόριθμος αυτός παρουσιάστηκε από την εργασία του Ολλανδού επιστήμονα υπολογιστών και καθηγητή στο Stanford Edsger Dijkstra (1959). Ανήκει στην κατηγορία των άπληστων αλγορίθμων και έχει σημαντικές διαφορές στην εκτέλεση από τον αλγόριθμο που αναπτύχθηκε από τον Moore (1957). Πιο αναλυτικά σε κάθε στάδιο επιλέγεται η τοπική βέλτιστη λύση, δηλαδή ο πλησιέστερος κόμβος στον αρχικό που οδηγεί στην εύρεση της καθολικής βέλτιστης λύσης. Ο αλγόριθμος αυτός αφού βρει την τοπική βέλτιστη λύση προχωρά στην εύρεση της επόμενης βέλτιστης λύσης, δηλαδή τον αμέσως επόμενο πλησιέστερο κόμβο στον αρχικό χωρίς να εξετάσει ξανά τη λύση που έχει ήδη βρεθεί. Αυτή είναι και η βασική διαφορά με τον αλγόριθμο Moore και το δυναμικό προγραμματισμό γενικότερα όπου σε κάθε νέο βήμα αναζητείται το βέλτιστο με βάση τις παρούσες συνθήκες, επανεξετάζοντας δηλαδή και τους κόμβους που έχουν ήδη επιλεγεί. Στο παρακάτω σχήμα γίνεται μία παρουσίαση της παραπάνω αρχής :



Σχήμα 5.1.Δίκτυο για την ανάλυση των Άπληστων αλγορίθμων

Θεωρώντας κόμβο προέλευσης τον κόμβο A, τότε D(A) = 0. Σε μία πρώτη επανάληψη προκύπτει ότι το κόστος μετακίνησης από το A στο B : D(B) = D(A) + w(A,B) είναι μικρότερο από το αντίστοιχο κόστος D(C) = D(A) + w(A,C). Τότε το κόστος D(B) είναι το ελάχιστο δυνατό και δε χρειάζεται να επανελεγχθεί. Αυτή είναι η έννοια του τοπικού βέλτιστου. Το κόστος D(B) δε χρειάζεται να επανελεγχθεί διότι δε μπορεί να έχει μικρότερη τιμή από αυτήν. Αν γινόταν αυτό θα έπρεπε να υπάρχει πιθανότητα να ισχύει D(C) + w(C,B) < D(B). Κάτι τέτοιο όμως δε μπορεί να ισχύει γιατί D(C) > D(B), άρα D(C) + w(C,B) > D(B) $\forall C \epsilon N$, εκτός και αν υπάρχουν σύνδεσμοι με αρνητικό βάρος w(C,B) < 0. Τότε η ιδιότητα του τοπικού βέλτιστου δεν έχει εφαρμογή και ο αλγόριθμος Dijkstra δε μπορεί να χρησιμοποιηθεί. Ωστόσο η περίπτωση αυτή δεν απασχολεί τα συγκοινωνιακά δίκτυα.

Από άποψη χρόνου εκτέλεσης μπορεί να υπάρξει βελτίωση σε σχέση με τον αλγόριθμο του Moore γιατί σε κάθε βήμα το επόμενο πλησιέστερο σημείο στον κόμβο προέλευσης τοποθετείται σε ένα κλειστό σύνολο και δεν επανελέγχεται στη συνέχεια. Λόγω αυτής της ιδιότητας ανήκει άλλωστε και στην κατηγορία των άπληστων αλγορίθμων. Η μορφή του αλγορίθμου Dijkstra είναι :

ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA

Βήμα 1(Αρχικοποιήσεις)

#Βασικές Πράξεις

Για Κάθε i = 1,N
D (i) ← +∞
Q (i) ← +∞
Τέλος Για Κάθε
D (r) ← 0 & Q (r) ← r
Βήμα 2(Αλγόριθμος Dijkstra)
u ← r
Για Κάθε j = 1,N

```
  Για Κάθε e = 1,E                                          E × N
    Εάν S( e ) = u & D( E(e) ) > D( S(e) ) + w( e )
    D( E(e) ) ← D( S(e) ) + w( e )
    Τέλος Εάν
  Τέλος Για Κάθε
  K ← +∞
  Για Κάθε i = 1,N
     Εάν Q( i ) ≠ i & D( i ) < K
     K ← D( i )
     u ← i
     Τέλος Εάν
  Τέλος για Κάθε
  Q(u) ← u
Τέλος για Κάθε
```

Σύμφωνα με τα παραπάνω ο χρόνος εκτέλεσης του αλγορίθμο Dijkstra είναι $O(\mathrm{N} \times \mathrm{E})$. Εάν όμως η αναπαράσταση του δικτύου γίνει μέσω Πίνακα Γειτνίασης (adjacency matrix), τότε ο χρόνος εκτέλεσης μειώνεται σε $O(\mathrm{N}^2)$, όμως οι απαιτούμενες θέσεις μνήμης για την αποθήκευση του βάρους κάθε συνδέσμου αυξάνονται από $\Theta(3 \times \mathrm{E})$ σε $\Theta(N^2)$, καθώς αποθηκεύονται ακόμα και οι σύνδεσμοι που δεν υπάρχουν στο δίκτυο με βάρος = $+\infty$. Για να βρεθούν τα δένδρα ελάχιστων διαδρομών η διαδικασία εκτελείται προς τα πίσω από κάθε κόμβο προς τον αρχικό r.

Το πιο σημαντικό στο χρόνο εκτέλεσης είναι ότι το σύνολο Q (Ουρά προτεραιότητας ή Priority queue) στο οποίο είναι αποθηκευμένοι όλοι οι κόμβοι του δικτύου είναι μία μονοδιάστατη μήτρα (ουρά) με N στοιχεία. Έτσι κάθε φορά που αναζητείται ο επόμενος πλησιέστερος κόμβος στον αρχικό r απαιτείται το πολύ χρόνος O(N), αφού ελέγχονται όλοι οι κόμβοι που βρίσκονται στην ουρά.

Γενικά έχουν προταθεί πολλές βελτιώσεις στις δομές δεδομένων που χρησιμοποιεί ο αλγόριθμος Dijkstra με αποτέλεσμα την περαιτέρω μείωση του χρόνου εκτέλεσής του. Κάτι τέτοιο όμως εξαρτάται σε μεγάλο βαθμό από το πόσο πυκνή είναι η δομή του δικτύου, καθώς σε πολύ πυκνά δίκτυα η κλασική του μορφή μπορεί να είναι και η πιο αποδοτική. Μερικές από τις βελτιώσεις αυτές αναλύονται παρακάτω.

5.2.3.Ο ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA ΜΕ ΒΕΛΤΙΩΣΕΙΣ ΣΤΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Όπως έχει αναφερθεί ήδη στην προηγούμενη ενότητα ο χρόνος εκτέλεσης του αλγορίθμου Dijkstra εάν το δίκτυο αναπαρασταθεί με Πίνακα Γειτνίασης είναι $O(N^2)$. Ο χρόνος αυτός εξαρτάται από τη μέθοδο αναπαράστασης του δικτύου, κάτι που θα εξεταστεί εκτενέστερα στην ενότητα 5.8. και από το χρόνο που απαιτείται για τον εντοπισμό του επόμενου πλησιέστερου κόμβου στον r σε κάθε βήμα της

επανάληψης. Κατά την κλασική διατύπωση του αλγορίθμου Dijkstra απαιτούνται N συγκρίσεις για τον εντοπισμό αυτού του κόμβου σε κάθε βήμα της επανάληψης :

```
K ← +∞
Για Κάθε i = 1,N
   Εάν D( i ) < K  και ο κόμβος i δεν έχει επιλεγεί από προηγούμενη επανάληψη
      K ← D( i )
      u ← i
   Τέλος Εάν
Τέλος για Κάθε
```

Από τα παραπάνω είναι εμφανές ότι γίνεται στην ουσία αναζήτηση της ελάχιστης τιμής σε στοιχεία που ανήκουν σε μία μήτρα, όπως στο ακόλουθο παράδειγμα :

Κόμβος	1	2	3	4	5	6	7	8	9	10	11	12
D(i)	1.2	4	6	13	7	8	5	2	11	17	19	21

Έτσι έπειτα από N συγκρίσεις προκύπτει ότι ο επόμενος πλησιέστερος κόμβος στον r είναι ο κόμβος 1 με κόστος D(1) = 1.2.

Στη συνέχεια παρουσιάζονται διάφορες Δομές Δεδομένων ώστε να μειωθεί ο χρόνος της παραπάνω αναζήτησης. Περισσότερες πληροφορίες σχετικά με τις δομές δεδομένων υπάρχουν στο βιβλίο : Γεωργακόπουλος (2008).

5.2.3.1.Δυαδική αναζήτηση και προεκτάσεις

Αρχικά προτείνεται οι κόμβοι να αποθηκεύονται ταξινομημένοι κατά αύξουσα σειρά ανάλογα με την τιμή D(i), όπως παρουσιάζεται ακολούθως :

Θέση	1	2	3	4	5	6	7	8	9	10	11	12
Κόμβος	1	8	2	7	3	5	6	9	4	10	11	12
D(i)	1.2	2	4	5	6	7	8	11	13	17	19	21

Με βάση αυτή τη δομή προκύπτει αμέσως ο επόμενος κόμβος με τη μικρότερη τιμή D(i), ο οποίος είναι ο κόμβος 1 με D(1) = 1.2 καθώς βρίσκεται στη θέση 1. Άρα απαιτείται χρόνος εκτέλεσης μόλις $O(1)$. Για να διατηρείται όμως η ταξινόμηση των κόμβων κατά αύξουσα σειρά του κόστους D(i) απαιτούνται επιπλέον βασικές πράξεις στην περίπτωση αλλαγής της τιμής κόστους ενός κόμβου, προσθήκης ενός κόμβου ή διαγραφής ενός κόμβου.

α) Διαγραφή ενός στοιχείου. Το στοιχείο που διαγράφεται σε κάθε επανάληψη του αλγορίθμου Dijkstra είναι ο κόμβος που βρίσκεται στη θέση 1 της δομής. Ο χρόνος για τη διαγραφή του στοιχείου είναι στη χειρότερη περίπτωση $O(N)$, όσες δηλαδή και οι απαιτούμενες μετατοπίσεις των υπόλοιπων κόμβων.

β) Εισαγωγή ενός κόμβου. Στην περίπτωση εισαγωγής ενός επιπλέον κόμβου θα εξεταστούν οι απαιτούμενες βασικές πράξεις. Έστω ότι στο παραπάνω παράδειγμα προστίθεται ο κόμβος 13 με κόστος : D(13) = 14. Αρχικά ο κόμβος αυτός καταλαμβάνει την τελευταία θέση. Λόγω της αποθήκευσης των κόμβων κατά αύξουσα σειρά μπορεί να βρεθεί η νέα θέση του κόμβου 13 ώστε να διατηρείται η συνθήκη της ταξινόμησης κατά αύξουσα σειρά με χρήση της δυαδικής αναζήτησης :

Θέση	1	2	3	4	5	**6**	7	8	9	10	11	12
Κόμβος	1	8	2	7	3	**5**	6	9	4	10	11	12
D(i)	1.2	2	4	5	6	**7**	8	11	13	17	19	21

D(13)>D(13/2)

Νέο Πεδίο Αναζήτησης

Θέση	7	8	**9**	10	11	12
Κόμβος	6	9	**4**	10	11	12
D(i)	8	11	**13**	17	19	21

D(13)>D(9)

Νέο Πεδίο Αναζήτησης

Θέση	10	**11**	12
Κόμβος	10	**11**	12
D(i)	17	**19**	21

D(13)<D(11)

Νέο Πεδίο Αναζήτησης

Θέση	**10**
Κόμβος	**10**
D(i)	**17**

D(13)<D(10)

Οπότε ο κόμβος 13 βρέθηκε ότι πρέπει να μετακινηθεί στη θέση 10 ώστε να μην παραβιάζεται η συνθήκη της ταξινόμησης κατά αύξουσα σειρά. Αυτό έγινε σε μόλις τέσσερις συγκρίσεις. Γενικότερα για την ταξινόμηση ενός κόμβου μέσω της δυαδικής αναζήτησης απαιτούνται log_2N συγκρίσεις, όπου N ο συνολικός αριθμός των στοιχείων καθώς το πεδίο αναζήτησης διχοτομείται συνεχώς. Αντίθετα με μία απλή αναζήτηση σε λίστα απαιτούνται N συγκρίσεις όπως έχει διατυπωθεί ήδη. Το πρόβλημα αυτής της μεθόδου είναι ότι απαιτείται αρκετά μεγάλος αριθμός βασικών πράξεων στη συνέχεια ώστε να μετακινηθούν όλοι οι κόμβοι που έχουν μεγαλύτερο κόστος από το κόστος του κόμβου 13 κατά μία θέση δεξιά. Σύμφωνα με το παράδειγμα κάθε κόμβος βρίσκεται σε μία θέση που είναι αποθηκευμένη σε μία

μήτρα. Εάν θεωρηθεί ότι η μήτρα αυτή είναι ένα σύνολο Q, όπου Q(j) = m με j τη θέση στη μήτρα και m τον αριθμό του κόμβου που βρίσκεται σε αυτή τη θέση, τότε :

Q(j)	1	8	2	7	3	5	6	9	4	10	11	12
D(Q(j))	1.2	2	4	5	6	7	8	11	13	17	19	21

Επομένως εάν ο κόμβος 13 μετακινηθεί στη θέση 10, τότε πρέπει να γίνουν και οι ακόλουθες μετακινήσεις :

Για Κάθε i=10,12
Q(i+1)=Q(i)
Τέλος Για Κάθε

Έτσι οι κόμβοι 10, 11, 12 μετακινούνται στις θέσεις 11, 12, 13 αντίστοιχα. Απαιτούνται δηλαδή επιπλέον 3 βασικές πράξεις και εάν ο κόμβος 13 μετακινούταν στη θέση 1 θα απαιτούνταν N επιπλέον βασικές πράξεις. Κάτι τέτοιο όμως δεν είναι αποδεκτό. Για το λόγο αυτό η θέση κάθε κόμβου μπορεί να διατηρείται σε διασυνδεόμενη αλυσίδα και όχι σε μήτρα. Κάθε κόμβος μπορεί να έχει έναν και μόνο απόγονο ο οποίος βρίσκεται ακριβώς μία θέση δεξιά του. Έτσι κατά τη μετακίνηση του κόμβου 13 στη θέση 10 απαιτείται να αλλάξουν οι τιμές σε 2 μόλις απογόνους. Ο απόγονος του κόμβου που βρίσκεται στη θέση 9 θα είναι πλέον ο κόμβος 13 και ο απόγονος του κόμβου 13 θα είναι αντίστοιχα ο κόμβος που βρισκόταν πριν την αναβάθμιση στη θέση 10. Έτσι η εισαγωγή ενός κόμβου απαιτεί το πολύ $log_2 N$ συγκρίσεις και 2 μετακινήσεις. Για τη δημιουργία λοιπόν της παραπάνω δομής μπορεί να απαιτηθούν έως και $N log_2 N$ βασικές πράξεις. Το πρόβλημα όμως παραμένει και με τη δομή αλυσίδας καθώς δε μπορεί να εφαρμοστεί πλέον η δυαδική αναζήτηση διότι δε μπορεί να είναι γνωστή η ακριβής θέση των κόμβων σε χρόνο $O(1)$ ώστε να γίνουν οι συγκρίσεις όπως στο παράδειγμα. Ως τελευταία λύση μπορεί να προταθεί η χρήση κυκλικών ταξινομημένων πινάκων. Έτσι τα στοιχεία δεν αποθηκεύονται σε έναν πίνακα κατά αύξουσα σειρά αλλά σε έναν κυκλικό πίνακα που μπορεί να θεωρηθεί ότι αποτελείται από επιμέρους υποπίνακες. Ένα παράδειγμα της δομής του κυκλικού πίνακα είναι το ακόλουθο :

Θέση	1	2	3	4	5	6	7	8	9	10	11	12
Κόμβος	5	6	9	4	10	11	12	1	8	2	7	3
D(i)	**7**	**8**	**11**	**13**	**17**	**19**	**21**	**1.2**	**2**	**4**	**5**	**6**
			→				Max	Min		→		

Έτσι η δυαδική αναζήτηση μπορεί να εφαρμοστεί στους δύο υποπίνακες ξεχωριστά αντί να εφαρμοστεί συνολικά στον κυκλικό πίνακα. Ο χρόνος εισαγωγής ενός στοιχείου είναι πλέον μικρότερος. Εάν για παράδειγμα εισάγεται ένα στοιχείο με κόστος < Min δε χρειάζεται παρά να μετακινηθεί στη θέση Max = 7 και το στοιχείο που βρισκόταν στη θέση Max να μετακινηθεί στην τελευταία θέση της δομής = 13. Έτσι διατηρείται η δομή του κυκλικού πίνακα. Γενικότερα κατά την εισαγωγή ή κατά τη διαγραφή ενός στοιχείου σε αυτή τη δομή απαιτείται χρόνος $O(N^{1/2} log_2 N)$. Εάν δε χρησιμοποιηθεί ένας κυκλικός πίνακας αλλά περισσότεροι κυκλικοί υποπίνακες,

τότε εάν ο αριθμός των κυκλικών υποπινάκων είναι k απαιτείται χρόνος $O(N^{1/k} log_2 \mathrm{N})$ κατά την εισαγωγή ή τη διαγραφή ενός στοιχείου. Τα παραπάνω αναφέρονται στην εργασία του Frederickson (1983) σχετικά με τις τεχνικές αφανούς δόμησης και πιο συγκεκριμένα περί περιστρεφόμενων πινάκων σε πολλά επίπεδα.

Ακόμα όμως και με τις παραπάνω βελτιώσεις η δομή αυτή δεν είναι αρκετά αποδοτική κατά την υλοποίηση του αλγορίθμου Dijkstra διότι απαιτεί μία ισχυρή ταξινόμηση των στοιχείων ενώ κατά την επανάληψη του αλγορίθμου είναι χρήσιμη μόνο η εύρεση του κόμβου με το μικρότερο κόστος.

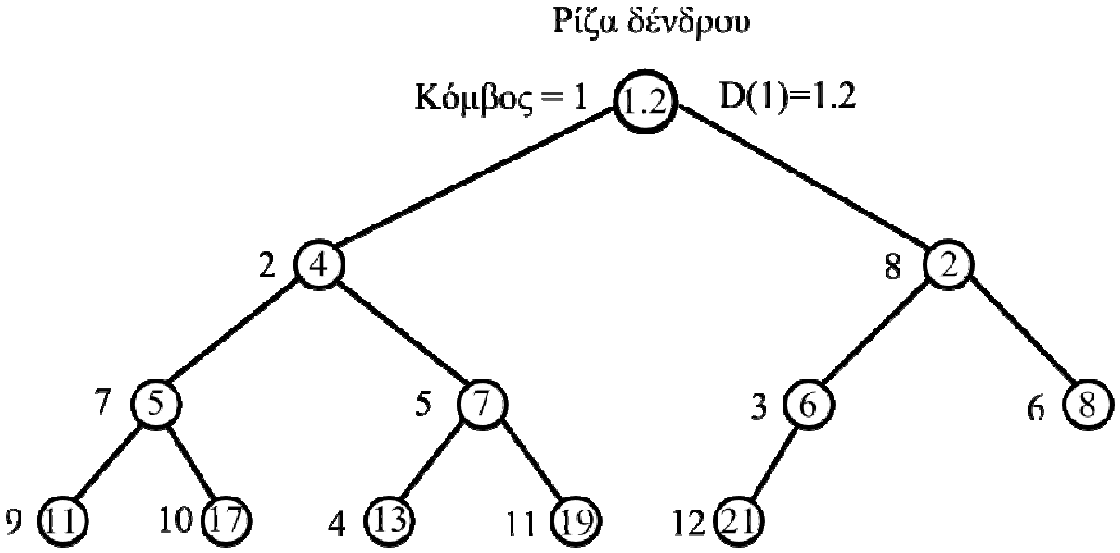
5.2.3.2.Δυαδικός Σωρός

Μία άλλη δομή δεδομένων είναι ο δυαδικός σωρός (Binary Heap). Για να υλοποιηθεί αυτή η δομή δεδομένων πρέπει να πληρούνται οι απαραίτητες συνθήκες :

α) Οι κόμβοι του δικτύου πρέπει να είναι αποθηκευμένοι στις κατάλληλες θέσεις με βάση το κόστος D(i). Στη θέση 1 (ρίζα του δυαδικού δένδρου) βρίσκεται ο κόμβος με τη μικρότερη τιμή D(i). Επίσης κάθε κόμβος που είναι αποθηκευμένος στη θέση j της μήτρας έχει δύο απογόνους, οι οποίοι θα πρέπει να είναι αποθηκευμένοι στις θέσεις $2 \times$ j, $2 \times$ j+1 της μήτρας. Απογόνους μπορεί να μην έχουν μόνο οι κόμβοι που είναι αποθηκευμένοι στα δύο τελευταία επίπεδα του δυαδικού δένδρου. Η δομή του δυαδικού δένδρου και των συνθηκών που το μετατρέπουν σε δυαδικό σωρό παρουσιάζονται στο Σχήμα 5.2.

β) Κάθε στοιχείο του δυαδικού σωρού έχει τιμή D(i) μικρότερη ή ίση από την τιμή των απογόνων του. Αυτή η μορφή του δυαδικού σωρού καλείται και δυαδικός σωρός με φθίνουσα σειρά κόστους προς τον πατρικό κόμβο. Επίσης δεν υπάρχει κανένας περιορισμός για τα στοιχεία που βρίσκονται στο ίδιο επίπεδο και ονομάζονται συγγενικά.

Στο ακόλουθο σχήμα παρουσιάζεται η αποτύπωση του παραπάνω παραδείγματος σε μορφή δυαδικού σωρού :



Σχήμα 5.2.Μορφή Δυαδικού Σωρού όπου οι κόμβοι ταξινομούνται με φθίνουσα σειρά κόστους προς τον πατρικό κόμβο

Ο δυαδικός σωρός αναπαριστάται μέσω μονοδιάστατων πινάκων ως εξής :

Θέση	1	2	3	4	5	6	7	8	9	10	11	12
Κόμβος	1	2	8	7	5	3	6	9	10	4	11	12
D(i)	1.2	4	2	5	7	6	8	11	17	13	19	21

Όπως αναφέρθηκε ήδη, εάν ένα στοιχείο βρίσκεται στη θέση j οι απόγονοί του θα βρίσκονται στις θέσεις 2j, 2j+1. Για παράδειγμα ο κόμβος 7 βρίσκεται στη θέση 4 και οι απόγονοί του 9, 10 βρίσκονται στις θέσεις 8, 9 αντίστοιχα.

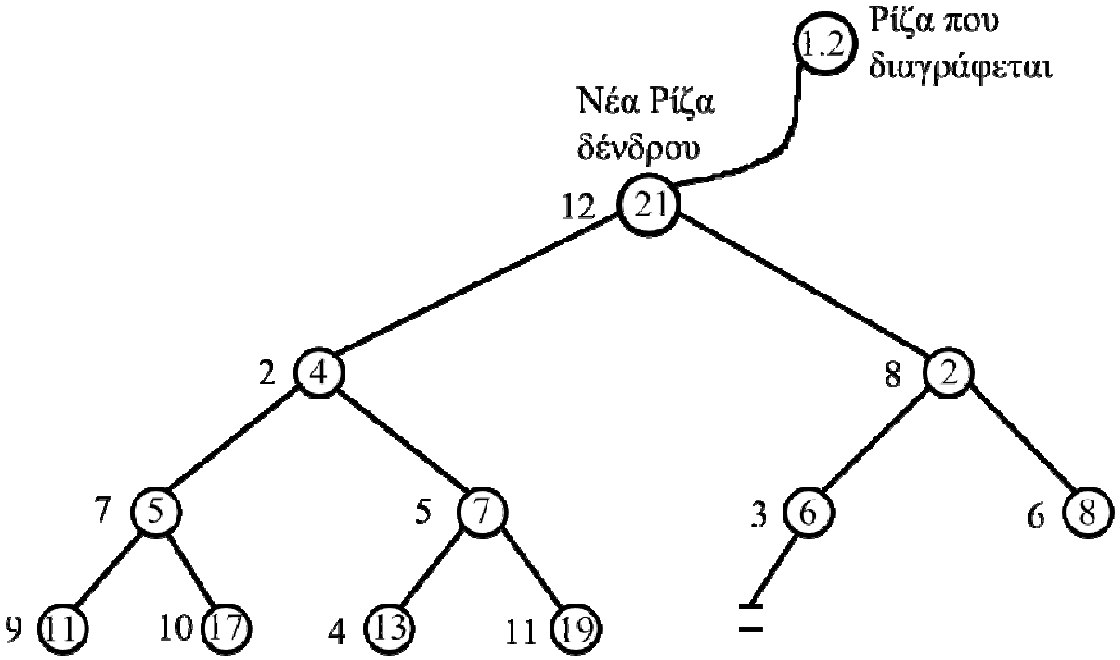
Σύμφωνα με την παραπάνω δομή δεδομένων ο κόμβος που βρίσκεται στη θέση 1 (ρίζα) είναι ο κόμβος με την ελάχιστη τιμή κόστους, άρα θα επιλεγεί κατά την τρέχουσα επανάληψη του αλγορίθμου Dijkstra σε χρόνο $O(1)$. Ωστόσο απαιτείται ένας αριθμός βασικών πράξεων ώστε να ισχύουν οι συνθήκες του δυαδικού σωρού στις περιπτώσεις :

α) Εισαγωγή νέου στοιχείου ή αλλαγή του κόστους στοιχείου που βρίσκεται ήδη στο σωρό. Έστω για παράδειγμα ότι στο δυαδικό σωρό του σχήματος 5.2. εισάγεται ο κόμβος 13 με κόστος D(13) = 14. Ο κόμβος αυτός αποθηκεύεται στην τελευταία θέση του σωρού, δηλαδή στη θέση 13. Στη συνέχεια το κόστος του : D(13) συγκρίνεται με το κόστος του προγόνου του ο οποίος βρίσκεται στη θέση A = (Θέση Κόμβου 13) / 2, όπου $A \epsilon Z^+$. Εάν D(13) < D(A), τότε ο κόμβος 13 μετακινείται στη θέση A = 6 και ο κόμβος 3 που βρίσκεται στη θέση 6 μετακινείται στη θέση 13 ώστε να ισχύει η συνθήκη ότι οι απόγονοι ενός στοιχείου έχουν μεγαλύτερο κόστος. Η διαδικασία αυτή επαναλαμβάνεται και ο κόμβος 13 μπορεί να φθάσει μέχρι και τη ρίζα του δένδρου εάν D(13) < D(1). Έτσι οι συγκρίσεις που μπορεί να γίνουν κατά την εισαγωγή ενός στοιχείου στο δίκτυο δεν είναι περισσότερες από τον αριθμό των επιπέδων του δυαδικού δένδρου, ο οποίος είναι προφανώς ίσος με $log_2 N$. Σύμφωνα

με τα παραπάνω για τη δημιουργία ενός δυαδικού σωρού που αποτελείται από N στοιχεία απαιτούνται N$log_2 N$ βασικές πράξεις.

Στην περίπτωση που δεν εισάγεται ένας νέος κόμβος στο σωρό αλλά αναπροσαρμόζεται το κόστος D(i) ενός κόμβου i που βρίσκεται στη θέση K του σωρού σε D'(i), τότε συγκρίνεται το κόστος D'(i) με το κόστος του κόμβου που βρίσκεται στη θέση A = K/2 και εάν είναι μικρότερο ανταλλάσσουν θέσεις και επαναλαμβάνεται η διαδικασία το πολύ $log_2 N$ φορές.

β) Διαγραφή στοιχείου. Στο τέλος κάθε επανάληψης του αλγορίθμου Dijktra απαιτείται η διαγραφή του κόμβου που βρίσκεται στη ρίζα του δυαδικού σωρού καθώς αυτός ο κόμβος είναι ο επόμενος πλησιέστερος κόμβος στον αρχικό. Στο παραπάνω παράδειγμα ο κόμβος αυτός είναι ο κόμβος 1. Στη θέση του στο δυαδικό δένδρο μετακινείται ο κόμβος που βρίσκεται στην τελευταία θέση του δένδρου, όπως παρουσιάζεται και σχηματικά :



Σχήμα 5.3.Διαγραφή ρίζας δυαδικού σωρού και μετακίνηση του κόμβου που βρισκόταν στην τελευταία θέση στη θέση της

Όπως φαίνεται και από το παραπάνω σχήμα δεν πληρούνται πλέον οι συνθήκες του δυαδικού σωρού. Για το λόγο αυτό συγκρίνεται το κόστος του κόμβου 12 με το κόστος των απογόνων του 2, 8 και μετακινείται στη θέση του ο κόμβος με το μικρότερο κόστος, δηλαδή ο κόμβος 8. Η διαδικασία αυτή επαναλαμβάνεται για τους νέους απογόνους του κόμβου 12 που είναι πλέον οι 3, 6 κ.ο.κ. οπότε απαιτούνται συνολικά $log_2 N$ συγκρίσεις.

Ο χρόνος εκτέλεσης του αλγορίθμου Dijkstra με χρήση δυαδικού σωρού είναι O((N+E) logN). Στη συνέχεια παρουσιάζεται η μορφή του.

ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA με χρήση Δυαδικού Σωρού

Βήμα 1(Αρχικοποιήσεις)

Για Κάθε i = 1,N
D (i) ← +∞
Q (i) ← 0 // Ο κόμβος i βρίσκεται στη θέση Q(i) του σωρού
H (i) ← 0 // Στη θέση i του σωρού βρίσκεται ο κόμβος H(i)
Τέλος Για Κάθε
D (r) ← 0
u ← r

Βήμα 2(Αλγόριθμος Dijkstra, Υλοποίηση με χρήση Δυαδικού Σωρού)

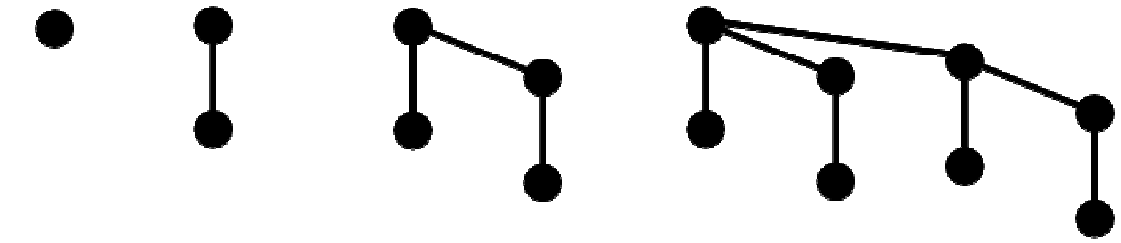
M ← 0 // Αριθμός Κόμβων που βρίσκονται στο δυαδικό σωρό
Για Κάθε i = 1,N
Για Κάθε e = 1,E
Εάν S(e) = u & D(E(e)) > D(S(e)) + w(e)
D(E(e)) ← D(S(e)) + w(e)
Εάν Q(E(e)) ≠ 0 // ο κόμβος E(e) υπάρχει ήδη στο σωρό και βρίσκεται
//στη θέση Q(E(e))
m ← Q(E(e))
Αρχή 1
Εάν D(H(m)) < D(H(m/2)) // Εάν το κόστος του κόμβου E(e) που
// βρίσκεται στη θέση m είναι μικρότερο από το κόστος του προγόνου του που
// βρίσκεται στη θέση m/2, τότε ανταλλάσσουν θέσεις στο σωρό
Z ← H(m) & H(m) ← H(m/2) & H(m/2) ← Z
Y ← Q(H(m)) & Q(H(m)) ← Q(H(m/2)) & Q(H(m/2)) ← Y
m ← m/2
Επέστρεψε στην Αρχή 1
Τέλος Εάν
Τέλος Εάν
Εάν Q(E(e)) ← 0 // ο κόμβος E(e) δεν υπήρχε στο σωρό και εισάγεται
// στην τελευταία θέση του
M ← M+1
H(M) ← E(e) // Στη θέση M του σωρού βρίσκεται ο κόμβος E(e)
Q(E(e)) ← M // ο κόμβος E(e) τοποθετείται στη θέση M
m ← M
Αρχή 2
Εάν D(H(m)) < D(H(m/2))
Z ← H(m) & H(m) ← H(m/2) & H(m/2) ← Z
Y ← Q(H(m)) & Q(H(m)) ← Q(H(m/2)) & Q(H(m/2)) ← Y
m ← m/2
Επέστρεψε στην Αρχή 2
Τέλος Εάν

```
            Τέλος Εάν
  Τέλος Εάν
Τέλος Για Κάθε
// Διαδικασία διαγραφής Ρίζας Κατά την τρέχουσα Επανάληψη
u ← H(1) // ο επόμενος κόμβος οδηγός, είναι η ρίζα του δένδρου
Q( H(M) ) ← 1
H( 1 ) ← H( M ) // ο τελευταίος κόμβος του δένδρου γίνεται ρίζα του
M ← M – 1 // το δένδρο μικραίνει κατά ένα στοιχείο
H( M ) ← 0 & D( H(M) ) ← +∞ // ο τελευταίος κόμβος του δένδρου αφαιρείται
m ← 1
  Αρχή 2
  Εάν D( H(2 × m) ) < D( H(2 × m+1) )
       Εάν D(H(m)) > D( H(2 × m) )
       Z ← H(m) & H(m) ← H(2 × m) & H(2 × m) ← Z
       Y ← Q( H(m) ) & Q( H(m) ) ← Q( H(2 × m) ) & Q( H(2 × m) ) ← Y
       m ← m × 2
       Επέστρεψε στην Αρχή 2
       Τέλος Εάν
  Τέλος Εάν
  Εάν D( H(2 × m) ) > D( H(2 × m+1) )
       Εάν D(H(m)) > D( H(2 × m+1) )
       Z ← H(m) & H(m) ← H(2 × m+1) & H(2 × m+1) ← Z
       Y ← Q( H(m) ) & Q( H(m) ) ← Q( H(2 × m+1) ) & Q( H(2 × m+1) ) ← Y
       m ← m × 2 + 1
       Επέστρεψε στην Αρχή 2
       Τέλος Εάν
   Τέλος Εάν
Τέλος Για Κάθε
```

5.2.3.3.Σωρός Fibonacci

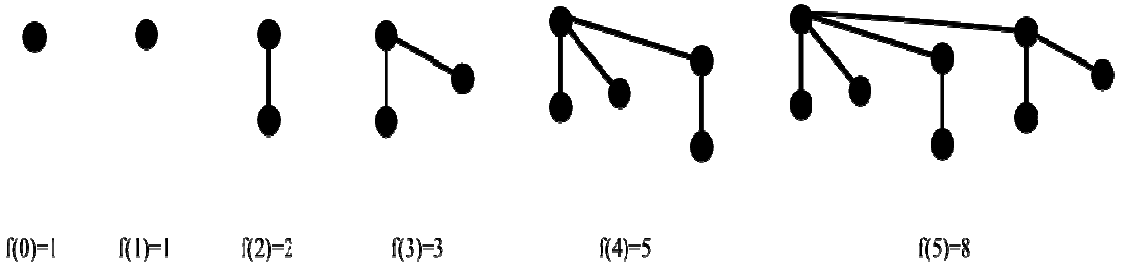
Μία ακόμη δομή δεδομένων είναι ο σωρός Fibonacci. Αυτή η δομή δεδομένων προτάθηκε στην εργασία των Fredman and Tarjan (1987). Μέσω αυτής της δομής μειώνεται ο χρόνος εκτέλεσης του αλγορίθμου Dijkstra από $O((N+E)log_2N)$ που απαιτείται με χρήση του δυαδικού σωρού σε $O(E+Nlog_2N)$.

Ο σωρός Fibonacci έχει κοινά στοιχεία τουλάχιστον ως προς την ιδέα υλοποίησης με το δυωνυμικό σωρό (Binomial Heap), στον οποίο κάθε δένδρο ενώνεται αναδρομικά με ένα δένδρο ίσου μεγέθους.



Σχήμα 5.4.Τα τέσσερα πρώτα βασικά Δυωνυμικά Δένδρα

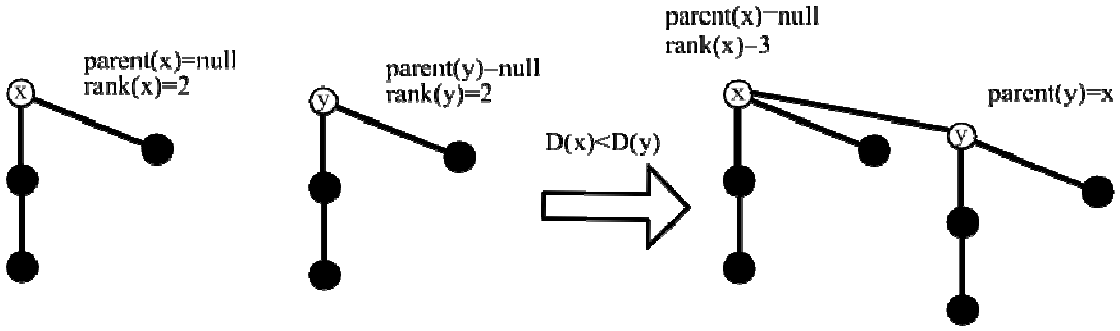
Έτσι κάθε δυωνυμικό δένδρο έχει στοιχεία σύμφωνα με την αναδρομική σχέση $f(k) = 2 \times f(k-1)$. Ο σωρός Fibonacci αποτελεί μία διαφορετική δομή δεδομένων και στη ουσία είναι ένα σύνολο από δένδρα, τα οποία ικανοποιούν την ιδιότητα της ταξινόμησης όπου στοιχείο-πρόγονος έχει μικρότερο κόστος από τον απόγονό του. Έτσι το μικρότερο κόστος το έχει πάντα η ρίζα του δένδρου. Αναλύοντάς τον περισσότερο παρουσιάζονται στη συνέχεια τα ελάχιστα δένδρα Fibonacci, όπου κάθε ελάχιστο δένδρο k έχει στοιχεία $f(k) = f(k-1) + f(k-2)$. Έτσι το μέγεθος των ελαχίστων δένδρων ισούται με την ακολουθία Fibonacci : 1, 2, 3, 5, 8, 13, 21, ……



Σχήμα 5.5.Ελάχιστα δένδρα Fibonacci

Η ουρά προτεραιότητας Fibonacci δημιουργείται από μία σειρά δένδρων Fibonacci των οποίων οι ρίζες βρίσκονται στο πρώτο επίπεδο και καμία ρίζα δεν έχει τους ίδιους απογόνους με κάποια άλλη ρίζα, αλλιώς τα δένδρα αυτά ενώνονται. Από όλα τα δένδρα των οποίων οι ρίζες βρίσκονται στο 1^o επίπεδο προσδιορίζεται και διατηρείται σε μία θέση μνήμης η ρίζα του δένδρου με το μικρότερο κόστος. Για τα επιμέρους στοιχεία κάθε δένδρου ισχύουν οι συνθήκες των δυαδικών δένδρων όπου κάθε πρόγονος έχει μικρότερο κόστος από τους απογόνους του. Ο αριθμός των απογόνων κάθε στοιχείου x μπορεί να δίνεται μέσα από μία μήτρα rank(x) και ο πρόγονος κάθε στοιχείου x μπορεί να είναι αποθηκευμένος σε μία μήτρα parent(x). Είναι προφανές ότι τα στοιχεία που βρίσκονται στο 1^o επίπεδο (ρίζες δένδρων) δεν έχουν προγόνους και για αυτά parent(x) = null. Επίσης για κάθε στοιχείο αποθηκεύεται σε μία μήτρα ο πρώτος απόγονός του child(x) και για κάθε στοιχείο το επόμενο συγγενικό του στοιχείο next(x), δηλαδή το επόμενο στοιχείο που έχει τον ίδιο πρόγονο με αυτό. Μέχρι τώρα υπάρχει ανάγκη για τη χρήση τριών μονοδιάστατων πινάκων. Εάν δύο ρίζες δένδρων x, y έχουν τον ίδιο αριθμό απογόνων rank(x) = rank(y), τότε τα δένδρα που αυτά ενώνονται σε χρόνο $O(1)$ συγκρίνοντας

τις τιμές κόστους των x, y και προσθέτοντας το δένδρο με το μεγαλύτερο κόστος στο δένδρο με το μικρότερο κόστος όπως φαίνεται στο ακόλουθο σχήμα.



Σχήμα 5.6.Ένωση Δένδρων όταν οι ρίζες έχουν ίδιο αριθμό απογόνων

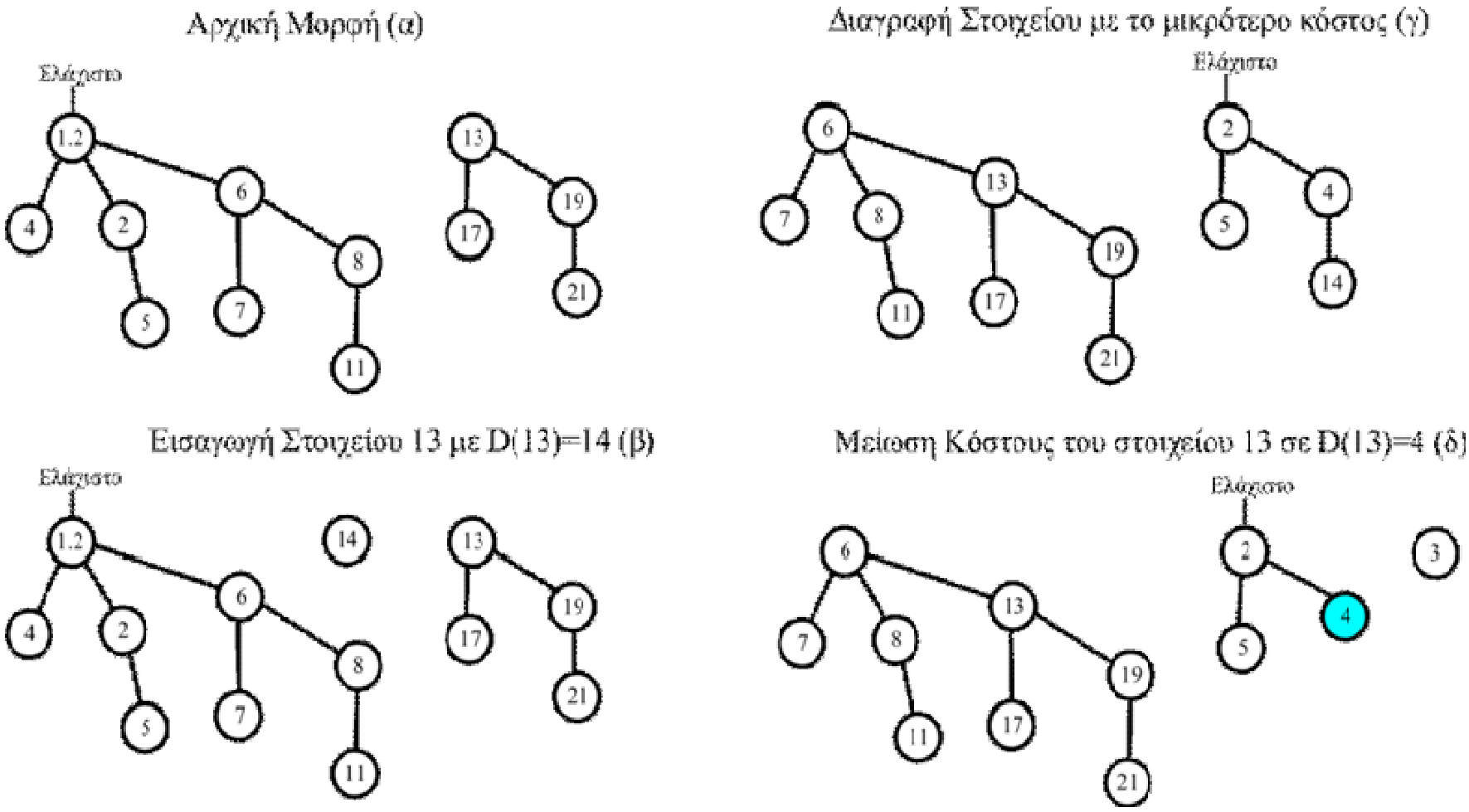
α) Εισαγωγή στοιχείου στο σωρό. Κατά την εισαγωγή ενός στοιχείου στο σωρό απαιτείται χρόνος $O(1)$. Το στοιχείο αυτό, έστω j, εισάγεται στο $1^{ο}$ επίπεδο και αποτελεί από μόνο του ένα δένδρο Fibonacci. Το στοιχείο αυτό αποτελεί μία ρίζα χωρίς απογόνους : rank(j) = 0 και εάν υπάρχει και άλλη ρίζα στο πρώτο επίπεδο k η οποία έχει τον ίδιο αριθμό απογόνων rank(k) = 0, τότε τα δένδρα αυτά ενώνονται και η ρίζα του νέου δένδρου είναι το στοιχείο με το μικρότερο κόστος. Το νέο δένδρο που προέκυψε έχει έναν απόγονο και για το λόγο αυτό ελέγχεται εάν υπάρχει και άλλο δένδρο στο οποίο η ρίζα του έχει μόνο έναν απόγονο ώστε να ενωθούν μεταξύ τους και η διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχουν ρίζες με τον ίδιο αριθμό απογόνων στα νέα δένδρα που προκύπτουν. Για την εισαγωγή ενός στοιχείου απαιτείται ο ίδιος χρόνος που απαιτείται και για την ένωση, καθώς επί της ουσίας πρόκειται για μία διαδικασία ένωσης δένδρων.

β) Διαγραφή στοιχείου από το σωρό. Σε κάθε επανάληψη του αλγορίθμου Dijkstra απαιτείται η διαγραφή του στοιχείου με το μικρότερο κόστος. Το στοιχείο αυτό είναι ρίζα ενός δένδρου h. Στη συνέχεια όλοι οι απόγονοι αυτής της ρίζας μεταφέρονται στο $1^{ο}$ επίπεδο και αποτελούν ρίζες νέων δένδρων. Έτσι από τη διαγραφή ενός στοιχείου με d απογόνους προκύπτουν αρχικά d-1 νέα δένδρα. Στη συνέχεια ελέγχονται όλα τα δένδρα που έχουν προκύψει και όποια δένδρα έχουν ρίζες με ίδιο αριθμό απογόνων ενώνονται μεταξύ τους. Τέλος συγκρίνεται το κόστος που έχουν οι ρίζες των νέων δένδρων και εντοπίζεται η νέα ρίζα με το ελάχιστο κόστος. Ο απαιτούμενος χρόνος για τη διαγραφή ενός στοιχείου είναι μεγαλύτερος σε σύγκριση με τις άλλες λειτουργίες και είναι $log_2 N$.

γ) Μείωση στο κόστος ενός στοιχείου i που ανήκει σε ένα δένδρο h. Εάν το νέο κόστος του στοιχείου αυτού είναι μικρότερο από το κόστος του πατρικού του στοιχείου : D(i) < D(p(i)), τότε το στοιχείο αυτό πρέπει να προβιβαστεί. Το στοιχείο αυτό δεν προβιβάζεται στο δένδρο στο οποίο ανήκει αλλά αποκόπτεται μαζί με τα στοιχεία που αποτελούν τους απογόνους του και δημιουργείται ένα νέο δένδρο. Ταυτόχρονα ο πρόγονος αυτού του στοιχείου, έστω y, εντοπίζεται και 'μαρκάρεται'. Αυτό μπορεί να γίνει μέσω μίας μήτρας mark(y) όπου κάθε στοιχείο της μήτρας αυτής είτε έχει μαρκαριστεί mark(j) = ναι, είτε δεν έχει μαρκαριστεί mark(j) = όχι.

Έτσι, πλέον ο πρόγονος του στοιχείου i που αποκόπηκε από το δένδρο είτε μαρκάρεται : mark(y) = ναι, είτε είναι ρίζα του δένδρου η οποία δε μπορεί να μαρκαριστεί, είτε είναι μαρκαρισμένος ήδη, οπότε αποκόπτεται και ο κόμβος αυτός και δημιουργεί ένα νέο δένδρο μόνος του και μαρκάρεται ο πρόγονός του. Κατά την τελευταία περίπτωση εάν ο πρόγονος και αυτού του στοιχείου είναι επίσης μαρκαρισμένος, τότε αποκόπτεται και το στοιχείο αυτό, δημιουργεί νέο δένδρο, μαρκάρεται ο πρόγονός του και η διαδικασία συνεχίζεται μέχρι να βρεθεί κάποιος πρόγονος που δεν είναι μαρκαρισμένος ή μέχρι ο πρόγονος να είναι η ρίζα του δένδρου η οποία εξ'ορισμού δε μπορεί να είναι μαρκαρισμένη.

Στη συνέχεια παρουσιάζονται σχηματικά οι βασικές λειτουργίες (εισαγωγή στοιχείου, διαγραφή στοιχείου, αλλαγή κόστους στοιχείου) στο σωρό Fibonacci για το παράδειγμα που εξετάστηκε ήδη και από τις προηγούμενες δομές δεδομένων.



Σχήμα 5.7.Εισαγωγή, Διαγραφή και μείωση κόστους στοιχείου σε σωρό Fibonacci

Συμπεραματικά οι χρόνοι εκτέλεσης του αλγορίθμου Dijkstra με χρήση των δομών δεδομένων που παρουσιάστηκαν μέχρι τώρα είναι οι ακόλουθοι :

Διαδικασίες	Λίστα αναμονής	Δυαδικός Σωρός (Binary Heap)	Σωρός Fibonacci
Εισαγωγή στοιχείου	1	log N	1
Διαγραφή στοιχείου u για το οποίο D (u)=MIN.	N	log N	log N
Αλλαγή της τιμής D (i) ενός στοιχείου	1	log N	1
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ	$O(N^2)$	*O((N+E) logN)*	*O(N+E log N)*

Πίνακας 5.1.Χρόνος εκτέλεσης του αλγορίθμου Dijkstra για διαφορετικές δομές δεδομένων (data structures)

Από την αρχική διατύπωση του αλγορίθμου Dijkstra μέχρι σήμερα έχουν προταθεί πολλές βελτιώσεις. Αυτό είναι ενδεικτικό του ερευνητικού ενδιαφέροντός του καθώς ο αλγόριθμος αυτός χρησιμοποιείται ως υπορουτίνα σε πολλά προβλήματα. Ορισμένες από αυτές τις βελτιώσεις παρατίθενται στον παρακάτω πίνακα όπου με n συμβολίζεται ο αριθμός των κόμβων, m ο αριθμός των συνδέσμων και W το βάρος του μεγαλύτερου συνδέσμου :

Χρόνος Εκτέλεσης	Εργασία
O(m lg lg W)	Donald B. Johnson -A priority queue in which initialization and queue operations take *O*(log log*D*) time, 1982
O(m lg n)	Peter M. Winkler –Proof of the Squashed Cube Conjecture, 1983
O(m+n lg n)	Michael Fredman and Roben Tarjan -Fibonacci Heaps and their uses in improved network optimization algorithms, 1987
$O(m+n\sqrt{\lg W})$	Ravindra K. Ahuja, Kurt Mehlhorn, James B. Orlin, and Robert Tarjan -Faster algorithms for the shortest path problem, 1990
$O(m\sqrt{\lg n})$	Michael L. Fredman and Dan E. Willard - Surpassing the information theoretic bound with fusion trees, 1993
$O(m+n\frac{\lg n}{\lg\lg n})$	Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths, 1994
O(m lg lg n)	Mikkel Thorup - On RAM priority queues, 1996
$O(m+nlg^{\frac{1}{2}+e}n)$	Mikkel Thorup - On RAM priority queues, 1996
$O(m+n\sqrt{\lg n \lg\lg n})$	Rajeev Raman -Priority queues: Small, monotone and trans-dichotomous, 1996
$O(m+nlg^{\frac{1}{3}+e}n)$	Rajeev Raman -Recent results on the single-source shortest paths problem, 1997
$O(m+n\sqrt{\lg\lg n})$	Yijie Han and Mikkel Thorup -Integer sorting in $O(n\sqrt{loglogn})$ expected time and linear space, 2002
O(m+n lg lg n)	Mikkel Thorup - Integer priority queues with decrease key in constant time and the single source shortest paths problem, 2003

Πίνακας 5.2.Χρόνοι εκτέλεσης βελτιωμένων εκδόσεων αλγορίθμου Dijkstra

5.2.4.ΑΛΓΟΡΙΘΜΟΙ ΤΗΣ ΚΑΤΗΓΟΡΙΑΣ “LABEL CORRECTING”

Μέχρι τώρα όλοι οι αλγόριθμοι που έχουν αναλυθεί ανήκουν στην κατηγορία των αλγορίθμων για την εύρεση ελάχιστων διαδρομών από έναν κόμβο του δικτύου προς όλους τους υπόλοιπους κόμβους του δικτύου (Single Source Shortest Path Problems). Η γενικότερη μορφή όλων των μεθόδων που χρησιμοποιούνται στα προβλήματα αυτά δόθηκε από την εργασία των Gallo and Pallottino (1986). Σύμφωνα με την εργασία αυτή η τυπική επανάληψη είναι :

Όσο ($Q \neq Ø$){
Q = Q - {i} : Αφαίρεσε έναν κόμβο i από τη λίστα Q.
Για Κάθε e = 1,E
  Εάν S(e) = i & D(E(e)) > D(S(e)) + w(e)
  D(E(e)) ← D(S(e)) + w(e)
  Τέλος Εάν
Q = Q + {e} : Πρόσθεσε τον κόμβο e στη λίστα Q εάν δεν ανήκει ήδη σε αυτήν.
Τέλος Όσο

Οι διαφορετικοί αλγόριθμοι ξεχωρίζουν από τον τρόπο με τον οποίο επιλέγουν τον κόμβο που αφαιρείται από τη λίστα Q. Έτσι στον αλγόριθμο Dijkstra ο κόμβος που αφαιρείται από τη λίστα σε κάθε βήμα είναι αυτός με τη μικρότερη τιμή D(u) σε σχέση με τους υπόλοιπους κόμβους. Οι αλγόριθμοι που ακολουθούν αυτή τη λογική ανήκουν στην κατηγορία “label setting”, καθώς κάθε κόμβος που αφαιρείται δε μπορεί να αλλάξει τιμή ξανά. Οι υπόλοιπες μέθοδοι που δεν ακολουθούν αυτήν τη διαδικασία ονομάζονται “label correcting” και σε αυτές τις μεθόδους η επιλογή του κόμβου που θα αφαιρεθεί από τη λίστα Q είναι ταχύτερη καθώς δε χρειάζεται να βρεθεί ο κόμβος με τη μικρότερη τιμή. Παράδειγμα τέτοιων μεθόδων είναι ο αλγόριθμος Moore-Bellman-Ford.

Οι “label correcting” αλγόριθμοι ανήκουν στην κατηγορία του δυναμικού προγραμματισμού και όχι των άπληστων αλγορίθμων αφού δεν αναζητούν συνεχώς το τοπικό βέλτιστο. Χρησιμοποιούν μία λίστα Q παράλληλα με την υποψήφια λίστα V. Οι κόμβοι μπορούν να εισέλθουν στη λίστα Q και να εξέλθουν από αυτήν σε χρόνο *O(1)* χρησιμοποιώντας διαφορετικές δομές δεδομένων για κάθε αλγόριθμο που εξετάζεται και ανήκει σε αυτές τις μεθόδους. Οι αλγόριθμοι που έχουν αναπτυχθεί και ανήκουν σε αυτή την κατηγορία μεθόδων διαφέρουν από τον τρόπο που επιλέγεται η θέση στη λίστα Q στην οποία εισέρχεται ένας νέος κόμβος όταν εισέλθει στο σύνολο V. Μερικά παραδείγματα είναι :

1)Αλγόριθμος Moore : Σε μία παραλλαγή της πρωτότυπης διατύπωσής του θεωρείται ότι οι κόμβοι που εισέρχονται στο σύνολο V προστίθενται στο τέλος της λίστας Q και αφαιρούνται πάντα από την αρχή της ίδιας λίστας. Έτσι οι κόμβοι που εισέρχονται και εξέρχονται από αυτό ακολουθούν τη διαδικασία (first in-first out). Αυτή η μέθοδος απαιτεί $O(N \times E)$ βασικές πράξεις όπως έχει αναλυθεί ήδη. Εδώ πρέπει να

σημειωθεί ότι παρά το μεγαλύτερο αριθμό των επαναλήψεων που απαιτούνται σε σχέση με τον αλγόριθμο Dijkstra η μέθοδος αυτή μπορεί να είναι ταχύτερη γιατί δε ζητείται σε κάθε επανάληψη η εύρεση του κόμβου με την ελάχιστη τιμή D(u) που απαιτεί πολλές επιπλέον συγκρίσεις που επιβαρύνουν τον τελικό χρόνο εκτέλεσης. Αυτό επισημαίνεται και στην εργασία του Golden B. (1976).

2)Αλγόριθμος D'Esopo, Pape : Αυτή η μέθοδος προτάθηκε πρώτη φορά στην εργασία του Pape (1974). Κατά την εφαρμογή της κάθε κόμβος που εισέρχεται στο σύνολο V για πρώτη φορά τοποθετείται στο τέλος της λίστας Q και εάν ξαναεισέλθει στο σύνολο V τοποθετείται στην αρχή της. Απαιτούνται πολλές επαναλήψεις για να ολοκληρωθεί αυτή η διαδικασία που στη χειρότερη περίπτωση μπορεί να φτάσει και τις 2^N (εκθετικός χρόνος) την ώρα που οι επαναλήψεις για τον αλγόριθμο Moore δεν ξεπερνούν τις N^2. Στην πράξη όμως παρατηρείται σημαντική διαφοροποίηση και ο αλγόριθμος αυτός μπορεί να είναι αρκετά ταχύτερος σε αραιά δίκτυα από τον αντίστοιχο του Moore ενώ δε διαφέρει σημαντικά σε χρόνο εκτέλεσης ούτε από τις ταχύτερες μεθόδους της κατηγορίας "label setting". Αξίζει να σημειωθεί ότι δεν υπάρχει συγκεκριμένη εξήγηση για το λόγο που ισχύει αυτό.

3)Αλγόριθμος Gallo, Pallottino : Αυτή η μέθοδος προτάθηκε πρώτη φορά στις εργασίες των Gallo and Pallottino (1986) και Gallo and Pallottino (1988). Στη μέθοδο αυτή που αποτελεί βελτίωση της μεθόδου D'Esopo-Pape η λίστα Q χωρίζεται σε δύο λίστες Q1 και Q2 όπου κάθε κόμβος που εισέρχεται για πρώτη φορά στο σύνολο V τοποθετείται στο τέλος της λίστας Q2 και αν εισέλθει ξανά στο σύνολο V τοποθετείται στο τέλος της λίστας Q1. Επίσης κάθε στοιχείο που εξέρχεται από το σύνολο V αφαιρείται από την αρχή της λίστας Q1 εφόσον αυτή δεν είναι κενή, αλλιώς αφαιρείται από την αρχή της λίστας Q2. Οι βασικές πράξεις που αναμένεται να εκτελεστούν κατά την εφαρμογή του αλγορίθμου δίνουν χρόνο υλοποίησης $O(N^2 \times E)$.

3)Αλγόριθμος Threshold : Προτάθηκε από τους Glover et al. (1986). Στη μέθοδο αυτή η λίστα Q όπως και πριν χωρίζεται σε δύο λίστες Q1 και Q2. Σε κάθε επανάληψη ο κόμβος που αφαιρείται από το σύνολο V είναι αυτός που βρίσκεται στην αρχή της λίστας Q1 και ο κόμβος που εισέρχεται στο σύνολο V τοποθετείται στο τέλος της λίστας Q2 ή στο τέλος της λίστας Q1 ανάλογα με το αν η τιμή του κόμβου D(u) παραβιάζει την παράμετρο "threshold". Έτσι προκύπτει και η ονομασία του αλγορίθμου. Εάν λοιπόν ο κόμβος που εισέρχεται έχει ετικέτα με τιμή μεγαλύτερη της παραμέτρου "threshlold" τοποθετείται στο τέλος της λίστας Q2, αλλιώς τοποθετείται στο τέλος της λίστας Q1. Όταν η λίστα Q1 αδειάσει η τιμή της παραμέτρου "threshold" αυξάνεται και οι κόμβοι στη λίστα Q2 των οποίων οι τιμές παραβιάζουν τη νέα τιμή της παραμέτρου αφαιρούνται από αυτήν και τοποθετούνται στη λίστα Q1. Στην πιο απλή περίπτωση όπου κάθε κόμβος που εισέρχεται στο V τοποθετείται πάντα στο τέλος του Q2 ανεξάρτητα από το αν η τιμή του παραβιάζει την παραπάνω παράμετρο ο αλγόριθμος απαιτεί χρόνο υλοποίησης $O(N^2 \times E)$ και N^3 επαναλήψεις.

5.2.5.Ο ΑΛΓΟΡΙΘΜΟΣ BELLMAN-KALABA

Ο αλγόριθμος αυτός προτάθηκε από τους Bellman and Kalaba (1960). Ο αλγόριθμος χρησιμοποιείται για την εύρεση των συντομότερων διαδρομών από όλους τους κόμβους του δικτύου προς τον κόμβο προορισμού s. Επιλύει δηλαδή το αντίστροφο πρόβλημα από αυτό που εξεταζόταν μέχρι τώρα.
Η μορφή του αλγορίθμου είναι :

ΑΛΓΟΡΙΘΜΟΣ BELLMAN - KALABA

Βήμα 1(Αρχικοποιήσεις)

Για Κάθε i=1,N
D(i) ← w(i)(s)
u(i) ← s
Τέλος Για Κάθε

Βήμα 2(Αλγόριθμος Bellman-Kalaba)

Για Κάθε k = 1,N
A ← 0
  Για Κάθε i=1,N
  B ← D(i)
    Για Κάθε j=1,N
      Εάν w(i)(j) + D (j) < D(i)
      D(i) ← w(i)(j) + D(j)
      u(i) ← j
      Τέλος Εάν
    Τέλος Για Κάθε
    Εάν B = D(i)
    A ← A + 1
    Τέλος Εάν
  Τέλος Για Κάθε
 Εάν A = N {Βγες απ'την επανάληψη και συνέχισε παρακάτω}
Τέλος Για Κάθε
k ← 1 // Αποθήκευση κόμβων συντομότερης διαδρομής
J ← r
p(k) ← r
Όσο J ≠ s
k ← k+1
p(k) ← u(J)
J ←u(J)
Τέλος Όσο

Ο χρόνος εκτέλεσής του είναι στη χειρότερη περίπτωση $O(N^3)$, δηλαδή πολυωνυμικός. Για το ίδιο πρόβλημα μπορεί να χρησιμοποιηθεί και ο αλγόριθμος Dijkstra όπως παρουσιάστηκε ήδη με μικρές τροποποιήσεις.

5.3.ΔΕΝΔΡΑ ΕΛΑΧΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ

Στις προηγούμενες ενότητες έγινε ανάλυση ορισμένων από τους βασικότερους αλγορίθμους που επιλύουν προβλήματα εύρεσης της βέλτιστης διαδρομής από ένα σημείο του δικτύου προς όλα τα υπόλοιπα. Στην πραγματικότητα όμως τα αποτελέσματα αυτών των αλγορίθμων δε δίνουν τα δένδρα ελάχιστων διαδρομών παρά μόνο τα ελάχιστα κόστη. Έτσι για την εύρεση των δένδρων ελαχίστων διαδρομών πρέπει να επιλυθεί το εξής πρόβλημα :

Έστω δίκτυο G=(N,E) όπου κάθε σύνδεσμος του δικτύου έχει κόστος διάνυσης w(e), e∈E. Με χρήση ενός αλγορίθμου από αυτούς που αναπτύχθηκαν ήδη, μπορεί να υπολογιστεί το μικρότερο κόστος διαδρομής από τον κόμβο προέλευσης r προς όλους τους υπόλοιπους κόμβους του δικτύου : D(i), ∀ i∈1,N-{r}. Για την εύρεση του δένδρου ελάχιστης διαδρομής μπορεί να χρησιμοποιηθεί το επόμενο επαναληπτικό βήμα.

Θέτοντας ως κόμβο αναφοράς τον κόμβο προορισμού u ← i ο επόμενος κόμβος που ανήκει στο δένδρο ελάχιστης διαδρομής είναι αυτός που το άθροισμα της απόστασής του από τον αρχικό κόμβο και το βάρος σύνδεσής του με τον κόμβο u είναι το ελάχιστο. Δηλαδή αναζητείται κόμβος k ο οποίος ικανοποιεί τις σχέσεις :
D (k) + w(e) = Minimum, ∀k∈{1,N}, e∈{1,E}
Υπό τον Περιορισμό : E(e) = u
Ο κόμβος k αποθηκεύεται σε μία λίστα Q(1) ← k και η διαδικασία επαναλαμβάνεται με κόμβο αναφοράς τον κόμβο k, δηλαδή u ← k.
Η διαδικασία τερματίζεται μόλις επιστρέψουμε στον αρχικό κόμβο, δηλαδή u ← r.
Η μορφή του είναι :

ΑΛΓΟΡΙΘΜΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΔΕΝΔΡΩΝ ΕΛΑΧΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ

Βήμα 1(Αρχικοποιήσεις)

Έστω D(i) το ελάχιστο κόστος για τη μετακίνηση από τον κόμβο r στον κόμβο i∈{1,N}. Αυτό μπορεί να προκύψει με χρήση κάποιου από τους αλγορίθμους της ενότητας 5.2.

Βήμα 2(Κύριο Μέρος)

```
Για Κάθε i = 1,N
u ← i
M ← 1
Q( M ) ← u  // Θεωρείται ότι ο πρώτος κόμβος από τον οποίο αποτελείται η διαδρομή
// είναι ο κόμβος προορισμού
    Όσο u ≠ r
    K ← +∞
        Για Κάθε e = 1,E
            Εάν E( e ) = u & D( S(e) ) + w( e ) < K
            K ← D( S(e) ) + w( e )
            T ← S( e )
```

```
            Τέλος Εάν
        Τέλος Για Κάθε
    u ← T
    M ← M + 1
    Q( M ) ← u
    Τέλος Όσο
  Για Κάθε i = 1,M/2 // Τοποθέτηση των κόμβων της διαδρομής στη σωστή σειρά
  H ← Q( i )
  Q( i ) ← Q( M-i+1 )
  Q( M-i+1 ) ← H
  Τέλος Για Κάθε
Τέλος Για Κάθε
```

Τα δένδρα ελάχιστων διαδρομών βέβαια μπορούν να προκύψουν και με πιο σύντομη μέθοδο. Η μέθοδος αυτή στηρίζεται στην εξής παρατήρηση :

Για τον υπολογισμό της διαδρομής ελαχίστου κόστους από τον κόμβο r στον τυχαίο κόμβο i, αρχικά θεωρείται D(i) = +∞. Κατά τη διαδικασία επίλυσης μέσω οποιουδήποτε αλγορίθμου γίνεται πολλές φορές επανάληψη του ακόλουθου βήματος :
Εάν D(S(e)) + w(e) < D(i), όπου E(e) = i, τότε
D(i) = D(S(e)) + w(e).

Σε κάθε βήμα της επανάληψης το κόστος D(i) μπορεί να μειωθεί ή να παραμείνει όπως ήταν πριν την επανάληψη σύμφωνα με την ανίσωση. Θεωρώντας ότι έχει φθάσει η νιοστή επανάληψη κατά την οποία το κόστος D(i) έχει πάρει την ελάχιστη τιμή του, τότε :
$D(i)_{min}$ = D(S(e_x)) + w(e_x). Από αυτό προκύπτει ότι η διαδρομή ελαχίστου κόστους από το r στο i διέρχεται από τον κόμβο S(e_x) ο τελευταίος σύνδεσμος από τον οποίο αποτελείται είναι ο σύνδεσμος e_x. Αν για ∀i∈{1,N} είχε αποθηκευτεί ο τελευταίος σύνδεσμος της ελάχιστης διαδρομής από το r στο i σε ένα σύνολο Parent(i) : N→E, τότε αναδρομικά μπορεί να προκύψει η ελάχιστη διαδρομή από το r στο i ως ακολουθία συνδέσμων :
p(1) = Parent(i), δηλαδή ο πρώτος σύνδεσμος της διαδρομής είναι ο Parent(i),
k = S(Parent(i)), θεωρείται k ο κόμβος προέλευσης του συνδέσμου Parent(i),
p(2) = Parent(k), ο δεύτερος σύνδεσμος είναι ο τελευταίος σύνδεσμος της ελάχιστης διαδρομής από το r στο k.

Η διαδικασία επαναλαμβάνεται μέχρι ο κόμβος προέλευσης κάποιου συνδέσμου Parent(x) να είναι ο αρχικός : S(Parent(x)) = r, οπότε και ο Parent(x) είναι και ο τελευταίος σύνδεσμος από τον οποίο αποτελείται η διαδρομή. Σύμφωνα με τα παραπάνω προτείνεται ο ακόλουθος αλγόριθμος :

ΣΥΝΤΟΜΟΣ ΑΛΓΟΡΙΘΜΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΩΝ ΔΕΝΔΡΩΝ ΕΛΑΧΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ

Βήμα 1(Αρχικοποιήσεις)

Για Κάθε i = 1,N
D (i) ← +∞
Q (i) ← +∞
Τέλος Για Κάθε
D (r) ← 0
Q (r) ← r

Βήμα 2(Αλγόριθμος Dijkstra)

u ← r
Για Κάθε j = 1,N
Για Κάθε e = 1,E
  Εάν S(e) = u & D(E(e)) > D(S(e)) + w(e)
  D(E(e)) ← D(S(e)) + w(e)
  Parent (E(e)) ← e
  Τέλος Εάν
Τέλος Για Κάθε
K ← +∞
Για Κάθε i = 1,N
    Εάν Q(i) ≠ i & D(i) > K
    K ← D(i)
    u ← i
    Τέλος Εάν
Τέλος για Κάθε
Τέλος για Κάθε
Για Κάθε i = 1,N
k ← i
T ← 1
Αρχή 1
  Εάν k ≠ r
  p (T) = Parent (k)
  T ← T + 1
  k = S (Parent (k))
  Τύπωσε το σύνδεσμο p (T)
  Επέστρεψε στην Αρχή 1
  Τέλος Εάν
Τέλος Για Κάθε

Έτσι τυπώνονται όλοι οι σύνδεσμοι e∈p από τους οποίους αποτελείται η βέλτιστη διαδρομή από το r στο i, για ∀i∈{1,N-r}.

5.4.ΕΥΡΕΣΗ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΜΕΤΑΞΥ ΔΥΟ ΚΟΜΒΩΝ ΤΟΥ ΔΙΚΤΥΟΥ

Το πρόβλημα αυτό αποτελεί υποπερίπτωση του προβλήματος που εξετάστηκε στην ενότητα 5.2. Για το λόγο αυτό μπορούν να χρησιμοποιηθούν με μικρές τροποποιήσεις οι αλγόριθμοι που αναπτύχθηκαν στην ενότητα αυτή. Μερικές εργασίες πάνω σε αυτό το πρόβλημα είναι οι ακόλουθες : Hart et al. (1968), Nicholson (1966), Goldberg et al. (2005), Cho and Lan (2008). Υπάρχουν και άλλες εργασίες που δεν παρατίθενται εδώ και αφορούν τις περιπτώσεις που τα κόστη των συνδέσμων μεταβάλλονται με το χρόνο, δηλαδή υπάρχει αβεβαιότητα σχετικά με τις συνθήκες του δικτύου. Η βασική διατύπωση του παραπάνω προβλήματος είναι :

Ο κόμβος προορισμού s προσεγγίζεται από τον κόμβο προέλευσης r με κόστος D(s) : {s}→R^+. Εάν τυχαία διαδρομή από το r στο s αποτελείται από τους συνδέσμους p = (e_x, e_y ... . , e_z), θεωρώντας σύνολο P = {x, y, ……, z}, πρέπει :
$S(e_x)$ = r
E(e_z) = s
∀ k ϵ P-{z} : E(e_k) = S(e_{k+1})
Έτσι το κόστος της τυχαίας διαδρομής r→s δίνεται από τη σχέση :
D(s) = $\sum_{e \epsilon p} w(e)$.
Το ζητούμενο είναι να βρεθεί η διαδρομή p_* για την οποία
D_* = min{ D(s) } = Min{ $\sum_{e \epsilon p} w(e)$ }, ∀p.

Αλγόριθμοι που επιλύουν το παραπάνω πρόβλημα χρησιμοποιούνται σήμερα από τα συστήματα πλοήγησης. Εκτός από αυτούς που παρουσιάστηκαν στην Ενότητα 5.2. και μπορούν με μικρές τροποποιήσεις να επιλύσουν και αυτό το πρόβλημα θα παρουσιαστούν στη συνέχεια και άλλοι αλγόριθμοι.

5.4.1.Ο ΑΛΓΟΡΙΘΜΟΣ A STAR

Ήδη από το 1968 έγιναν οι προσπάθειες μείωσης του χρόνου επίλυσης του προβλήματος εύρεσης της ελάχιστης διαδρομής από έναν κόμβο προέλευσης προς ένα κόμβο προορισμού με ευρετικές μεθόδους. Οι προσπάθειες αυτές κατέληξαν στη διατύπωση του αλγορίθμου A* που προφέρεται ως A star. Σκοπός αυτού του αλγόριθμου είναι η εύρεση της ελάχιστης διαδρομής από ένα σημείο σε ένα άλλο μέσα σε ένα δίκτυο. Η πρώτη διατύπωσή του έγινε από τους Hart et al. (1968). Αυτός ο αλγόριθμος είναι από τους πιο αποδοτικούς σε αυτή την κατηγορία προβλημάτων (Point To Point Shortest Path Problems). Χρησιμοποιεί μία συνάρτηση με δεδομένο κόστος διαδρομής που είναι στην ουσία το κόστος διαδρομής μεταξύ δύο γειτονικών κόμβων και μία συνάρτηση που υποθέτει το κόστος διαδρομής από τον κόμβο αυτό στον κόμβο τελικού προορισμού. Η συνάρτηση που υποθέτει το κόστος από κάθε κόμβο προς τον τελικό κόμβο προορισμού πρέπει να υποθέτει ένα τέτοιο κόστος ώστε να είναι μικρότερο από το πραγματικό. Αυτό μπορεί να βελτιώσει την

αποτελεσματικότητα και την εκτέλεση του αλγορίθμου και περιορίζει τις δυνατές επιλογές που μπορούν να εξερευνηθούν.

Γενικά ο A star είναι ένας αλγόριθμος που περιέχει δύο λίστες, μία ανοικτή και μία κλειστή. Η ανοικτή λίστα είναι μία λίστα προτεραιότητας με κόμβους από την οποία μπορεί να επιλεγεί ο επόμενος κόμβος με το ελάχιστο κόστος διαδρομής σε σχέση με τον κόμβο προέλευσης. Η ανοιχτή λίστα περιέχει αρχικά όλους τους κόμβους του δικτύου, εκτός από τον κόμβο προέλευσης που τοποθετείται στην κλειστή λίστα. Κατά το πρώτο βήμα επιλέγεται ο κόμβος με το ελάχιστο κόστος προς τον αρχικό και εξετάζεται εάν αυτός είναι ο κόμβος προορισμού. Εάν αυτό συμβαίνει έχει βρεθεί η ελάχιστη διαδρομή. Αλλιώς αφαιρείται από την ανοιχτή λίστα και τοποθετείται στην κλειστή ο κόμβος j, για τον οποίο :

$D(j) + H(j) = \min$, όπου $D(j)$ η απόστασή του από τον κόμβο προέλευσης και $H(j)$ η αναμενόμενη απόστασή του προς τον κόμβο προορισμού. Έτσι επιλέγεται ένας κόμβος που έχει κατεύθυνση προς τον κόμβο προορισμού. Αυτή είναι και η ουσιαστική διαφορά από τον αλγόριθμο Dijkstra. Η απόδοση του αλγορίθμου εξαρτάται από το πόσο αντιπροσωπευτικές είναι οι τιμές της συνάρτησης $H(j), \forall j \epsilon N$. Στη συνέχεια παρουσιάζεται η μορφή του.

ΕΥΡΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ A*

Βήμα 1(Αρχικοποιήσεις)

Για Κάθε i = 1,N

D (i) ← +∞

Q (i) ← +∞

Διάβασε H (i) : Το προβλεπόμενο κόστος διαδρομής κατά τη μετακίνηση i→s

Τέλος Για Κάθε

D (r) ← 0 & Q (r) ← r

Βήμα 2(Αλγόριθμος A*) # Βασικές Πράξεις

u ← r

Όσο u ≠ s $N \times E$

Για Κάθε e = 1,E

Εάν S(e) = u & D(E(e)) > D(S(e)) + w(e)

D(E(e)) > D(S(e)) + w(e)

Τέλος Εάν

Τέλος Για Κάθε

K ← +∞

Για Κάθε i = 1,N

Εάν Q(i) ≠ i & D(i) + H(i) < K

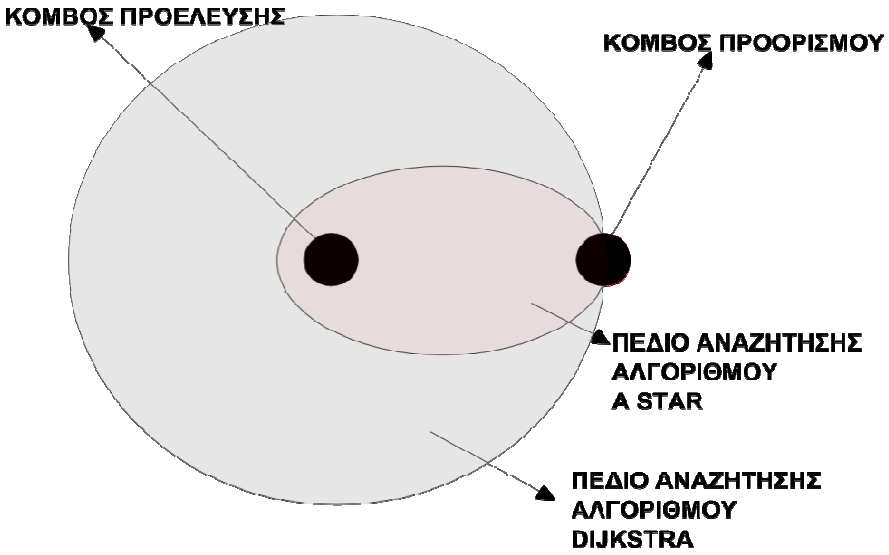
K ← D(i) + H(i)

u ← i

Τέλος Εάν

Τέλος Για Κάθε
Q(i) ← i
Τέλος Όσο

Στη χειρότερη περίπτωση ο αλγόριθμος A star έχει χρόνο εκτέλεσης O($N \times E$) εάν χρησιμοποιηθεί λίστα αναμονής για την επιλογή του κόμβου που είναι κάθε φορά πλησιέστερος στον αρχικό. Ο χρόνος αυτός όμως στην ουσία είναι πολύ μικρότερος γιατί ο αλγόριθμος επιλέγει συνέχεια κόμβους που πλησιάζουν στον κόμβο τελικού προορισμού. Οι αναμενόμενες αποστάσεις είναι ένα επιπλέον δεδομένο που επιβαρύνει τη διαδικασία καθώς είναι δύσκολο να αξιολογηθούν και να δοθούν ως στοιχείο σε ένα πολύπλοκο δίκτυο. Εάν όμως αυτό συμβεί το πεδίο αναζήτησης μειώνεται σημαντικά σε σχέση με τον αλγόριθμο Dijkstra και έχει ελλειψοειδή μορφή όπως φαίνεται και στο ακόλουθο σχήμα :



Σχήμα 5.8.Πεδίο Αναζήτησης αλγορίθμων A star και Dijkstra

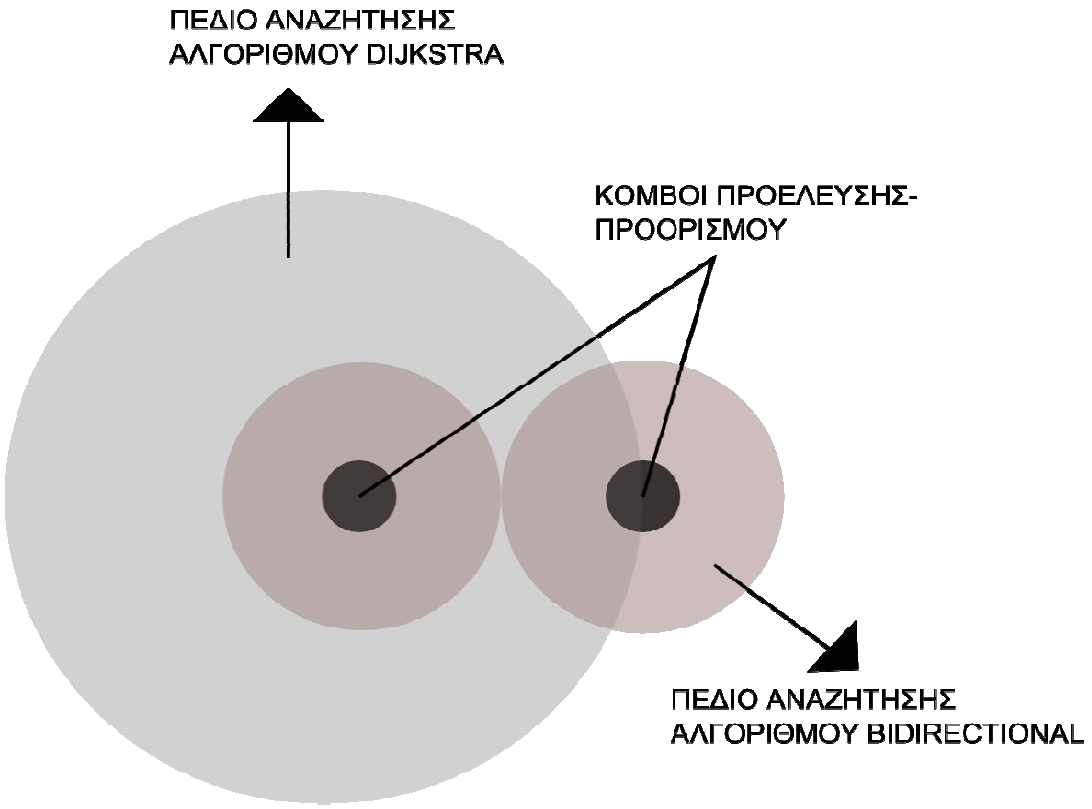
5.4.2.Ο ΑΛΓΟΡΙΘΜΟΣ ΔΙΠΛΗΣ ΚΑΤΕΥΘΥΝΣΗΣ (BIDIRECTIONAL)

Ο αλγόριθμος διπλής κατεύθυνσης έχει απλή μορφή του είναι απλή και στηρίζεται στον αλγόριθμο Dijkstra. Ο αλγόριθμος αυτός αναζητά την συντομότερη διαδρομή μεταξύ δύο κόμβων εκτελώντας δύο ταυτόχρονες αναζητήσεις :

α) Από τον αρχικό κόμβο χρησιμοποιώντας τον αλγόριθμο Dijkstra αναζητά διαδοχικά τους επόμενους κόμβους που είναι πιο κοντά σε αυτόν.

β) Από τον τελικό κόμβο χρησιμοποιώντας τον αλγόριθμο Dijkstra και εκτελώντας αντίστροφη διαδικασία αναζητά διαδοχικά τους πλησιέστερους κόμβους προς αυτόν.

γ) Ο αλγόριθμος τερματίζει την αναζήτησης όταν βρεθεί κοινός κόμβος από τις δύο διαδικασίες. Έτσι περιορίζεται ουσιαστικά το πεδίο αναζήτησης σε σχέση με την απλή χρήση του αλγορίθμου Dijkstra όπως παρουσιάζεται και στο παρακάτω σχήμα :



Σχήμα 5.9.Πεδίο Αναζήτησης αλγορίθμων Bidirectional και Dijkstra

Η μορφή του αλγόριθμου είναι :

ΕΥΡΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΔΙΠΛΗΣ ΚΑΤΕΘΥΝΣΗΣ

Βήμα 1(Αρχικοποιήσεις)

Για Κάθε i = 1,N {
D1(i) ← +∞
D2(i) ← +∞
Q1(i) ← +∞
Q2(i) ← +∞
Τέλος για Κάθε
D1(r) ← 0
D2(s) ← 0
u ← r & uu ← s
Q1(r) ←0 & Q2(s)←0

Βήμα 2(Αλγόριθμος Bidirectional)

Για Κάθε e = 1,E
Εάν S(e) = u & D1(E(e)) > D1 (S (e)) + w(e)
D1(E(e)) ← D1 (S (e)) + w(e)

```
  Q1( E(e) ) ← D1( E(e) )
  Εάν E( e ) = uu & D2( S(e) ) > D2( E(e) ) + w( e )
  D2( S(e) ) ← D2( E(e) ) + w( e )
  Q2( S(e) ) ← D2( E(e) )
  Τέλος Εάν
Τέλος Για Κάθε
K1 ← +∞ & K2 ← +∞
Για Κάθε i = 1,N
  Εάν Q1( i ) ≠ 0 & D1( i ) < K1
  K1←D1( i )
  u ← i
  Τέλος Εάν
  Εάν Q2( i ) ≠ 0 & D2( i ) < K2
  K2 ← D2( i )
  uu ← i
  Τέλος Εάν
Τέλος Για Κάθε
Q1( u ) ← 0 & Q2( uu ) ← 0
Για Κάθε i = 1,N
 Εάν Q1( i ) = 0 & uu = i
 Η διαδικασία τερματίζεται και υπολογίζεται η απόσταση της
 Συντομότερης διαδρομής ως D1(uu) + D2(uu)
 Τέλος Εάν
Τέλος Για Κάθε
Επέστρεψε στην αρχή και επανέλαβε τη διαδικασία.
```

5.4.3.ΑΛΓΟΡΙΘΜΟΙ ΜΕ ΧΡΗΣΗ ΙΕΡΑΡΧΙΑΣ

Για να χρησιμοποιηθούν οι αλγόριθμοι στα συστήματα πλοήγησης είναι πολύ σημαντικό να έχουν μεγάλη ταχύτητα εκτέλεσης και να μη δεσμεύουν πολύ χώρο στη μνήμη της υπολογιστικής μηχανής. Επειδή τα δίκτυα παρέχουν χιλιάδες εναλλακτικές διαδρομές τις οποίες μπορεί να ακολουθήσει ο χρήστης η ανάγκη αυτή είναι επιτακτική. Όπως έχει επισημανθεί ήδη ο αλγόριθμος Dijkstra με χρήση του σωρού Fibonnaci ως δομή δεδομένων είναι ο ταχύτερος σε δίκτυα με θετικά βάρη συνδέσμων για εύρεση των ελάχιστων διαδρομών προς όλες τις κατευθύνσεις, δεν είναι όμως καλύτερη δυνατή επιλογή στη επίλυση των Point to point shortest path problems. Εάν ληφθεί υπόψη η πραγματική διαδικασία λήψης απόφασης από το χρήστη, αν και είναι πολύ δύσκολο από μόνος του να επιλέξει την ελάχιστη διαδρομή προς τον προορισμό του θα επιλέξει ωστόσο μία πολύ καλή εναλλακτική της σε πολύ λίγο χρόνο. Αυτό συμβαίνει διότι η σκέψη του δομείται ιεραρχικά. Ιεραρχεί δηλαδή τις επιλογές του και επιλέγει την καλύτερη λύση σκεπτόμενος μόνο τις

σημαντικότερες εναλλακτικές. Με τον τρόπο αυτό αποφεύγει την εξερεύνηση σε όλο το δίκτυο, κάτι που θα ήταν πρακτικά αδύνατο.

Τα τελευταία χρόνια έχουν αρχίσει να αναπτύσσονται αλγόριθμοι που ακολουθούν τη διαδικασία ιεράρχησης του δικτύου ώστε η επίλυση να είναι ευκολότερη και πιο σύντομη. Οι αλγόριθμοι αυτής της μορφής επικεντρώνονται αρχικά στα βασικότερα σημεία και στη συνέχεια να ασχολούνται με τις λεπτομέρειες. Μία μελέτη των αλγορίθμων που βασίζονται στην ιεράρχηση έχει γίνει από τους Huang and Jin (1996). Στην εργασία τους χώρισαν ένα μεγάλο οδικό δίκτυο σε μικρότερα και μέσα σε αυτά υπολόγισαν τις ελάχιστες διαδρομές από όλα τα σημεία προς όλα τα σημεία. Αν και ο αλγόριθμος που ανέπτυξαν έχει πολύ μεγάλη ταχύτητα χρησιμοποιεί κι αυτός προϋπολογισμένες ελάχιστες διαδρομές και απαιτείται πολύ μεγάλη χωρητικότητα μνήμης για την υλοποίησή του. Ένα άλλο μειονέκτημα αυτών των αλγορίθμων είναι, όπως είχε επισημανθεί και στην εργασία των Car and Frank (1994), ότι στο 50% των περιπτώσεων δε βρίσκουν την ελάχιστη διαδρομή αλλά μία καλή εναλλακτική της. Στη συνέχεια ακολουθεί η περιγραφή ενός αλγορίθμου ο οποίος οδηγεί το μεταφορικό μέσο σε μεγάλες οδικές αρτηρίες και προτάθηκε από τους Sanders and Schultes (2005) και Cho and Lan (2009). Κάτι τέτοιο φαίνεται αρχικά παράλογο από άποψη βελτιστοποίησης, ωστόσο συχνά οι χρήστες προτιμούν μία διαδρομή σε καλό οδόστρωμα που διέρχεται από σημαντικές οδικές αρτηρίες από τη διαδρομή ελαχίστου κόστους. Το γεγονός αυτό βοηθά τον αλγόριθμο αναζήτησης, καθώς όταν το μεταφορικό μέσο εισέλθει σε μία κεντρική αρτηρία μειώνονται οι εναλλακτικές επιλογές. Θεωρώντας ένα πολύπλοκο οδικό δίκτυο μπορούμε να το διασπάσουμε σε τρεις περιοχές. Στην περιοχή όπου βρίσκεται το σημείο εκκίνησης του χρήστη και εκτείνεται μέχρι την πιο κοντινή ελεύθερη λεωφόρο ή άλλο μεγάλο οδικό άξονα. Χρησιμοποιώντας τη λογική της ιεράρχησης και θεωρώντας ότι η μέση ταχύτητα μέσω μεγάλων οδικών αξόνων είναι μεγαλύτερη (κάτι που πρέπει να ερευνηθεί με περισσότερη επιφύλαξη) βρίσκουμε την ελάχιστη διαδρομή από το σημείο εκκίνησης προς τον κοντινότερο μεγάλο οδικό άξονα. Ταυτόχρονα χρησιμοποιώντας τη λογική των αλγορίθμων διπλής κατεύθυνσης βρίσκουμε την ελάχιστη διαδρομή από το σημείο τελικού προορισμού με την κοντινότερη έξοδο από κάποιον μεγάλο οδικό άξονα. Τέλος από το αρχικό σημείο εισόδου σε κάποιο μεγάλο οδικό άξονα μέχρι το εξόδου από αυτόν υπολογίζουμε την ελάχιστη διαδρομή. Συνδυάζοντας τέλος όλα αυτά βρίσκουμε το συνολικό δένδρο ελάχιστων διαδρομών που θα προταθεί στο χρήστη.

Για την εύρεση των ελάχιστων διαδρομών σε κάθε επιμέρους τμήμα μπορεί να χρησιμοποιηθεί ο αλγόριθμος Dijkstra που είναι και πιο ακριβής [D-D-D] ή ο αλγόριθμος A* [A-A-A] ή στο πυκνό δίκτυο ο A* που είναι ταχύτερος και στις λεωφόρους ο Dijkstra [A-D-A]. Η μορφή του αλγορίθμου παρουσιάζεται ακολούθως.

ΕΥΡΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ D-D-D

Βήμα 1(Αρχικοποιήσεις)

Οι κόμβοι εισόδου στις ελεύθερες λεωφόρους και εξόδου από αυτές επισημαίνονται για να είναι αναγνωρίσιμοι, για παράδειγμα ανήκουν στα σύνολα R={R1, R2, …}, T={T1, T2, ..} αντίστοιχα.

Για Κάθε i = 1,N

D1(i) ← +∞ & D2(i) ← +∞ & Q1(i) ← +∞ & Q2(i) ← +∞

Τέλος Για Κάθε

D1(r) ← r & D2(s) ← s & Q1(r) ← r & Q2(s) ← s

Βήμα 2(Χρήση αλγορίθμου Dijkstra)

Από το σημείο εκκίνησης r μέσω του αλγόριθμου Dijkstra βρίσκουμε την ελάχιστη διαδρομή προς τον κοντινότερο κόμβο ελεύθερης λεωφόρου με τη μέθοδο:

u ← r

Όσο u ≠ s ή u ∩ R = Ø

Για Κάθε e = 1,E

  Εάν S(e) = u & D1(E(e)) > D1(S(e)) + w(e)

  D1(E(e)) ← D1(S(e)) + w(e)

  Τέλος Εάν

Τέλος Για Κάθε

K ← +∞

Για Κάθε j=1,N

  Εάν D1(j) < K & Q1(j) ≠ j

  K ← D1(j)

  u ← j

  Τέλος Εάν

Τέλος Για Κάθε

Q1(u) ← u

k ← u

Εάν u = s : τερμάτισε την εφαρμογή γιατί βρέθηκε ο κόμβος προορισμού

Τέλος Όσο

Βήμα 3(Χρήση αλγορίθμου Dijkstra)

Από το σημείο προορισμού s μέσω του αλγόριθμου Dijkstra βρίσκουμε την ελάχιστη διαδρομή προς τον κοντινότερο κόμβο εξόδου με τη μέθοδο:

Όσο u ≠ r ή u ∩ T = Ø

Για Κάθε e = 1,E

  Εάν E(e) = u & D2(S(e)) > D2(E(e)) + w(e)

  D2(S(e)) ← D2(E(e)) + w(e)

  Τέλος Εάν

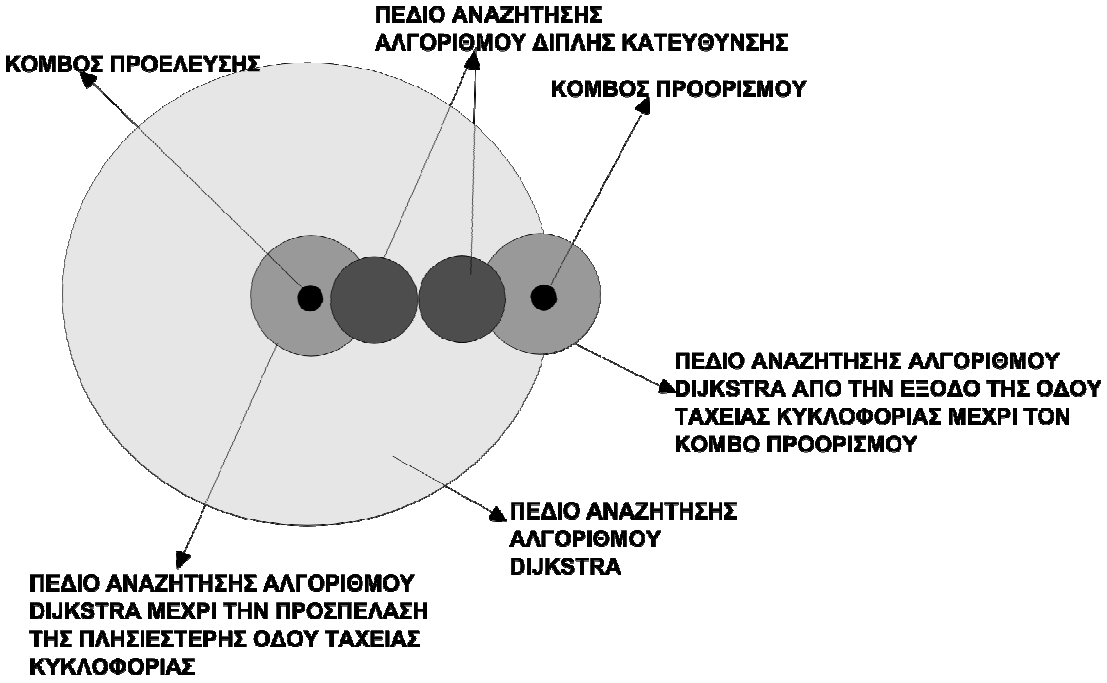
Τέλος Για Κάθε

K ← +∞

Για Κάθε j=1,N
  Εάν D2(j) < K & Q2(j) ≠ j
  K ← D2(j)
  u ← j
  Τέλος Εάν
Τέλος Για Κάθε
Q2(u) ← u
Εάν u = r : τερμάτισε την εφαρμογή γιατί βρέθηκε ο κόμβος προορισμού
Τέλος Όσο

Βήμα 4(Χρήση αλγορίθμου Διπλής Κατεύθυνσης [Bidirectional])
Από το σημείο εισόδου σε ελεύθερη λεωφόρου 'k' μέχρι το σημείο εξόδου από αυτήν 'u' βρες την ελάχιστη διαδρομή μέσω του αλγόριθμου διπλής κατεύθυνσης :
Αρχή :
Για Κάθε e = 1,E
  Εάν S(e) = k & D1(E(e)) > D1 (S (e)) + w(e)
  D1(E(e)) ← D1 (S (e)) + w(e)
  Q1(E(e)) ← D1(E(e))
  Εάν E(e) = u & D2(S(e)) > D2(E(e)) + w(e)
  D2(S(e)) ← D2(E(e)) + w(e)
  Q2(S(e)) ← D2(E(e))
  Τέλος Εάν
Τέλος Για Κάθε
K1 ← +∞ & K2 ← +∞
Για Κάθε i = 1,N
  Εάν Q1(i) ≠ i & D1(i) < K1
  K1←D1(i)
  k ← i
  Τέλος Εάν
  Εάν Q2(i) ≠ i & D2(i) < K2
  K2 ← D2(i)
  u ← i
  Τέλος Εάν
Τέλος Για Κάθε
Q1(k) ← k & Q2(u) ← u
Για Κάθε i = 1,N
  Εάν Q1(i) = i & u = i
  Η διαδικασία τερματίζεται και υπολογίζεται η απόσταση της
  Συντομότερης διαδρομής ως D1(u) + D2(u)
  Τέλος Εάν
Τέλος Για Κάθε
Επέστρεψε στην αρχή και επανέλαβε τη διαδικασία

Το πεδίο αναζήτησης του αλγορίθμου που αναπτύχθηκε μπορεί να παρασταθεί σχηματικά σε σύγκριση με το αντίστοιχο του αλγορίθμου Dijkstra :



Σχήμα 5.10.Πεδίο Αναζήτησης αλγορίθμου D-D-D σε σύγκριση με το πεδίο αναζήτησης του αλγόριθμου Dijkstra

5.5.ΕΥΡΕΣΗ ΤΩΝ ΒΕΛΤΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ ΑΠΟ ΟΛΟΥΣ ΤΟΥΣ ΚΟΜΒΟΥΣ ΠΡΟΣ ΟΛΟΥΣ ΤΟΥΣ ΚΟΜΒΟΥΣ ΕΝΟΣ ΔΙΚΤΥΟΥ

Το πρόβλημα αυτό αποτελεί τη γενικότερη μορφή του προβλήματος εύρεσης των βέλτιστων διαδρομών από έναν κόμβο προς τους υπόλοιπους του δικτύου. Ορισμένες εργασίες σε αυτή την κατεύθυνση είναι : Roy (1959), Warshall (1962), Floyd (1962), Johnson, (2007), Mofat and Takaoka (1987), Seidel (1992), Takaoka (2004). Η γενικότερη διατύπωση του προβλήματος είναι :

Κάθε κόμβος i ϵ N-{ j } προσεγγίζεται από τον κόμβο προέλευσης j με κόστος D(i) : N-{ j } $\rightarrow R^+$. Εάν τυχαία διαδρομή από το j στο i αποτελείται από τους συνδέσμους p = (e_x, e_y ... ., e_z), θεωρώντας σύνολο P = {x, y, ……, z}, πρέπει :
$S(e_x)$ = j
E(e_z) = i
∀ k ϵ P-{z} : E(e_k) = S(e_{k+1})
Έτσι το κόστος της τυχαίας διαδρομής j→i δίνεται από τη σχέση :
D(i) = $\sum_{e \epsilon p} w(e)$.

Το ζητούμενο είναι να βρεθεί η διαδρομή για την οποία D(i) = Minimum
Αυτή η διαδικασία επαναλαμβάνεται για ∀ i ϵ N-{j}, δηλαδή για κάθε κόμβο.
Στη συνέχεια αυτή η διαδικασία επαναλαμβάνεται θέτοντας κόμβο προέλευσης κάθε κόμβο του δικτύου, δηλαδή επαναλαμβάνεται για ∀ j ϵ N.

Οι αλγόριθμοι που παρουσιάζονται στις ενότητες 5.5.1. και 5.5.2. έχουν εφαρμογή στην επίλυση αυτού του προβλήματος αφού πρώτα τροποποιηθούν κατάλληλα.

5.5.1.Ο ΑΛΓΟΡΙΘΜΟΣ FLOYD - WARSHALL

Ο αλγόριθμος αυτός ανήκει στην κατηγορία του δυναμικού προγραμματισμού. Δημοσιεύθηκε από τον Floyd (1962) και ήταν παρόμοιος με τους αλγορίθμους των Bernerd Roy και Stephen Warshall που είχαν δημοσιευθεί εκείνη την περίοδο. Ο αλγόριθμος αυτός συγκρίνει όλες τις δυνατές διαδρομές σε ένα γράφημα μεταξύ κάθε ζεύγους κόμβων. Μπορεί να το κάνει αυτό με Θ(N^3) συγκρίσεις στο γράφημα, όπου N ο αριθμός των κόμβων. Χρησιμοποιείται έτσι για την εύρεση της ελάχιστης διαδρομής από κάθε κόμβο του δικτύου προς όλους τους υπόλοιπους κόμβους (All Pairs Shortest path Problem). Στην ουσία ο αλγόριθμος αυτός στηρίζεται στον αλγόριθμο Moore-Bellman-Ford, ο οποίος υπολογίζει τα ελάχιστα κόστη διαδρομών από τον κόμβο r προς τους υπόλοιπους κόμβους του δικτύου : D(i), ∀ i ϵ N. Η διαφοροποίησή του είναι ότι επαναλαμβάνει N φορές τη μέθοδο του αλγόριθμου Moore-Bellman-Ford χρησιμοποιώντας κάθε φορά διαφορετικό κόμβο προέλευσης j ϵ N, μέχρι να χρησιμοποιηθούν όλοι οι κόμβοι του δικτύου.

Η μορφή του παρουσιάζεται ακολούθως.

ΑΛΓΟΡΙΘΜΟΣ FLOYD - WARSHALL

Βήμα 1(Αρχικοποιήσεις)

#Βασικές Πράξεις

Για Κάθε j = 1,N
  Για Κάθε i = 1,N
  D (j)(i) ← +∞
  Τέλος Για Κάθε
Τέλος για Κάθε

Βήμα 2(Αλγόριθμος Floyd-Warshall)

Για Κάθε j = 1,N
  D(j)(j) ← 0
  Για Κάθε k = 1,N
    Για Κάθε e = 1,E $N \times N \times E$
      Εάν D(j)(E(e)) > D(j)(S(e)) + w(e)
      D(j)(E(e)) ← D(j)(S(e)) + w(e)
      Τέλος Εάν
    Τέλος Για Κάθε
  Τέλος Για Κάθε
Τέλος Για Κάθε

Τυπικά, ο αλγόριθμος Floyd–Warshall βρίσκει μόνο τις ελάχιστες αποστάσεις των διαδρομών μεταξύ όλων των κόμβων του δικτύου. Με κάποιες τροποποιήσεις μπορεί να βρεθεί μία μέθοδος που θα κατασκευάζει και τα δέντρα ελάχιστων διαδρομών μεταξύ όλων των κόμβων. Για να αποθηκευτούν ωστόσο όλα τα μονοπάτια σε θέσεις μνήμης απαιτείται μεγάλος αποθηκευτικός χώρος. Οι πληροφορίες για την κατασκευή αυτών των διαδρομών μπορούν να δοθούν από ένα πίνακα $N \times N$, π.χ.next(i,j), όπου θα αναπαριστά τον κόμβο από τον οποίο πρέπει να διέλθει η διαδρομή με αρχικό κόμβο τον i και τελικό το j. Έτσι για την εύρεση του συντομότερου μονοπατιού από το i στο j αρκεί να βρεθούν τα συντομότερα μονοπάτια από το i στο next(i,j) και από το next(i,j) στο j. Αυτό μπορεί να επιτευχθεί μέσω της δημιουργίας κατάλληλης αναδρομικής συνάρτησης.

5.5.2.Ο ΑΛΓΟΡΙΘΜΟΣ JOHNSON

Αυτός ο αλγόριθμος πήρε το όνομά του από τον Johnson (1977). Χρησιμοποιείται για την επίλυση των All Pairs Shortest Path Problems, δηλαδή προβλήματα εύρεσης ελάχιστων διαδρομών μεταξύ όλων των κόμβων του δικτύου. Δέχεται αρνητικά βάρη στους συνδέσμους αρκεί βέβαια οι σύνδεσμοι αυτοί να μη σχηματίζουν κύκλο που δεσμεύει την επίλυση του προβλήματος. Τα βασικά βήματα του αλγορίθμου που αποτελούν και την κεντρική του ιδέα αποτελούνται από τη χρήση άλλων αλγορίθμων που έχουν αναπτυχθεί μέχρι τώρα. Τα βήματα αυτά είναι :

1)Αλλαγή στα βάρη του γραφήματος ώστε να μην υπάρχει κόμβος με αρνητικό βάρος συνδέσμου (reweighten graph). Έστω ένας κόμβος J που συνδέεται με άλλους κόμβους του δικτύου, τότε για κάθε τυχαίο κόμβο u του δικτύου υπάρχει μία απόσταση D(u) μεταξύ του κόμβου J και του κόμβου u η οποία υπολογίζεται από τον αλγόριθμο του Moore για εύρεση ελάχιστων αποστάσεων από σταθερό σημείο προς όλα τα υπόλοιπα σημεία του δικτύου. Το νέο βάρος κάθε συνδέσμου είναι $w'(e) = D(S(e)) + w(e) – D(E(e))$. Έτσι όλα τα βάρη των συνδέσμων παίρνουν θετικές τιμές.

2)Επίλυση του προβλήματος με χρήση του αλγορίθμου Dijkstra στο δίκτυο με τα θετικά βάρη συνδέσμων που προέκυψαν.

Η δομή του αλγορίθμου Johnson έχει τη μορφή :

ΑΛΓΟΡΙΘΜΟΣ JOHNSON

Βήμα 1(Αρχικοποιήσεις) #Βασικές Πράξεις
Πρόσθεσε νέο κόμβο J στο δίκτυο με μηδενικό βάρος συνδέσμων προς όλους τους υπόλοιπους κόμβους του δικτύου
Για Κάθε i = 1,N
w(J)(i) ← 0
Τέλος Για Κάθε

Βήμα 2(Χρήση Αλγορίθμου Moore)
Για Κάθε u = 1,N *N*
D(u) ← +∞
Τέλος Για Κάθε
D(J) ← 0
Για Κάθε e = 1,E *E*
Εάν D(S(e)) + w(e) < D(E(e))
D(E(e)) ← w(e) + D(S(e))
Τέλος Εάν
Τέλος Για Κάθε
Για Κάθε e = 1,E //Αλλαγή στα βάρη συνδέσμων *E*
$w'(e)$ ← D(S(e)) + w(e) – D(E(e))
Τέλος Για Κάθε
Βήμα 3(Χρήση Αλγορίθμου Dijkstra)
Για Κάθε r = 1,N // Επανάληψη του αλγορίθμου Dijkstra *N* φορές
Για Κάθε j = 1,N
D (j) ← +∞
Q(j) ← +∞
Τέλος Για Κάθε
D(r) ← 0

```
Q ( r ) ← r
u ← r
Για Κάθε j = 1,N
Για Κάθε e = 1,E                                          E ×N
  Εάν S( e ) = u & D( E(e) ) > D( S(e) ) + w( e )
  D( E(e) ) ← D( S(e) ) + w( e )
  Τέλος Εάν
Τέλος Για Κάθε
K ← +∞
Για Κάθε i = 1,N
     Εάν Q( i ) ≠ i & D( i ) > K
     K ← D( i )
     u ← i
     Τέλος Εάν
Τέλος για Κάθε
Τέλος για Κάθε
Τέλος Για Κάθε
```

Όσον αφορά το χρόνο υλοποίησης του αλγορίθμου, απαιτείται θεωρητικός χρόνος $O(\mathrm{E} \times \mathrm{N})$ για την επίλυση του δικτύου με χρήση του αλγορίθμου Moore εφόσον δεν προκύπτει αρνητικός κύκλος. Έπειτα στο δίκτυο με μη αρνητικά βάρη συνδέσμων χρησιμοποιείται N φορές ο αλγόριθμος Dijkstra για την εύρεση της ελάχιστης διαδρομής μεταξύ όλων των κόμβων του δικτύου, κάτι για το οποίο απαιτείται χρόνος $O(\mathrm{N} \times (\mathrm{E}+\mathrm{N}\log\mathrm{N}))$ εάν χρησιμοποιηθεί ως δομή δεδομένων ο σωρός Fibonacci. Έτσι ο συνολικός χρόνος εκτέλεσης του αλγορίθμου προκύπτει ως $O(N \times E + N^2 logN)$. Προκύπτει δηλαδή τετραγωνικός χρόνος εκτέλεσης. Εάν χρησιμοποιηθεί ο δυαδικός σωρός ως δομή δεδομένων ο θεωρητικός χρόνος εκτέλεσης είναι αντίστοιχα $O(N \times E + N^2)\log\mathrm{N}$ και εάν χρησιμοποιηθεί η απλή λίστα $O(N^3)$.

5.6.ΕΥΡΕΣΗ ΟΛΩΝ ΤΩΝ ΔΙΑΔΡΟΜΩΝ ΜΕΤΑΞΥ ΔΥΟ ΚΟΜΒΩΝ ΜΕ ΚΑΤΑΤΑΞΗ ΒΑΣΕΙ ΤΟΥ ΕΛΑΧΙΣΤΟΥ ΚΟΣΤΟΥΣ

Αυτό το πρόβλημα παρουσιάζεται στη διεθνή βιβλιογραφία ως "the kth shortest path problem". Η επίλυσή του παρουσιάζει ιδιαίτερο ενδιαφέρον για τα συστήματα πλοήγησης καθώς ο χρήστης μπορεί να μην προτιμά πάντα τη βέλτιστη διαδρομή αλλά κάποια άλλη εναλλακτική της. Ορισμένες εργασίες σε αυτή την κατεύθυνση είναι των Hoffman and Pavley (1959), Bellman and Kalaba (1960), Pollack (1961), Dreyfus (1969), Yen (1971), Yen (1972), Lawler (1972), Katoh et al. (1982), Eppstein (1998), Martins and Pasqoal (2003). Η γενικότερη μορφή του προβλήματος είναι :

Ο κόμβος προορισμού s προσεγγίζεται από τον κόμβο προέλευσης r με κόστος D(s) : {s}→R^+. Εάν τυχαία διαδρομή από το r στο s αποτελείται από τους συνδέσμους p = (e_x, e_y ... . , e_z), θεωρώντας σύνολο P = {x, y, ……, z}, πρέπει :
$S(e_x)$ = r
E(e_z) = s
∀ k ϵ P-{z} : E(e_k) = S(e_{k+1})
Έτσι το κόστος της τυχαίας διαδρομής r→s δίνεται από τη σχέση :
D(s) = $\sum_{e \epsilon p} w(e)$.

Το ζητούμενο είναι να βρεθεί η διαδρομή για την οποία D(s) = Minimum.
Έπειτα η διαδρομή αυτή αποθηκεύεται σε ένα κλειστό σύνολο και το ζητούμενο είναι να βρεθεί η επόμενη διαδρομή για την οποία D(s) = Minimum.
Η διαδικασία ολοκληρώνεται όταν δεν υπάρχει άλλη διαθέσιμη διαδρομή για τη μετακίνηση r→s.

Στη συνέχεια της ενότητας παρουσιάζονται ορισμένοι από τους πιο γνωστούς αλγόριθμους επίλυσης αυτού του προβλήματος.

5.6.1.Ο ΑΛΓΟΡΙΘΜΟΣ HOFFMAN-PAVLEY

Όπως αναφέρθηκε ήδη συχνά είναι χρήσιμο για το χρήστη ενός συγκοινωνιακού δικτύου να γνωρίζει όχι μόνο τη συντομότερη διαδρομή μεταξύ δύο κόμβων, αλλά και τη $2^η$ συντομότερη και τη $3^η$ συντομότερη κ.ο.κ. Μπορεί να οριστεί αρχικά ότι δύο διαδρομές θεωρούνται διαφορετικές εάν έχουν έστω και έναν κόμβο διαφορετικό ή υπάρχει αλλαγή στη σειρά προσπέλασης των κόμβων. Κάθε διαδρομή όμως θα πρέπει να έχει κοινούς κόμβους με τη συντομότερη διαδρομή, οι οποίοι μπορεί να είναι στην ακραία περίπτωση μόνο ο κόμβος προέλευσης και ο κόμβος προορισμού. Έτσι κάθε διαδρομή έχει τουλάχιστον δύο κοινούς κόμβους με την συντομότερη διαδρομή. Οι παρατηρήσεις αυτές χρησιμοποιούνται από τους περισσότερους αλγόριθμους επίλυσης αυτού του προβλήματος, οι οποίοι καλούνται και αλγόριθμοι διαχωρισμού (Deviation algorithms). Ο όρος αυτός χρησιμοποιείται γιατί οποιαδήποτε εναλλακτική διαδρομή από το r στο s έχει κοινούς κόμβους με τη

βέλτιστη μέχρι ενός σημείου (κόμβος αλλαγής), πέραν του οποίου εμφανίζεται κάποιος διαχωρισμός.

Ο αλγόριθμος Hoffman-Pavley χρησιμοποιείται για την εύρεση μόνο της $2^{ης}$ διαδρομής ελαχίστου κόστους μεταξύ δύο κόμβων r, s και θα εξεταστεί αρχικά λόγω της απλότητάς του. Κατά την υλοποίησή του ακολουθούνται τα παρακάτω στάδια.

Αρχικά με χρήση ενός αλγορίθμου, π.χ. αλγορίθμου Bellman-Kalaba ή Dijkstra προσδιορίζεται η ελάχιστη διαδρομή p = {r, x_1, x_2, …., x_q, s} μεταξύ δύο κόμβων προέλευσης προορισμού r, s η οποία έχει μία τιμή κόστους που προκύπτει ως D(s) = $\sum_{e \epsilon p} w(e)$. Έπειτα για τον κόμβο x_q ο οποίος ανήκει στην ελάχιστη διαδρομή p και συνδέεται με τον κόμβο προορισμού s μέσω ενός μόνο συνδέσμου, υπολογίζεται η ελάχιστη διαδρομή από τον κόμβο x_q στον κόμβο s μέσω ενός άλλου κόμβου. Εάν ο κόμβος x_q δε συνδέεται με κανέναν άλλο κόμβο εκτός του N, τότε v(x_q) = +∞, όπου με v(x_q) συμβολίζεται το κόστος της δεύτερης διαδρομή ελαχίστου κόστους από τον κόμβο x_q προς τον κόμβο s. Έπειτα από τον κόμβο x_{q-1} που ανήκει στη διαδρομή p και απέχει κατά δύο συνδέσμους από τον κόμβο s υπολογίζεται το κόστος της ελάχιστης διαδρομής από τον κόμβο r προς τον κόμβο s που δε διέρχεται από τον κόμβο x_q και αποθηκεύεται ως ποσότητα A.

Επίσης υπολογίζεται και η ποσότητα B, όπου :
B = w(e) + v(x_q) με S(e) = x_{q-1}, E(e) = x_q.
Τότε η τιμή v(x_{q-1}) = min{A, B} και αυτή η διαδικασία επαναλαμβάνεται συνεχώς μέχρι τον κόμβο r. Στη χειρότερη περίπτωση απαιτούνται N επαναλήψεις του αλγορίθμου Dijkstra.

Η ορθότητα αυτού του αλγορίθμου προκύπτει από τις ακόλουθες παρατηρήσεις :

α) Η $2^{η}$ διαδρομή ελαχίστου κόστους από το r στο s ακολουθεί τους ίδιους κόμβους με τη διαδρομή p μέχρι κάποιον κόμβο x_i ϵ p-{s} πέραν του οποίου διαφοροποιείται.

β) Η $2^{η}$ διαδρομή ελαχίστου κόστους από το r στο s που διαχωρίζεται από την p στον κόμβο x_i έχει κόστος ίσο με το άθροισμα του κόστος της ελάχιστης διαδρομής από το r στο x_i : D(x_i) και του κόστους της $2^{ης}$ διαδρομής ελαχίστου κόστους από το x_i στο s : v(x_i), οπότε το κόστος της είναι D(x_i) + v(x_i). Η $2^{η}$ διαδρομή ελαχίστου κόστους από το x_i στο s προκύπτει ως η ελάχιστη διαδρομή από το x_i στο s και μπορεί να προσδιοριστεί μέσω του αλγορίθμου Dijkstra θεωρώντας ότι ο σύνδεσμος που συνδέει τον κόμβο x_i με τον επόμενο κόμβο του στη διαδρομή p : x_{i+1} προσωρινά αφαιρείται. Έτσι η νέα διαδρομή διαχωρίζεται από τη διαδρομή p στον κόμβο x_i καθώς σίγουρα μετά τον κόμβο x_i δεν οδηγείται στον κόμβο x_{i+1}.

γ) Η επίλυση του προβλήματος εκφυλίζεται στην εύρεση του κόμβου x_i ϵ p-{s} στον οποίο η $2^{η}$ διαδρομή ελαχίστου κόστους διαφοροποιείται από την p ώστε :
D(x_i) + v(x_i) = Ελάχιστο. Έτσι θεωρώντας αρχικά κόμβο διαχωρισμού x_i = x_q και συνεχίζοντας τη διαδικασία μέχρι x_i = r προκύπτει ως αποτέλεσμα το κόστος της $2^{ης}$

διαδρομής ελαχίστου κόστους από το r στο s και ο κόμβος $x_i \in$ p-{s} μετά τον οποίο διαφοροποιείται από την p.

Συνοπτικά η μορφή του αλγορίθμου είναι :

ΑΛΓΟΡΙΘΜΟΣ HOFFMAN-PAVLEY

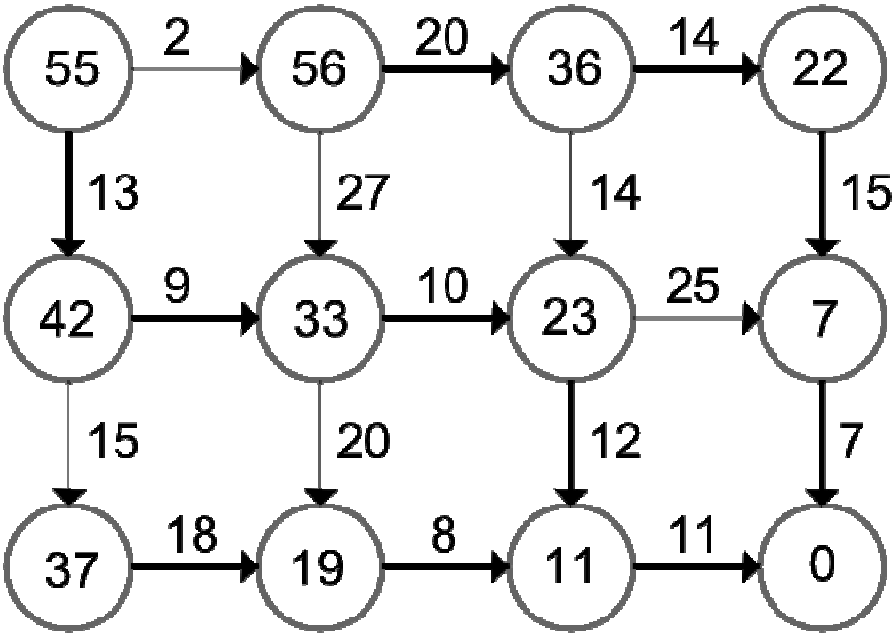
Βήμα 1(Αρχικοποιήσεις)
Δίκτυο G=(N,N^2), όπου N ο αριθμός των κόμβων και N^2 ο αριθμός των συνδέσμων του δικτύου (πυκνό δίκτυο).
Βήμα 2(Εύρεση Συντομότερης Διαδρομής μεταξύ r, s)
Χρησιμοποιώντας έναν αλγόριθμο εύρεσης συντομότερης διαδρομής, π.χ. αλγόριθμο Dijkstra αποθηκεύονται οι κόμβοι από τους οποίους αποτελείται η συντομότερη διαδρομή p. Αν το πλήθος αυτών των κόμβων είναι k, τότε κάθε ένας από αυτούς είναι αποθηκευμένος στη μήτρα p(i), για ∀iϵ{1,k}. Έτσι p(1) = r και p(k) = s.
Βήμα 3(Εύρεση $2^{ης}$ Συντομότερης Διαδρομής μεταξύ r, s)

```
u ← 0
V ← +∞
Όσο u ≠ k-1
u ← u+1
H ← w( p(k-u) )( p(k-u+1) )
w( p(k-u) )( p(k-u+1) ) ← +∞   /// Το βάρος μεταξύ των κόμβων p(ks-u) και
/// p(k-u+1) που ανήκουν στη βέλτιστη διαδρομή γίνεται ίσο με άπειρο ώστε να
/// θεωρηθεί προσωρινά ότι δεν υπάρχει σύνδεσμος μεταξύ τους.
METRHTHS ← 0
lp ← p(k-u)
Για  Κάθε i = 1,N
Q ( i ) ← +∞
D ( i ) ← +∞
Τέλος Για Κάθε
Q ( lp ) ← 0
D ( lp ) ← 0
Όσο (METRHTHS ≠ s)
Για Κάθε j=1,N
  Εάν  j ≠ lp & D ( j ) > D ( lp ) + w( lp )( j )
  D ( j ) ←D( lp ) + w( lp )( j )
  Q ( j ) ← D ( j )
  Τέλος Εάν
Τέλος Για Κάθε
ShortestDistance ← 1:
Για  Κάθε i = 1,N
 Εάν  Q( i ) ≠ 0 & D ( i ) < ShortestDistance
```

```
  ShortestDistance ← D( i )
  lp←i
  Τέλος Εάν
Τέλος Για Κάθε
Εάν  lp = p( k-u )   ή   lp = p( k )
Πήγαινε στη Συνέχεια
Τέλος Εάν
Q( lp ) ← 0
METRHTHS ← METRHTHS+1
Τέλος  Όσο
Συνέχεια :
w( p(k-u) )( p(k-u+1) ) ← H   // Επανατοποθετείται ο σύνδεσμος που αφαιρέθηκε
// για τη συνέχεια των επαναλήψεων
A ← D ( p(k) ) - D ( p(k-u) )
B ← w ( p(k-u) )( p(k-u+1) ) + V
V ← min ( A,B )   //Επιλέγεται η μικρότερη ποσότητα από τη συντομότερη διαδρομή
//από τον κόμβο i στον κόμβο s θεωρώντας w( i )( i+1 ) ← +∞ ή από τον κόμβο i+1
//στον κόμβο s συν το βάρος w( i )( i+1 ), όπου οι κόμβοι i και i+1 ανήκουν στην
//ελάχιστη διαδρομή.
Τέλος  Όσο
```

5.6.2.Ο ΑΛΓΟΡΙΘΜΟΣ DREIFUS

Ο Dreyfus (1969) πρότεινε έναν αλγόριθμο με εφαρμογή στο πρόβλημα εύρεσης των k διαδρομών ελαχίστου κόστους από όλους τους κόμβους ενός δικτύου προς έναν κόμβο προορισμού s. Ο αλγόριθμος αυτός βασίστηκε στον αντίστοιχο των R.Bellman and R.Kalaba και είχε βελτιωμένο χρόνο εκτέλεσης. Ο χρόνος αυτός είναι $O(\mathrm{K} \times N^2)$, όπου K ο αριθμός των ελάχιστων διαδρομών που υπολογίζονται. Προχωρώντας στην ανάλυσή του παρουσιάζεται αρχικά το δίκτυο στο οποίο αναφέρεται, το οποίο αποτελείται από N κόμβους και E συνδέσμους. Κατά το πρώτο στάδιο μπορεί να βρεθεί ποιά είναι η ελάχιστη απόσταση από κάθε κόμβο προς τον τελικό κόμβο προορισμού s και να αποθηκευτεί σε έναν πίνακα. Αυτό μπορεί να γίνει μέσω διάφορων αλγορίθμων που έχουν αναπτυχθεί ήδη, ένας από τους οποίους είναι και ο αλγόριθμος των Bellman and Kalaba (1960). Έτσι στο δίκτυο του παρακάτω σχήματος προκύπτει :



Σχήμα 5.11.Ελάχιστες αποστάσεις όλων των κόμβων από τον τελικό κόμβο

Κάθε κόμβος έχει μία τιμή που δείχνει την ελάχιστη απόσταση από αυτόν προς τον τελικό κόμβο s, για παράδειγμα D(1)←55,…., D(s=12)←0 κ.ο.κ.
Αν w(i)(j) το βάρος του συνδέσμου μεταξύ των κόμβων i, j, τότε η 2^{η} συντομότερη διαδρομή από τον τελικό κόμβο s στον τελικό κόμβο s ισούται με :
V(s) ← + ∞
Αρχικά δηλαδή θεωρείται ότι η 2^{η} συντομότερη διαδρομή από τον τελικό κόμβο s στον τελικό κόμβο s έχει κόστος ίσο με + ∞ και γίνεται η επανάληψη :
Για Κάθε k = 1,N
Εάν V(s) < w(s)(k) + D(k)
V(s) ← w(s)(k) + D(k)
Τέλος Εάν
Τέλος Για Κάθε
Από τα παραπάνω προκύπτει ότι αν υπάρχει σύνδεσμος με κατεύθυνση από τον τελικό κόμβο s προς έναν άλλο κόμβο k, τότε το κόστος της $2^{\eta\varsigma}$ διαδρομής ελαχίστου κόστους από το s στο s είναι το V(s) ← w(s)(k) + D(k), αλλιώς τέτοια διαδρομή δεν υπάρχει και V(s) ← + ∞ .

Η 2^{η} διαδρομή ελαχίστου κόστους από κάθε κόμβο iϵ{1,N-1} προς τον τελικό κόμβο s έχει κόστος που προκύπτει από τη διαδικασία :

```
Για Κάθε i = N-1,1   //Ξεκινώντας από τον κόμβο N-1 και προχωρώντας προς τον 1
V( i ) ← + ∞   //Αρχικά θεωρείται ότι δεν υπάρχει 2η συντομότερη διαδρομή από τον
//κόμβο i προς τον s
  Για Κάθε j = 1,N{
     Εάν V( i ) < w( i )( j ) + D( j )
     SecondShortestPath ← V( i )
     V( i ) ← w( i )( j ) + D( j )
```

Τέλος Εάν
Τέλος Για Κάθε
V(i) ← SecondShortestPath
Εάν V(i) > w(i)(z) + V(z)
V(i) ← w(i)(z) + V(z)
Τέλος Εάν

Από τα παραπάνω είναι φανερό ότι το κόστος της $2^{η}$ ελάχιστης διαδρομής από τον κόμβο i στον κόμβο s : V(i) προκύπτει ως

α) Η $2^{η}$ μικρότερη ποσότητα του αθροίσματος w(i)(j) + D(j) για ∀ j∈{1,N}, όπου D(j) το ελάχιστο κόστος για τη διαδρομή j→s ή προκύπτει ως

β) Το άθροισμα w(i)(z) + V(z), όπου V(z) το κόστος της δεύτερης διαδρομής ελαχίστου κόστους από το z→s και z ο κόμβος που βρίσκεται έπειτα από τον i στη διαδρομή ελαχίστου κόστους από το i στο s.

Έτσι το κόστος V(i) είναι αυτό που έχει τη μικρότερη τιμή από τα (α),(β).

Τέλος Για Κάθε

Εάν W(i) το κόστος της $3^{ης}$ διαδρομής ελαχίστου κόστους από κάθε κόμβο i στον κόμβο s, τότε :

Εάν ο κόμβος z είναι ο επόμενος κόμβος του i και κατά την $1^{η}$ και κατά τη $2^{η}$ διαδρομή ελαχίστου κόστους, τότε :

Για Κάθε i = N-1,1
W(i) ← + ∞
Για Κάθε j = 1,N{
Εάν W(i) < w(i)(j) + D(j)
SecondShortestPath ← W(i)
W(i) ← Weight(i)(j) + D(j)
Τέλος Εάν
Τέλος Για Κάθε
W(i) ← SecondShortestPath
Εάν W(i) > w(i)(z) + W(z)
W(i) ← w(i)(z) + W(z)
Τέλος Εάν

Εάν ο κόμβος z είναι ο επόμενος κόμβος του i κατά την $1^{η}$ και ο κόμβος m είναι ο επόμενος κόμβος του i κατά τη $2^{η}$ διαδρομή ελαχίστου κόστους, τότε :

Για Κάθε i = N-1,1
W(i) ← + ∞
Για Κάθε j = 1,N
Εάν W(i) < w(i)(j) + D(j)
ThirdShortestPath ← SecondShortestPath
SecondShortestPath ← W(i)
W(i) ← w (i)(j) + u(j)
Τέλος Εάν
Τέλος Για Κάθε
W(i) ← ThirdShortestPath

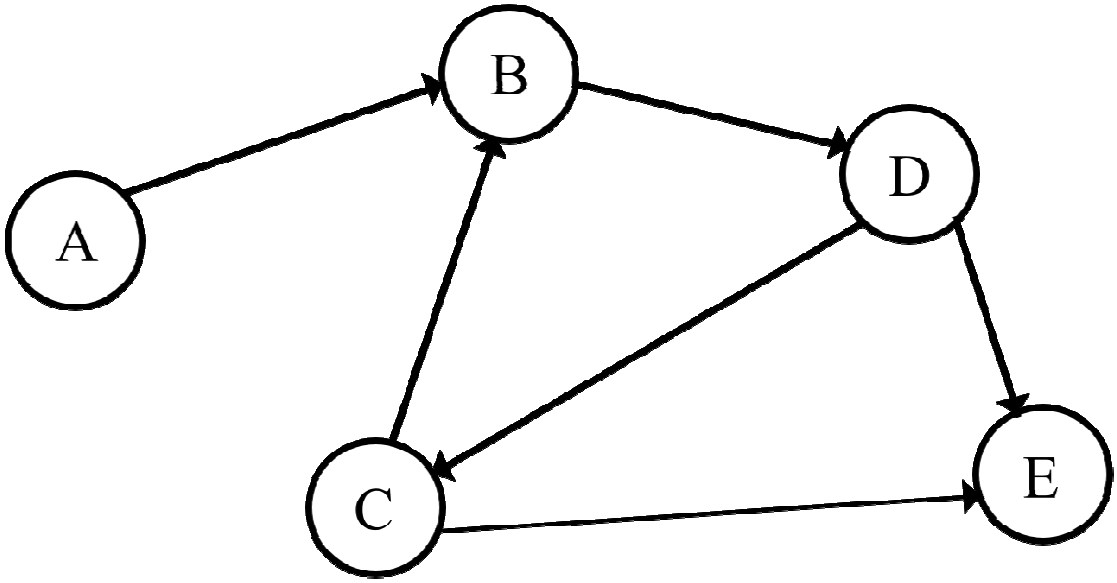
Εάν W(i) > w(i)(z) + V(z)
W(i) ← w(i)(z) + V(z){
Εάν W(i) > w(i)(m) + V(m)
Τέλος Εάν
Τέλος Για Κάθε

Αυτή η διαδικασία μπορεί να γενικευτεί για την εύρεση των k πρώτων διαδρομών ελαχίστου κόστους από κάθε κόμβο i προς τον τελικό κόμβο s. Η διαδικασία που ακολουθήθηκε εάν ο κόμβος z είναι ο επόμενος κόμβος του i και κατά την $1^{η}$ και κατά τη $2^{η}$ διαδρομή ελαχίστου κόστους μπορεί να επαναληφθεί και κατά τις k διαδρομές ελαχίστου κόστους, εφόσον όλες κατευθύνονται από τον κόμβο i σε επόμενο κοινό κόμβο z.

5.6.3.Ο ΑΛΓΟΡΙΘΜΟΣ YEN

Στις δύο προηγούμενες ενότητες παρουσιάστηκε ένας ευρετικός αλγόριθμος για τον υπολογισμό της $2^{ης}$ διαδρομής ελαχίστου κόστους μεταξύ δύο κόμβων r, s και ένας αλγόρθιμος για τον υπολογισμό των k ελάχιστων διαδρομών από κάθε κόμβο του δικτύου προς έναν κόμβο προορισμού s. Στην ενότητα αυτή γίνεται αναφορά στις μεθόδους προσδιορισμού των k συντομότερων διαδρομών μεταξύ δύο κόμβων r, s. Μία από τις σπουδαιότερες μεθόδους αυτής της κατηγορίας προτάθηκε από τον Yen (1971) και χρησιμοποιείται για την εύρεση των k συντομότερων διαδρομών, οι οποίες δεν περιέχουν κυκλικούς βρόγχους (loopless paths). Αμέσως μετά την αρχική παρουσίαση της μεθόδου ο E.L.Lawler (1972) πρότεινε κάποιες βελτιώσεις στη μορφή της. Οι βελτιώσεις αυτές όμως δεν κατάφεραν να μειώσουν το χρόνο εκτέλεσής της όταν επικρατούν δυσμενείς συνθήκες στο δίκτυο.

Γενικότερα ο χρόνος εκτέλεσής της δεν έχει βελτιωθεί αισθητά μέχρι και σήμερα και είναι $O(k \times N(E+N(N \times \log N)))$. Όπως αναφέρθηκε ήδη υπολογίζει διαδρομές που δεν περιέχουν κυκλικούς βρόγχους. Για το λόγο αυτό σε καμία διαδρομή δε γίνεται προσπέλαση του ίδιου κόμβου περισσότερες από μία φορά. Μία διαδρομή που περιέχει κυκλικό βρόγχο παρουσιάζεται στο ακόλουθο σχήμα :



Σχήμα 5.12.Διαδρομή που περιέχει κυκλικό βρόγχο

Για την παρουσίαση του αλγορίθμου Yen, θεωρείται συγκοινωνιακό δίκτυο N κόμβων, όπου (1) ο κόμβος προέλευσης της μετακίνησης και (N) ο κόμβος προορισμού της. Επίσης θεωρείται w(i,j)ϵR^+ το βάρος του συνδέσμου i→j.

A^k = (1)-(2^k)-(3^k)-….-(Q^k)-(N) είναι η k$^{\text{οστή}}$ συντομότερη διαδρομή για τη μετακίνηση (1)→(N), όπου (2^k), (3^k),….,(Q^k) ο $2^{ος}$, ο $3^{ος}$ και ο $Q^{οστός}$ κόμβος της διαδρομής αυτής.

A_i^k , i=1,2,..,Q είναι ένα σύνολο διαδρομών που ¨διαχωρίζονται¨ από τη διαδρομή A^{k-1} στον κόμβο i. Οι διαδρομές αυτές είναι οι συντομότερες διαδρομές που αποτελούνται από τους ίδιους κόμβους με τη διαδρομή A^{k-1} μέχρι τον κόμβο (i) και στη συνέχεια διαφοροποιούνται ακολουθώντας ένα νέο κόμβο (i+1). Αυτός ο κόμβος είναι διαφορετικός από όλους τους κόμβους (i+1) των διαδρομών A^j, j=1,2,…,k-1 οι οποίες αποτελούνται από τους ίδιους κόμβους (1)-..-(i) με τη διαδρομή A^{k-1}. Τέλος καμία συντομότερη διαδρομή A_i^k , i=1,2,..,Q δε μπορεί να περιέχει δύο φορές τον ίδιο κόμβο και για το λόγο αυτό καλούνται διαδρομές που δεν περιέχουν κυκλικό βρόγχο.

R_i^k είναι η ρίζα κάθε διαδρομής A_i^k η οποία συμπίπτει με τη διαδρομή A^{k-1} μέχρι τον κόμβο (i) και S_i^k είναι το πέρας κάθε διαδρομής A_i^k που έχει κοινό κόμβο με τη διαδρομή A^{k-1} τον (i) και καταλήγει στον κόμβο (N).

Κατά την αρχικοποίηση ο αλγόριθμος υπολογίζει τη συντομότερη διαδρομή A^1 μέσω ενός αλγόριθμου υπολογισμού της συντομότερης διαδρομής μεταξύ δύο κόμβων. Θεωρούνται επίσης δύο λίστες. Η λίστα Α στην οποία περιέχονται σε σειρά προτεραιότητας οι k συντομότερες διαδρομές και η λίστα Β στην οποία περιέχονται οι υποψήφιες συντομότερες διαδρομές. Σε κάθε επανάληψη η συντομότερη διαδρομή από τη λίστα Β μεταφέρεται στη λίστα Α, μέχρι η λίστα Α να διαθέτει τις εναλλακτικές διαδρομές που απαιτούνται. Αρχικά στη λίστα Α περιέχεται η διαδρομή A^1. Κατά την k$^{\text{οστή}}$ επανάληψη του αλγορίθμου υπάρχουν στη λίστα Α οι διαδρομές

A^1, A^2, …, A^{k-1} και πρέπει να επιλεγεί η $k^{οστή}$ συντομότερη διαδρομή από τη λίστα Β. Ταυτόχρονα αυξάνεται ο αριθμός των διαδρομών που περιέχονται στη λίστα Β σύμφωνα με την ακόλουθη διαδικασία.

$k^{οστή}$ ΕΠΑΝΑΛΗΨΗ ΑΛΓΟΡΙΘΜΟΥ ΥΕΝ

A^{k-1} = (1)-(2^k)-(3^k)-….-(Q^k)-(N), οι κόμβοι από τους οποίους αποτελείται η διαδρομή A^{k-1}.

Για Κάθε (i) = (1), (2^k), (3^k),…., (Q^k), (N)

α) Έλεγξε ποιές διαδρομές A^j, j = 1,2,…,k-1 αποτελούνται από τους ίδιους κόμβους με τη διαδρομή A^{k-1} μέχρι και τον κόμβο (i). Τότε, για όσες από αυτές διαχωρίζονται από την A^{k-1} ακολουθώντας κάποιον άλλο κόμβο (i+1), θέσε w(i,i+1) ← +∞ . Ομοίως για τη διαδρομή A^{k-1} : w(i,i+1) ← +∞ .

β) Χρησιμοποίησε τον αλγόριθμο Dijkstra για την εύρεση της συντομότερης διαδρομής από τον κόμβο (i) στον κόμβο (N). Έτσι προκύπτει το πέρας της διαδρομής : S_i^k.

γ) Προσδιόρισε τη διαδρομή A_i^k ως : $A_i^k \leftarrow R_i^k + S_i^k$ και στη συνέχεια τοποθέτησέ τη στη λίστα Β.

δ) Επανέφερε τις αρχικές τιμές στα βάρη συνδέσμων που τροποποιήθηκαν κατά το πρώτο βήμα.

Τέλος Για Κάθε

Από όλες τις διαδρομές που υπάρχουν στη λίστα Β επέλεξε τη συντομότερη διαδρομή και μετέφερέ τη στη λίστα Α. Αυτή είναι πλέον η $k^{οστή}$ συντομότερη διαδρομή.

Εάν στη λίστα Α περιέχονται λιγότερες εναλλακτικές διαδρομές από τις ζητούμενες επέστρεψε στην αρχή και επανέλαβε τη διαδικασία για k←k+1.

5.6.4.ΕΝΑΛΛΑΚΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΠΟΥ ΒΑΣΙΖΕΤΑΙ ΣΤΗ ΜΕΘΟΔΟ ΤΟΥ EPPSTEIN

Σε αυτή την ενότητα προτείνεται ένας αλγόριθμος με εφαρμογή στο πρόβλημα εύρεσης των k συντομότερων διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού που αποτελεί παραλλαγή της μεθόδου που προτάθηκε από τον D.Eppstein (1997) με χρόνο εκτέλεσης O(E+NlogN+kN).

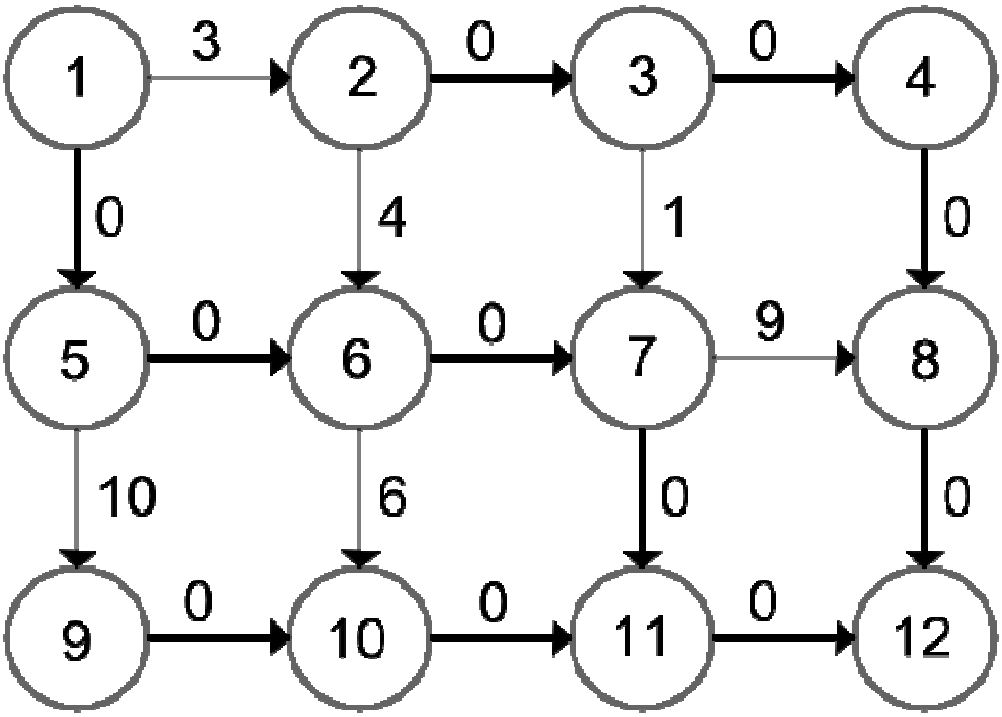
Έστω συγκοινωνιακό δίκτυο G={N,E} με κόμβους προέλευσης-προορισμού r, s. Κάθε διαδρομή p^i μεταξύ αυτών των κόμβων έχει κόστος μετακίνησης :
$w(p^i) = \sum_{e \epsilon p^i} w(\ e\ )$.

Αρχικά επιλύεται το πρόβλημα εύρεσης των συντομότερων διαδρομών από όλους τους κόμβους προς τον κόμβο προορισμού. Το κόστος μετακίνησης της διαδρομής ελαχίστου κόστους r→s είναι : D(r). Για την αναπαράσταση μίας διαδρομής μεταξύ των κόμβων r, s δεν παρουσιάζονται όλοι οι σύνδεσμοι που ανήκουν σε αυτήν τη

διαδρομή αλλά οι σύνδεσμοι που διαφέρουν από αυτούς της διαδρομής ελαχίστου κόστους. Έτσι η διαδρομή ελαχίστου κόστους p: (r→s) μπορεί να παρασταθεί ως το κενό σύνολο {}. Στο δίκτυο του Σχήματος 5.11. θεωρείται ένα δένδρο που περιέχει τους συνδέσμους που ανήκουν σε διαδρομές ελαχίστου κόστους από κάθε κόμβο iϵN προς τον κόμβο προορισμού s. Το ελάχιστο κόστος μετακίνησης από κάθε κόμβο iϵN προς τον κόμβο s μπορεί να υπολογιστεί μέσω του αλγορίθμου Dijkstra ή των Bellman-Kalaba, από την εφαρμογή των οποίων προκύπτουν και τα αποτελέσματα του Σχήματος 5.11. Μέσω της εφαρμογής αυτών των αλγορίθμων προκύπτει και ο επόμενος κόμβος κάθε κόμβου iϵN, ώστε ο κόμβος αυτός να ανήκει σε μία διαδρομή ελαχίστου κόστους. Έτσι για κάθε κόμβο iϵN αποθηκεύεται σε μία μήτρα : next(i) ο επόμενος κόμβος του i. Έστω ποσότητα δ(e) που δείχνει πόσο επιβαρύνει ο σύνδεσμος e μία διαδρομή ελαχίστου κόστους, τότε για όλους τους συνδέσμους που ανήκουν στο δένδρο T (σύνδεσμοι διαδρομών ελαχίστου κόστους) ισχύει δ(e) = 0.

Γενικά, το κατά πόσο επιβαρύνει ένας σύνδεσμος μία διαδρομή ελαχίστου κόστους δίνεται από τη σχέση : δ(e) = w(e) + D(E(e)) – D(S(e)), ∀e ϵ E , όπου D(E(e)) το ελάχιστο κόστος μετακίνησης από τον κόμβο E(e) στον κόμβο s.

Έτσι εάν ένας σύνδεσμος έχει δ(e) = 0, τότε ανήκει στο δένδρο T, αλλιώς ανήκει στο δένδρο G-T, στο οποίο ανήκουν οι σύνδεσμοι που δεν ανήκουν σε καμία διαδρομή ελαχίστου κόστους. Σύμφωνα με τα παραπάνω το Σχήμα 5.11. τροποποιείται στο ακόλουθο σχήμα, όπου για κάθε σύνδεσμο δίνεται η ποσότητα δ(e).



Σχήμα 5.13.Σύνδεμοι που ανήκουν στο δένδρο T : δ(e) = 0 και σύνδεμοι που ανήκουν στο δένδρο G-T : δ(e) ≠ 0

Σύμφωνα με το Σχήμα 5.11. η διαδρομή ελαχίστου κόστους από το r στο s, όπου r = 1 και s = 12 έχει κόστος : D(r) = 55. Η διαδρομή αυτή συμβολίζεται με το κενό σύνολο {}, αφού όλοι οι σύνδεσμοί της ανήκουν στη διαδρομή ελαχίστου κόστους

r→s. Κάθε εναλλακτική διαδρομή από το r στο s θα διέρχεται από ένα σύνδεσμο με δ(e) ≠ 0. Έτσι το βάρος κάθε εναλλακτικής διαδρομής p^i από το r στο s ισούται με : w(p^i) = D(r) + $\sum_{e \epsilon p^i} \delta(e)$

Ένα από τα πλεονεκτήματα αυτής της διαδικασίας είναι η ευκολία με την οποία προκύπτει η $2^η$ διαδρομή ελαχίστου κόστους από τον κόμβο r στον κόμβο s, η οποία απαιτούσε πολλές επαναλήψεις του αλγορίθμου Dijkstra σύμφωνα με τη μέθοδο των Hoffman-Pavley. Πιο αναλυτικά με αρχή τον κόμβο r = 1 καταγράφεται μία τιμή A = δ(e), όπου δ(e) ≠ 0 και S(e) = r. Έτσι σύμφωνα με το Σχήμα 5.13. προκύπτει A = 3. Στη συνέχεια για τον επόμενο κόμβο του r ο οποίος ανήκει στη διαδρομή ελαχίστου κόστους : next(r) = 5 επαναλαμβάνεται η παραπάνω διαδικασία και προκύπτει ότι δ(e) = 10 ώστε δ(e) ≠ 0 και S(e) = 5. Επειδή A < 10 η τιμή της σταθεράς A δεν αλλάζει τιμή. Συνεχίζοντας για next(5) = 6 προκύπτει δ(6) = 6 και A < 6 και αυτή η διαδικασία συνεχίζεται μέχρι next(i) = s, όπου s = 12. Έτσι προκύπτει ότι A = 3, άρα η $2^η$ διαδρομή ελαχίστου κόστους πρέπει να διέρχεται από το σύνδεσμο με δ(e) = 3 και αποτελείται από τους κόμβους {1,2,3,4,8,12}. Το κόστος της είναι : w(p^2) = D(r) + $\sum_{e \epsilon p^2} \delta(e)$ = 55 + 3 = 58. Αριθμώντας τους οριζόντιους συνδέσμους του δικτύου ως {1,2,3,8,9,10,15,16,17} και τους κατακόρυφους συνδέσμους ως {4,5,6,7,11,12,13,14} η διαδρομή p^2 συμβολίζεται ως {1}.

Για τον υπολογισμό των k διαδρομών ελαχίστου κόστους από τον κόμβο r στον κόμβο s μπορεί να χρησιμοποιηθεί η παρακάτω μέθοδος :

ΠΑΡΑΛΛΑΓΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ EPPSTEIN

Έστω μήτρα Storage που αποθηκεύει το σύνδεσμο από τον οποίο διαφοροποιείται κάθε διαδρομή από τις διαδρομές ελαχίστου κόστους. Έστω μήτρα W που αποθηκεύει το κόστος κάθε εναλλακτικής διαδρομής. Αν μία διαδρομή διαφοροποιείται από τις διαδρομές ελαχίστου κόστους στο σύνδεσμο x_q αλλά έχει ήδη διαφοροποιηθεί από τις διαδρομές ελαχίστου κόστους άλλες z φορές, όπου x_{q-1} ο τελευταίος σύνδεσμος που διαφοροποιήθηκε πριν το x_q, τότε ο σύνδεσμος αυτός αποθηκεύεται σε μία μήτρα : Parent(x_q) = x_{q-1} , για να προκύψει δομή δένδρου.

k ← 1

u ← r

λ ← r

Storage(λ) = ∅ // η διαδρομή ελαχίστου κόστους δεν έχει σύνδεσμο με δ(e) ≠ 0

k ← 2

Βήμα 1(Εύρεση Διαδρομών από το r στο s που διαφοροποιούνται κατά ένα σύνδεσμο με δ(e) ≠ 0 από τις διαδρομές ελαχίστου κόστους)

Όσο u ≠ s

Για Κάθε e = 1,E

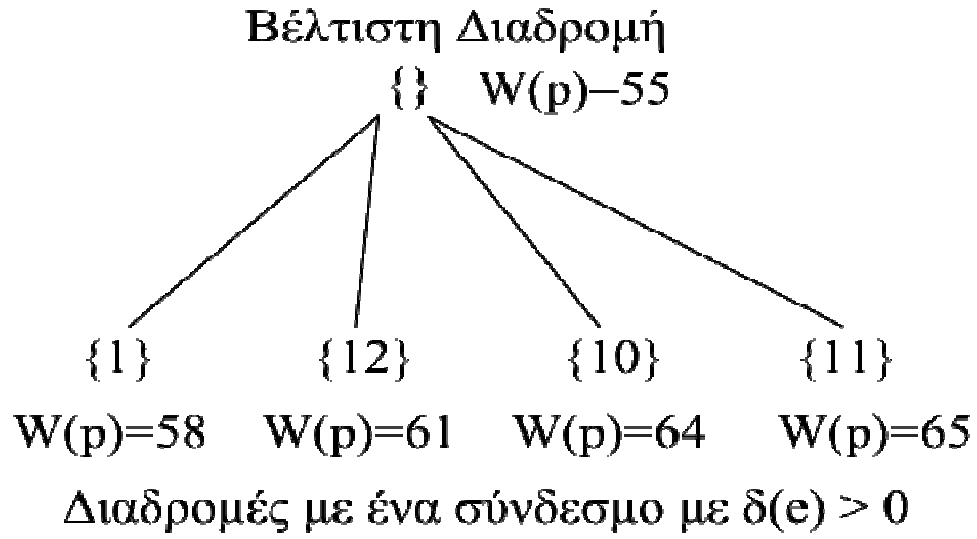
Εάν S(e) = u & δ(e) ≠ 0

W(k) ← D(r) + δ(e) // Το κόστος της διαδρομής k
Parent(k) = Storage(λ)
Storage(k) = e
k ← k+1
Τέλος Εάν
Τέλος Για Κάθε
u ← next(u)
Τέλος Όσο

Βήμα 2(Εύρεση Διαδρομών από το r στο s που διαφοροποιούνται κατά δύο συνδέσμους από τις διαδρομές ελαχίστου κόστους)

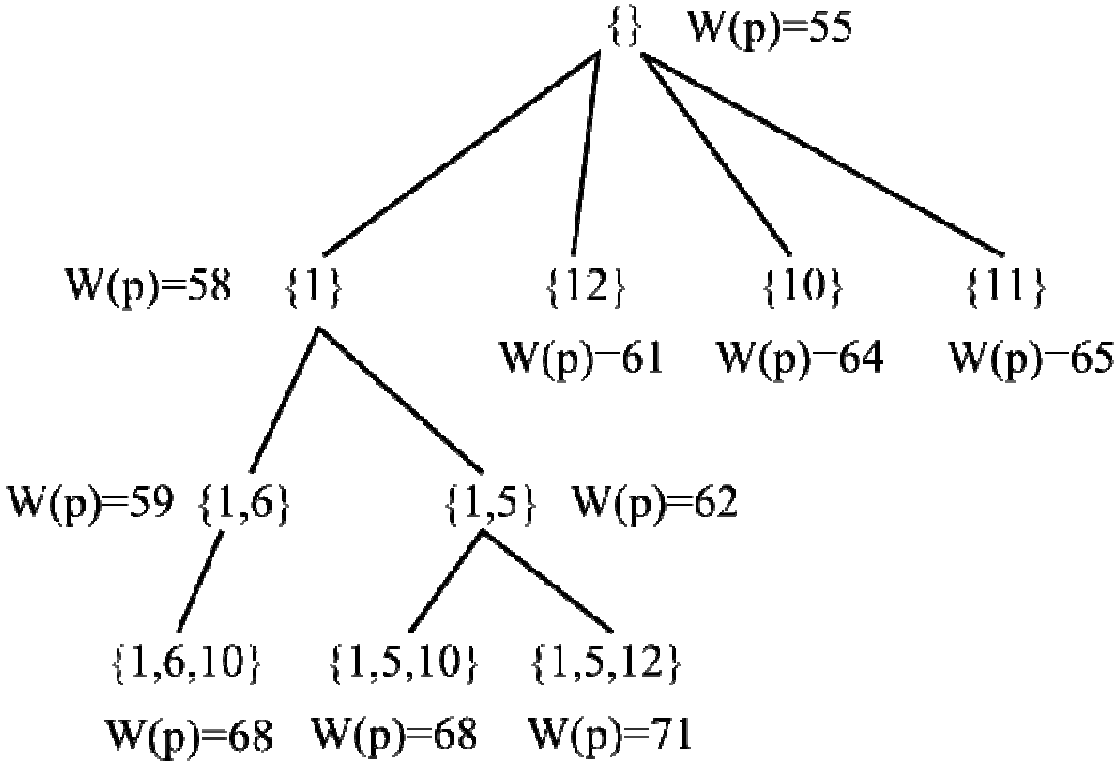
t ← 1
Για Κάθε i = 2,k
u ← S(Storage(k))
Όσο u ≠ s
Για Κάθε e = 1,E
Εάν S(e) = u & δ(e) ≠ 0
W(t) ← W(k) + δ(e) // Το κόστος της διαδρομής t
Parent(t) = Storage(k)
Storage(t) = e
t ← t+1
Τέλος Εάν
Τέλος Για Κάθε
u ← next(u)
Τέλος Όσο
Τέλος Για Κάθε
Η διαδικασία επαναλαμβάνεται και για τις εναλλακτικές διαδρομές που αποτελούνται από 3, 4, κ.ο.κ. συνδέσμους του δένδρου G-T

Σύμφωνα με το **Βήμα 1** του παραπάνω αλγορίθμου, κατά την εφαρμογή του στο δίκτυο του Σχήματος 5.13. προκύπτει το ακόλουθο δένδρο :



Σχήμα 5.14.Διαδρομές Ελάχιστου Κόστους μετά το 1^{ο} Βήμα επανάληψης του αλγορίθμου

Έτσι συνεχίζοντας την επανάληψη προκύπτουν όλες οι εναλλακτικές διαδρομές από τον κόμβο r στον κόμβο s όπως παρουσιάζεται στο ακόλουθο σχήμα :



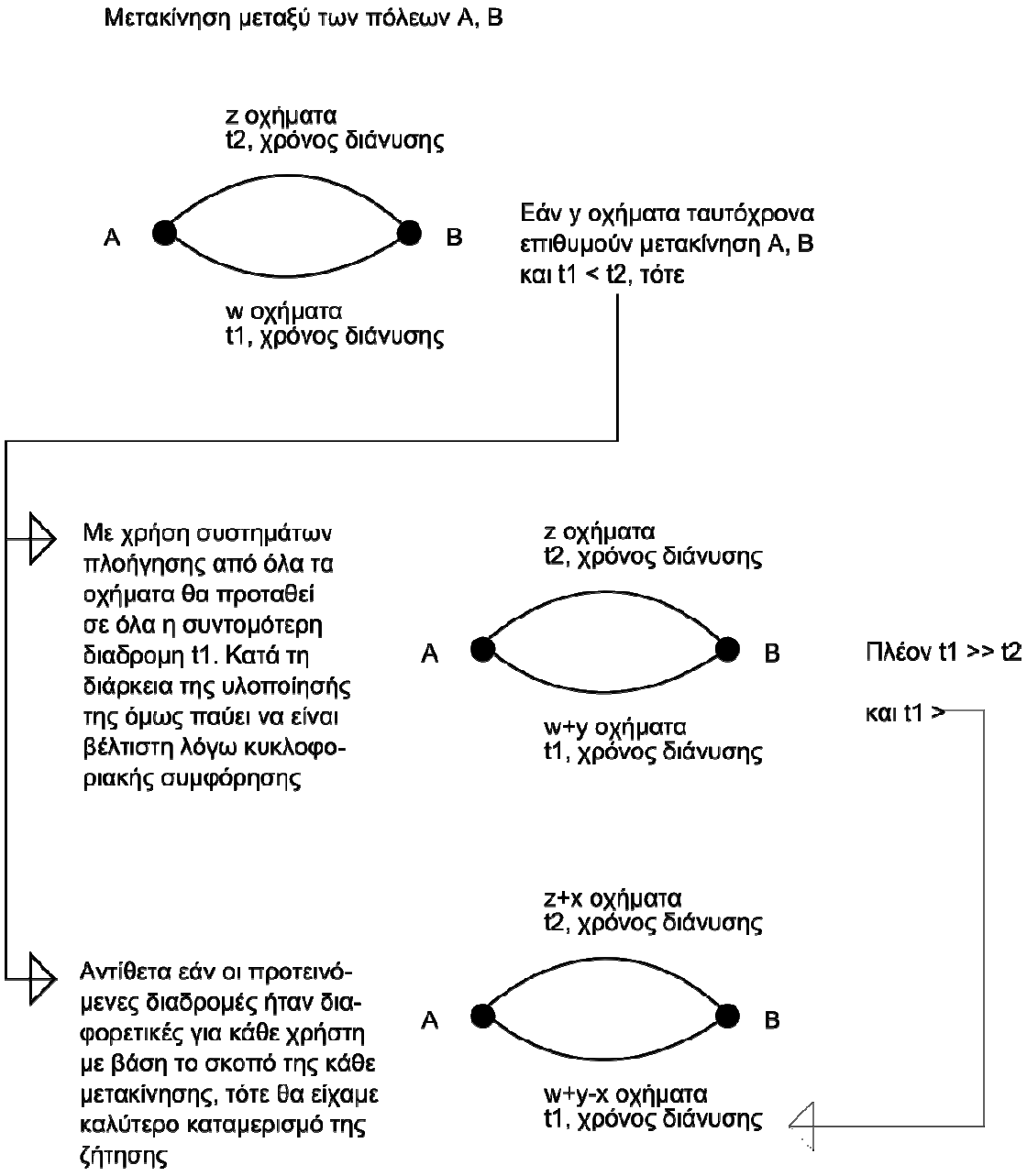
Σχήμα 5.15.Σύνολο Εναλλακτικών διαδρομών από το r στο s

5.7.ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ ΣΥΝΤΟΜΟΤΕΡΗΣ ΔΙΑΔΡΟΜΗΣ ΠΟΥ ΙΚΑΝΟΠΟΙΕΙ ΔΙΑΦΟΡΟΥΣ ΠΕΡΙΟΡΙΣΜΟΥΣ

Τα προβλήματα εύρεσης των συντομότερων διαδρομών έχουν μελετηθεί αρκετά μέχρι σήμερα. Πολλοί από τους αλγόριθμους που έχουν αναπτυχθεί χρησιμοποιούνται από τα συστήματα πλοήγησης με σκοπό την καθοδήγηση του χρήστη. Ωστόσο, πρέπει να ληφθούν υπόψη οι συνέπειες αυτής της καθοδήγησης όταν το ποσοστό των χρηστών που διαθέτει συστήματα πλοήγησης αυξηθεί. Σε αυτή την περίπτωση η χρήση των παραπάνω αλγορίθμων επηρεάζει ευθέως τον καταμερισμό της ζήτησης. Σύμφωνα με τη σημερινή τεχνολογία εάν όλοι οι χρήστες διέθεταν συστήματα πλοήγησης και επιθυμούσαν ταυτόχρονα την ίδια μετακίνηση θα προτεινόταν σε όλους η ίδια διαδρομή. Αυτό βέβαια έχει ως αποτέλεσμα η διαδρομή αυτή να μην αποδεικνύεται βέλτιστη στην πορεία λόγω κυκλοφοριακής συμφόρησης. Στο επόμενο κεφάλαιο γίνεται ιδιαίτερη αναφορά σε αυτή την περίπτωση και το αν και υπό ποιές προϋποθέσεις μπορούν να χρησιμοποιηθούν τα συστήματα πλοήγησης ώστε να βελτιωθεί η αποδοτικότητα του δικτύου, κάτι που μέχρι σήμερα αποτελεί αντικείμενο έρευνας.

Στην ενότητα αυτή μελετάται η αντιμετώπιση κάθε μετακίνησης ξεχωριστά ώστε να προτείνεται σε κάθε χρήστη άλλη διαδρομή αποφεύγοντας το πρόβλημα της παραπάνω περίπτωσης. Αυτό μπορεί να συμβεί εάν οι εναλλακτικές διαδρομές δε βελτιστοποιούνται ως προς ένα μέγεθος, όπως ο χρόνος ταξιδιού, κάτι που αναγκάζει όλους τους χρήστες να ακολουθήσουν ίδιες διαδρομές. Αντίθετα μπορούν να εξετάζονται διάφορες παράμετροι και για κάθε χρήστη να προτείνεται διαφορετική διαδρομή σύμφωνα με το σκοπό της μετακίνησής του. Έτσι κάθε σύνδεσμος ενός δικτύου δε θα έχει πλέον κάποιο μονοσήμαντα ορισμένο βάρος(π.χ., χρόνο διάνυσης). Αντίθετα αυτό θα προκύπτει από μία συνάρτηση w(e) που εξαρτάται από διάφορες παραμέτρους με συγκεκριμένη βαρύτητα (π.χ. εκπομπές CO_2, ποιότητα οδοστρώματος, χρόνος διάνυσης, αριθμός διασταυρώσεων, διέλευση από διόδια κ.α.). Ο χρήστης μπορεί να επιλέξει έτσι τι είναι σημαντικότερο γι'αυτόν και αυτό μεταφράζεται σε αλλαγή των συντελεστών βαρύτητας της συνάρτησης w(e). Οι διαδρομές που προκύπτουν έτσι δεν είναι ίδιες για όλους αλλά έχουν το χαρακτήρα του προσωπικού βέλτιστου (personal optimum) με στόχο να ικανοποιήσουν το σκοπό για τον οποίο επιτελείται η μετακίνηση.

Από αυτή τη διαδικασία προκύπτουν πολλά θετικά για τους χρήστες και τα τελευταία χρόνια γίνεται έντονη προσπάθεια ανάπτυξης μεθόδων προς αυτή την κατεύθυνση. Εάν μάλιστα χρησιμοποιηθούν αλγόριθμοι αυτής της μορφής από τα συστήματα πλοήγησης μπορεί ο καταμερισμός της ζήτησης στο δίκτυο να μην επηρεάζεται ακόμα και αν το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης είναι μεγάλο. Για να γίνει κάτι τέτοιο εποπτικά κατανοητό παρατίθεται το ακόλουθο σχήμα.



Σχήμα 5.16.Ταυτόχρονη δρομολόγηση οχημάτων από συστήματα πλοήγησης

Έτσι αν και στην πρώτη περίπτωση τα συστήματα πλοήγησης πρότειναν σε όλους τους χρήστες τη διαδρομή με χρόνο διάνυσης t1 αποδείχθηκε ότι λόγω κυκλοφοριακής συμφόρησης στη συνέχεια (Φόρτος : w+y) προέκυψε t2 < t1. Δηλαδή η συντομότερη διαδρομή ήταν πλέον η διαδρομή με χρόνο διάνυσης t2. Έτσι πολλοί χρήστες δεν ακολούθησαν τη συντομότερη διαδρομή και είναι πιθανό να καθυστέρησαν περισσότερο από την περίπτωση που κανείς δε θα χρησιμοποιούσε σύστημα πλοήγησης και θα επέλεγαν όλοι μεμονωμένα μία τυχαία διαδρομή. Στη δεύτερη περίπτωση κάθε χρήστης επέλεξε διαδρομή η οποία πληροί ορισμένα χαρακτηριστικά που αυτός έχει επιλέξει. Εάν όλοι οι χρήστες ενδιαφέρονται μόνο για το χρόνο της μετακίνησης, τότε προκύπτει η πρώτη περίπτωση. Συνήθως όμως η επιλογή είναι πολυκριτηριακή και η ζήτηση καταμερίζεται τυχαία στο δίκτυο. Αυτή η

μορφή καταμερισμού της ζήτησης δεν είναι βέλτιστη ούτε ως προς το σύστημα, κάτι που θα βελτίωνε την αποδοτικότητα του δικτύου, ούτε ως προς το χρόνο διάνυσης της μετακίνησης. Δε μπορεί όμως να παραβλεφθεί ότι πλησιάζει το βέλτιστο ως προς την επιθυμία του κάθε χρήστη ξεχωριστά, ο οποίος είναι και ο τελικός κριτής του δικτύου.

Οι πρώτες μελέτες προς αυτή την κατεύθυνση δεν εστιάζουν στη δημιουργία συνάρτησης που θα δίνει το βάρος κάθε συνδέσμου w(e). Άλλωστε η βαρύτητα που θα έχει κάθε παράμετρος σε αυτή τη συνάρτηση είναι υποκειμενικό μέγεθος και ακόμα και αν ο χρήστης επιλέξει τις παραμέτρους που τον ενδιαφέρουν είναι δύσκολο να μεταφραστούν σε ένα αριθμητικό μοντέλο. Εάν πάντως κάτι τέτοιο είναι εφικτό και ο χρήστης επιλέξει μία σειρά από χαρακτηριστικά που τον ενδιαφέρουν όπως για παράδειγμα χρόνος διάνυσης, κατανάλωση καυσίμου, αριθμός αριστερών στροφών κ.α. τότε για κάθε σύνδεσμο όλοι οι παράμετροι θα πολλαπλασιαστούν με τους ανάλογους συντελεστές και θα προκύψουν τα βάρη των συνδέσμων. Το δύσκολο σε αυτήν τη διαδικασία είναι ο αντικειμενικός υπολογισμός για όλα τα βάρη των συνδέσμων γιατί στη συνέχεια το πρόβλημα ανάγεται σε ένα απλό πρόβλημα υπολογισμού της βέλτιστης διαδρομής μεταξύ δύο κόμβων (Point to Point Shortest Path Problem) για το οποίο έχουν προταθεί οι ανάλογοι αλγόριθμοι. Σύγχρονες έρευνες όμως εστιάζουν στην επίλυση ενός άλλου είδους προβλήματος. Το πρόβλημα αυτό γνωστό ως πρόβλημα εύρεσης της συντομότερης διαδρομής σε δίκτυα με περιορισμούς διαθέσιμων πόρων (Resource Constrained Shortest Path Problem), έχει μελετηθεί σε διάφορες εργασίες όπως οι εργασίες των Handler and Zang (1980), Hartel and Glaser (1996), Desaulniers et al. (1998), Avela et al. (2004) και έχει την εξής μορφή :

Σε δίκτυο G = {N,E} το οποίο διαθέτει συνδέσμους e∈E πρέπει να ευρεθεί η συντομότερη διαδρομή μεταξύ δύο κόμβων προέλευσης-προορισμού r, s. Η διαδρομή αυτή (p) μπορεί να βρεθεί μέσω ενός (SSSP) αλγορίθμου, όπως ο αλγόριθμος Dijkstra ώστε : D(p) = Minimum, όπου η διαδρομή p αποτελείται από τους συνδέσμους $\{e_x, e_{y,\ldots\ldots\ldots,} e_z\}$,e∈E με S(e_x) = r , E(e_z) = s και

$$D(p) = \sum_{e \in p} w(e)$$

Επίσης κάθε σύνδεσμος e έχει μία σειρά από τιμές παραμέτρων $r_e = \{R_1, R_2, \ldots., R_n\}^T$, οι οποίες μπορεί να είναι η κατανάλωση βενζίνης, η ποιότητα οδοστρώματος κ.α. Για κάθε παράμετρο R_i υπάρχει μία ακραία τιμή b_i η οποία δεν πρέπει να παραβιάζεται από το άθροισμα των τιμών των συνδέσμων που ανήκουν στη συντομότερη διαδρομή :

$$\sum_{e \in p} (R_e)_i \leq b_i \ , \forall \ i \in \{1,n\}$$

Εάν ο περιορισμός αυτός παραβιάζεται, τότε υπολογίζεται η δεύτερη συντομότερη διαδρομή και πραγματοποιείται ο παραπάνω έλεγχος, η τρίτη κ.ο.κ. μέχρι να ισχύει ο περιορισμός. Η διαδρομή που επιλέγεται είναι στην ουσία η συντομότερη η οποία ικανοποιεί μία σειρά από περιορισμούς. Οι περιορισμοί αυτοί μπορεί να είναι πολλοί και ο χρήστης να επιλέγει κάθε φορά ποιοί από αυτούς θέλει να ικανοποιούνται. Για

παράδειγμα η διαδρομή να μην περιέχει πάνω από *x* διασταυρώσεις και να μην είναι πάνω από k χιλιόμετρα.

Το πρόβλημα αυτό μπορεί να διατυπωθεί και ως πρόβλημα Γραμμικού Ακέραιου προγραμματισμού ως εξής : Έστω ότι κάθε σύνδεσμος e ϵ E με κατεύθυνση από το i→j μπορεί να ανήκει στη βέλτιστη διαδρομή ή όχι. Τα προβλήματα στα οποία οι εναλλακτικές επιλογές που παρέχονται είναι της μορφής «ναι ή όχι» καλούνται και προβλήματα της μορφής «0 - 1». Αν ο σύνδεσμος ανήκει τελικά στη βέλτιστη διαδρομή τότε θεωρείται $x_{i.j}$=1, ενώ σε αντίθετη περίπτωση $x_{i.j}$=0. Ζητείται η εύρεση της βέλτιστης διαδρομής από τον κόμβο r στον κόμβο s ώστε :

Min $z = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j} x_{i,j}$

όπου $w_{i,j}$ η αντίσταση μετακίνησης από τον κόμβο i→j και

Υπό τους περιορισμούς :

$x_{i,j} \in [0,1]$, $x_{i,j} \in Z^+$

$\sum_{j=1}^{N} x_{r,j} = 1$

$\sum_{j=1}^{N} x_{j,s} = 1$

$\sum_{i=1}^{N} x_{i,j} = \sum_{t=1}^{N} x_{j,t}$, j ϵ N \ {r,s}

} Περιορισμοί Λόγω της Γεωμετρίας του Δικτύου

$\sum_{i=1}^{N} \sum_{j}^{N} R_{i,j,k}\, x_{i,j} \leq b_k$, $\forall$ k $\in$ {1,n}, όπου $R_{i,j,k}$ η ποσότητα κατανάλωσης του k διαθέσιμου πόρου κατά τη μετακίνηση από τον κόμβο i→j και n ο αριθμός των περιορισμών.

Το παραπάνω πρόβλημα μπορεί να επιλυθεί με δυσκολία μέσω των κλασικών μεθόδων που χρησιμοποιούνται για την επίλυση προβλημάτων γραμμικού ακέραιου προγραμματισμού όπως η μέθοδος της απαρίθμησης, που είναι εξαντλητική και δεν ενδείκνυται σε προβλήματα μεγάλου μεγέθους. Αντίθετα περισσότερο χρήσιμη είναι η μέθοδος του κλάδου φράγματος (branch and bound) η οποία είναι συνήθως αποδοτική σε προβλήματα της μορφής «0 - 1».

Για την επίλυσή του έχουν προταθεί διάφορες μέθοδοι στη διεθνή βιβλιογραφία. Αξίζει να σημειωθεί ότι δεν έχει προσδιοριστεί ακριβής αλγόριθμος που να το επιλύει σε πολυωνυμικό χρόνο. Στη συνέχεια αναλύονται τρεις από αυτές τις μεθόδους και στο τέλος προτείνεται μία νέα μέθοδος.

5.7.1.ΧΡΗΣΗ ΜΕΘΟΔΩΝ ΑΠΑΡΙΘΜΗΣΗΣ

Αρχικά μπορεί να χρησιμοποιηθεί ένας αλγόριθμος εύρεσης των k συντομότερων διαδρομών από έναν κόμβο προέλευσης r σε έναν κόμβο προορισμού s, όπως οι αλγόριθμοι που παρουσιάστηκαν στην Ενότητα 5.6. και στη συνέχεια να επιλεγεί η διαδρομή που ικανοποιεί όλους τους περιορισμούς και έχει το μικρότερο κόστος διάνυσης. Κάτι τέτοιο βέβαια απαιτεί πολλές θέσεις μνήμης για την αποθήκευση των διαδρομών καθώς επίσης και σημαντικό χρόνο εκτέλεσης. Ειδικά μέσω συστημάτων πλοήγησης που έχουν μικρή υπολογιστική ισχύ η αποδοτικότητα αυτών των μεθόδων είναι ελεγχόμενη και μπορούν μα εφαρμοστούν μόνο σε δίκτυα περιορισμένου

μεγέθους. Επειδή οι μέθοδοι αυτοί έχουν αναπτυχθεί ήδη σε προηγούμενη ενότητα δε θα πραγματοποιηθεί περαιτέρω διερεύνησή τους.

5.7.2.ΜΕΘΟΔΟΣ ΤΩΝ ΠΟΛΛΑΠΛΑΣΙΑΣΤΩΝ LAGRANGE

Η πρώτη μέθοδος που ανήκει σε αυτή την κατηγορία προτάθηκε από τους Handler and Zang (1980) για προβλήματα με ένα περιορισμό διαθέσιμου πόρου και επεκτάθηκε στη συνέχεια από τους Beasly and Christofides (1989) και Borndörfer et al. (2001). Πριν την ανάπτυξη αυτής της μεθόδου προτείνεται μία διαφορετική μαθηματική διατύπωση του προβλήματος εύρεσης της συντομότερης διαδρομή με περιορισμούς διαθέσιμων πόρων. Πιο συγκεκριμένα, για τη μετακίνηση μεταξύ δύο κόμβων προέλευσης-προορισμού διατίθεται ένα σύνολο εναλλακτικών διαδρομών P(s,t). Κάθε διαδρομή p που ανήκει σε αυτό το σύνολο έχει κόστος : w(p) = $\sum_{e\epsilon p} w_{(e)}$. Οι διαθέσιμοι πόροι που καταναλώνονται κατά τη διάνυση αυτής της διαδρομής είναι $R_p^i = \sum_{e\epsilon p} R_{(e)}^i$ για κάθε i=1,.., n.

Το πρόβλημα διατυπώνεται ως η εύρεση της διαδρομής p^* η οποία ανήκει στο σύνολο P(s,t) και για την οποία ισχύουν :

$w_{p^*} = min_{p\epsilon P(s,t)} \sum_{e\epsilon p} w_{(e)}$ (1)

Υπό τον περιορισμούς

$\sum_{e\epsilon p^*} R_{(e)}^i \leq b^i$ για κάθε διαθέσιμο πόρο i=1, 2,…, n (2)

Στο παραπάνω πρόβλημα οι περιορισμοί του προβλήματος μπορούν να αντικατασταθούν από έναν όρο ο οποίος εισέρχεται στην αντικειμενική συνάρτηση. Οι όροι αυτοί είναι συναρτήσεις που περιλαμβάνουν τους περιορισμούς του προβλήματος επιβαρυμένους με ειδικούς πολλαπλασιαστές, οι οποίοι είναι γνωστοί ως πολλαπλασιαστές Lagrange. Για την κατανόηση της παραπάνω διαδικασίας παρουσιάζεται το ακόλουθο πρόβλημα γραμμικού ακέραιου προγραμματισμού (ILP).

L = Min { c^Tx }
Ax ≤ b
Bx ≤ d , x ⩾ 0 όπου x ϵ Z
Με c, b, d, x μονοδιάστατους πίνακες διαστάσεων : n, m, p, n και A: m × n , B: p × n

Για την αντικατάσταση του περιορισμό Ax ≤ b χρησιμοποιούνται πολλαπλασιαστές Lagrange $u_i \geqslant 0$, i=1, 2,.., m και το πρόβλημα επαναδιατυπώνεται ως :

L(u) = Min { c^Tx - u^T (b-Ax) }
Bx ≤ d , u ⩾ 0 , x ⩾ 0 όπου x ϵ Z

Σύμφωνα με τα παραπάνω η ποσότητα L(u) αποτελεί το κάτω όριο της ποσότητας L για κάθε τιμή των u_i, i=1, 2,.., m. Για τον προσδιορισμό του βέλτιστου κάτω ορίου της συνάρτησης L(u) επιλύεται το ακόλουθο πρόβλημα :

LB = Max { Min { c^Tx - u^T (b-Ax) } }
Υπό τους περιορισμούς Bx ≤ d , u ⩾ 0 , x⩾0 όπου x ϵ Z

Με τον τρόπο αυτό προσδιορίζονται οι βέλτιστοι πολλαπλασιαστές Lagrange. Το άνω όριο (UB) μπορεί να βρεθεί μέσω της διαδικασίας επίλυσης του παραπάνω προβλήματος. Εάν το κάτω όριο και το άνω όριο είναι ίσα, τότε η λύση αυτή είναι η βέλτιστη λύση του αρχικού προβλήματος. Σε διαφορετική περίπτωση υπάρχει κάποιο διάστημα που τα χωρίζει (duality gap) και χρησιμοποιούνται διάφορες μέθοδοι για τον προσδιορισμό της βέλτιστης λύσης εντός αυτού του διαστήματος. Όπως είναι φανερό το κύριο πρόβλημα είναι ο προσδιορισμός των βέλτιστων πολλαπλασιαστών Lagrange. Για τον προσδιορισμό αυτών των συντελεστών έχουν χρησιμοποιηθεί διάφορες μέθοδοι. Στην εργασία των Beasly and Christofides (1989) χρησιμοποιείται μία μέθοδος που αναπτύχθηκε το 1970 από τον N.Z.Shor, και παρουσιάζεται εκτενέστερα στο βιβλίο του Shor (1985). Η μέθοδος αυτή μεγιστοποιεί ή ελαχιστοποιεί μη διαφορίσιμες συναρτήσεις (subgradient method). Μια άλλη μέθοδος προτάθηκε στην εργασία των Juttner et al. (2001) αλλά αφορά προβλήματα με έναν περιορισμό. Στη συνέχεια θα γίνει πιο αναλυτική αναφορά στις μεθόδους αυτές.

Επανερχόμενοι στο αρχικό πρόβλημα αντικαθιστώντας τους περιορισμούς της σχέσης (2) μέσω της χρήσης πολλαπλασιαστών Lagrange $u_i \geqslant 0$, i=1, 2,.., n προκύπτει η νέα διατύπωση της αντικειμενικής συνάρτησης του προβλήματος :

$$w(u) = min_{p \epsilon P(s,t)} \{ \sum_{e \epsilon p} w_{(e)} - u_1 (b_1 - \sum_{e \epsilon p} R^1_{(e)}) - \ldots\ldots - u_n (b_n - \sum_{e \epsilon p} R^n_{(e)}) \}$$

Όπου w(u) είναι το κάτω όριο του προβλήματος για πολλαπλασιαστές $u_i \geqslant 0$, i=1, 2,.., n. Το βέλτιστο κάτω όριο του προβλήματος προκύπτει για πολλαπλασιαστές Lagrange $u_i^* \geqslant 0$, i=1, 2,.., n , τέτοιους ώστε :
w(u^*) = max w(u), άρα για τον προσδιορισμό του βέλτιστου κάτω ορίου πρέπει να επιλυθεί το πρόβλημα :

$$\max \{ min_{p \epsilon P(s,t)} \{ \sum_{e \epsilon p} w_{(e)} - u_1 (b_1 - \sum_{e \epsilon p} R^1_{(e)}) - \ldots\ldots - u_n (b_n - \sum_{e \epsilon p} R^n_{(e)}) \} \}$$

Αρχικά θα εξεταστεί η περίπτωση όπου υπάρχει μόνο ένας περιορισμός διαθέσιμων πόρων b. Σε αυτή την περίπτωση υπάρχει μόνο ένας πολλαπλασιαστής Lagrange u ⩾ 0 και το βέλτιστο κάτω όριο προκύπτει ως :

$$w_* = \max \{ min_{p \epsilon P(s,t)} \{ \sum_{e \epsilon p} w_{(e)} - u (b - \sum_{e \epsilon p} R_{(e)}) \} \}$$

όπου : $w(u) = min_{p \epsilon P(s,t)} \{ \sum_{e \epsilon p} w_{(e)} - u (b - \sum_{e \epsilon p} R_{(e)}) \}$

Είναι προφανές δε ότι για κάθε πολλαπλασιαστή u ⩾ 0 η συνάρτηση w(u) ελαχιστοποιείται εάν επιλυθεί το πρόβλημα εύρεσης της συντομότερης διαδρομής από το r στο s με τροποποιημένα βάρη συνδέσμων :

$\bar{w}(e)$ = w(e) + u × R(e) , u ⩾ 0.

Αρχικά παρουσιάζεται η μέθοδος που προτάθηκε στην εργασία των Juttner et al. (2001), γνωστή και ως αλγόριθμος LARAC (Lagrange Relaxation Based Aggregation Cost). Εάν για βάρος συνδέσμων $\overline{w}(e)$ προκύπτει ελάχιστη διαδρομή r, s με κατανάλωση διαθέσιμου πόρου R_p > b, τότε η διαδρομή αυτή ονομάζεται p_c, ενώ σε διαφορετική περίπτωση ονομάζεται p_d. Ο νέος πολλαπλασιαστής u είναι :

$u^* = \frac{\sum_{e \epsilon p_c} w(e) - \sum_{e \epsilon p_d} w(e)}{\sum_{e \epsilon p_d} R_{(e)} - \sum_{e \epsilon p_c} R_{(e)}}$ και προκύπτει η νέα διαδρομή p_l χρησιμοποιώντας τον αλγόριθμο Dijkstra με βάρη συνδέσμων $\overline{w}(e)$ = w(e) + u^* × R(e), εάν για τη νέα διαδρομή p_l ισχύει : $\sum_{e \epsilon p_c} w(e) = \sum_{e \epsilon p_l} w(e) = \sum_{e \epsilon p_d} w(e)$, τότε ο αλγόριθμος τερματίζεται αλλιώς η διαδρομή p_l ονομάζεται p_c ή p_d όπως προηγουμένως και η διαδικασία επαναλαμβάνεται. Για να ικανοποιηθεί η παραπάνω απαίτηση πρέπει το άνω όριο και το κάτω όριο της συνάρτησης να είναι ίσα για κάποιον πολλαπλασιαστή u^*, κάτι τέτοιο όμως δεν αναμένεται να συμβεί παρά μόνο εάν προκύψει το πραγματικό βέλτιστο.

ΑΛΓΟΡΙΘΜΟΣ LARAC ΓΙΑ ΤΟΝ ΠΡΟΣΔΙΟΡΙΣΜΟ ΤΟΥ ΒΕΛΤΙΣΤΟΥ ΚΑΤΩ ΟΡΙΟΥ ΓΙΑ ΤΟ ΧΡΟΝΟΥ ΔΙΑΔΡΟΜΗΣ

Βήμα 1(Αρχικοποιήσεις)
UpperBound ← 0 // Το κόστος της βέλτιστης διαδρομής που παραβιάζει οριακά
// τον περιορισμό διαθέσιμων πόρων
LowerBound ← + ∞ // Το κόστος της βέλτιστης διαδρομής που δεν παραβιάζει τον
// περιορισμό διαθέσιμων πόρων
Υπολόγισε τη βέλτιστη διαδρομή με κόστος συνδέσμων w(e)
SSSP(w(e)) : w_p, R_p // κόστος διαδρομής και κατανάλωση διαθέσιμου πόρου
Εάν $R_p \leq$ b
Η διαδρομή αυτή είναι βέλτιστη. Επέστρεψε w_p και τερμάτισε τον αλγόριθμο.
Αλλιώς
Constrained ← R_p // κατανάλωση διαθέσιμου πόρου
$w_c \leftarrow w_p$
Τέλος Εάν
Υπολόγισε τη βέλτιστη διαδρομή p^* με κόστος συνδέσμων : R(e)
SSSP(R(e)) : w_{p^*}, R_{p^*} // κόστος διαδρομής και κατανάλωση διαθέσιμου πόρου
Εάν R_p > b
Δεν υπάρχει εφικτή λύση στο πρόβλημα. Επέστρεψε w_{p^*} και τερμάτισε.
Αλλιώς
Unconstrained ← R_p
$w_d \leftarrow w_{p^*}$
Τέλος Εάν
Βήμα 2(Γενική Επανάληψη)
Αρχή
u = (w_c - w_d) / (Unconstrained – Constrained)

$\overline{w}(e) \leftarrow$ w(e) + u × R(e)
SSSP($\overline{w}$(e)) : w_p, R_p // όπου $w_p = \sum_{e \epsilon p} w(e)$ το πραγματικό κόστος της βέλτιστης
// διαδρομής p που προέκυψε για βάρος συνδέσμων w(e) + u × R(e)
Εάν UpperBound = LowerBound
Επέστρεψε το κόστος w_p και τερμάτισε τη διαδικασία, διότι βρέθηκε το πραγματικό βέλτιστο.
Τέλος Εάν
Εάν $R_p \leq$ b
$w_d \leftarrow w_p$
Unconstrained $\leftarrow R_p$
Αλλιώς
$w_c \leftarrow w_p$
Constrained $\leftarrow R_p$
Τέλος Εάν
Εάν UpperBound < w_c, τότε UpperBound $\leftarrow w_c$
Εάν LowerBound > w_d, τότε LowerBound $\leftarrow w_d$
Επέστρεψε στην Αρχή και επανέλαβε τη διαδικασία

Όπως έχει διατυπωθεί ο αλγόριθμος η διαδικασία μπορεί να συγκλίνει μόνο εάν υπάρχει πραγματικό βέλτιστο στο σημείο που καταναλώνεται όλη η ποσότητα διαθέσιμων πόρων : UpperBound = LowerBound. Για το λόγο αυτό η συνθήκη τερματισμού πρέπει να βελτιωθεί. Για παράδειγμα μπορεί να οριστεί ο τερματισμός της διαδικασίας μετά από έναν αριθμό επαναλήψεων ή όταν επιτευχθεί σύγκλιση σε αποδεκτά επίπεδα. Έτσι προκύπτουν δύο διαδρομές που δεν παραβιάζουν ή παραβιάζουν οριακά τον περιορισμό διαθέσιμων πόρων.

Η επόμενη μέθοδος (Subgradient Method) είναι μία μέθοδος που εφαρμόζεται για τον προσδιορισμό του μέγιστου ή του ελάχιστου σε μη διαφορίσιμες συναρτήσεις. Η μέθοδος αυτή επεκτείνεται και στη γενικότερη περίπτωση που το πρόβλημα έχει περισσότερους από έναν περιορισμούς και είναι η βασικότερη μέθοδος που χρησιμοποιείται σε προβλήματα σχετικά με τον υπολογισμό των βέλτιστων πολλαπλασιαστών Lagrange.

Επανερχόμενοι στο πρόβλημα προσδιορισμού του βέλτιστου κάτω ορίου, έχουμε :
w(u) = $min_{p \epsilon P(s,t)}$ { $\sum_{e \epsilon p} w_{(e)}$ - u (b - $\sum_{e \epsilon p} R_{(e)}$) }
w_* = max { $min_{p \epsilon P(s,t)}$ { $\sum_{e \epsilon p} w_{(e)}$ - u (b - $\sum_{e \epsilon p} R_{(e)}$) }

Χρησιμοποιώντας την παραπάνω μέθοδο ο πολλαπλασιαστής u ⩾ 0 παίρνει διαδοχικά διαφορετικές τιμές ξεκινώντας από την αρχική τιμή u^0 και για κάθε επανάληψη προκύπτει ως :
u^{k+1} = max{ 0 , u^k - t_k × $\widetilde{g_k}$ }

Αρχικά υπολογίζεται το κάτω όριο w(u_k). Για τον υπολογισμό αυτό επιλύεται ένα πρόβλημα εύρεσης της βέλτιστης διαδρομής στο δίκτυο με κόστη συνδέσμων :
$\bar{w}(e)$ = w(e) + u^k × R(e) , u ⩾ 0.
Κατά τον προσδιορισμό της παραπάνω διαδρομής χρησιμοποιείται ο αλγόριθμος Dijkstra και προκύπτει και η κατανάλωση διαθέσιμων πόρων $R_p = \sum_p R(e)$ για τη διαδρομή αυτή.

Στη συνέχεια υπολογίζεται η βαθμωτή μεταβλητή :
g_k = g_k (u^k) = b - $\sum_p R(e)$, όπου p η βέλτιστη διαδρομή για πολλαπλασιαστή Lagrange u^k όπως προέκυψε στο προηγούμενο βήμα.

Εάν g_k = 0, τότε το βέλτιστο έχει προσδιοριστεί και η διαδικασία τερματίζεται. Επειδή όμως κάτι τέτοιο συνήθως δεν είναι εφικτό μπορεί να προσδιοριστεί ένας αριθμός επαναλήψεων N πέρα του οποίου η διαδικασία θα τερματιστεί.

Η μεταβλητή t_k είναι βαθμωτή. Μάλιστα μετά από πολλές επαναλήψεις $t_k \rightarrow 0$. Μία βαθμωτή σχέση για τη μεταβλητή t_k που έχει αποδειχθεί αποδοτική σε πρακτικά προβλήματα είναι :

$$t_k = \frac{\varphi_k (\ w^* - w(u^k) \ )}{\|g_k\|^2}$$, Held et al. (1974).

Κατά την αρχικοποίηση μία αποδοτική τιμή για τη μεταβλητή t_k είναι t_0 = 10. Έτσι υπολογίζεται η μεταβλητή φ_0 ώστε t_0 = 10. Η φ_k είναι βαθμωτή μεταβλητή με τιμές κυρίως εντός του διαστήματος{0,2}. Κατά τη διάρκεια των επαναλήψεων η τιμή της μειώνεται. Εάν μάλιστα για ένα μεγάλο αριθμό επαναλήψεων η τιμή της $w(u^k)$ παραμένει αμετάβλητη, τότε υποδιπλασιάζεται. Επίσης w^* είναι η προσδοκώμενη τιμή στην οποία επιδιώκεται να συγκλίνει η $w(u^k)$ μέσω διαδοχικών επαναλήψεων.

Τέλος η τιμή $\widetilde{g_k}$ προσδιορίζει τη νέα κατεύθυνση προς την οποία θα επιχειρηθεί η σύγκλιση στο επόμενο βήμα. Στην τυπική διατύπωση της μεθόδου και σε πολλές εφαρμογές της προτείνεται : $\widetilde{g_k}$ = g_k (u^k), ωστόσο όσο περισσότερες παρελθοντικές τιμές της g_k ληφθούν υπόψη τόσο ταχύτερα επιτυγχάνεται η σύγκλιση. Αυτό διατυπώνονται και στην εργασία του Crowder (1976). Γενικές προτάσεις για τον υπολογισμό της μεταβλητής $\widetilde{g_k}$ σε κάθε βήμα είναι :

$\widetilde{g_k}$ = 0.7 g_k + 0.3 g_{k-1} , $\widetilde{g_k}$ = 0.6 g_k + 0.2 g_{k-1} + 0.1 g_{k-2} + 0.1 g_{k-3}

Η παραπάνω μέθοδος παρουσιάζεται σε μορφή αλγορίθμου ως :

ΑΛΓΟΡΙΘΜΟΣ ΠΡΟΣΔΙΟΡΙΣΜΟΥ ΤΟΥ ΒΕΛΤΙΣΤΟΥ ΚΑΤΩ ΟΡΙΟΥ
ΣΥΜΦΩΝΑ ΜΕ ΤΗ ΜΕΘΟΔΟ Subgradient

Έστω N ο αποδεκτός βαθμός επαναλήψεων
Επέλεξε έναν αρχικό πολλαπλασιαστή Lagrange u_0
k ← 0
Αρχή
Υπολόγισε το w(u_k) με χρήση του αλγορίθμου Dijkstra και βάρη συνδέσμων
$\bar{w}(e)$ ← w(e) + u_k × R(e)
R ← $\sum_{e \epsilon p} R_{(e)}$
Εάν k > 2 & w(u_k) = w(u_{k-1}) = w(u_{k-2})
$\varphi_k \leftarrow \frac{\varphi_k}{2}$
Τέλος Εάν
g_k ← b – R
Εάν g_k ← 0 ή N = k
Τερμάτισε τον αλγόριθμο
Τέλος Εάν
Εάν k = 0
t_k ← 10
$\varphi_k \leftarrow \frac{10 \times \|g_k\|^2}{(w^* - w(u^k))}$
Αλλιώς
$t_k \leftarrow \frac{\varphi_k (w^* - w(u^k))}{\|g_k\|^2}$
Τέλος Εάν
Εάν k > 3
$\widetilde{g_k}$ ← 0.6 g_k +0.2 g_{k-1} + 0.1 g_{k-2} + 0.1 g_{k-3}
Αλλιώς
$\widetilde{g_k} \leftarrow g_k$
Τέλος Εάν
$u_{k+1} \leftarrow u_k$ - $t_k \times \widetilde{g_k}$
Επέστρεψε στην Αρχή

Η μέθοδος αυτή εφαρμόζεται και σε προβλήματα με περισσότερους του ενός διαθέσιμους πόρους. Όπως παρουσιάστηκε και παραπάνω εάν ένα πρόβλημα έχει i=1,2,..,n περιορισμούς, τότε ορίζονται αντίστοιχα πολλαπλασιαστές Lagrange $u_i \geqslant 0$, i=1,2,..,n. Για την εύρεση του βέλτιστου κάτω ορίου απαιτείται η εύρεση των τιμών που θα έχουν οι πολλαπλασιαστές αυτοί ώστε :

max $\{min_{p \epsilon P(s,t)} \{ \sum_{e \epsilon p} w_{(e)} - u_1 (b_1 - \sum_{e \epsilon p} R^1_{(e)}) - \ldots\ldots - u_n (b_n - \sum_{e \epsilon p} R^n_{(e)}) \} \}$

Το πρόβλημα αυτό στη συνέχεια διασπάται σε τόσα επιμέρους υποπροβλήματα, όσοι είναι και οι πολλαπλασιαστές Lagrange. Η επίλυση του βασικού προβλήματος επιμερίζεται σε δύο επαναληπτικές διαδικασίες διαφορετικών επιπέδων. Κατά τη διαδικασία χαμηλού επιπέδου επιλύονται όλα τα επιμέρους υποπροβλήματα με τις τρέχουσες τιμές των πολλαπλασιαστών Lagrange και κατά τη διαδικασία υψηλού επιπέδου συγκεντρώνονται τα αποτελέσματα επίλυσης των επιμέρους υποπροβλημάτων και προσδιορίζονται οι νέες τιμές για τους πολλαπλασιαστές ώστε να συνεχιστούν οι επαναλήψεις. Αυτό απαιτεί αυξημένο υπολογιστικό φόρτο, καθώς στην περίπτωση ενός περιορισμού έπρεπε να επιλυθεί μόνο ένα υποπρόβλημα μέσω του αλγορίθμου Dijkstra σε κάθε επανάληψη. Στα προβλήματα με πολλούς περιορισμούς διαθέσιμων πόρων ο χρόνος εκτέλεσης εξαρτάται από τον αριθμό αυτών των υποπροβλημάτων και τον αριθμό των επαναλήψεων. Πιο συγκεκριμένα η μέθοδος αναλύεται ως εξής :

Οι νέες τιμές των πολλαπλασιαστών u_i , i=1,2,..,n προκύπτουν από τη σχέση :
u_i^{k+1} = max{ 0 , u_i^k - t_k × $\widetilde{g_k}$ } , ∀ i =1,n

Αρχικά υπολογίζεται το κάτω όριο w(u_i^k), i=1,2,…,n. Για τον υπολογισμό αυτό επιλύονται n υποπρόβληματα εύρεσης της βέλτιστης διαδρομής στο δίκτυο με κόστη συνδέσμων :
$\bar{w}(e)$ = w(e) + u_i^k × R_i(e) , $u_i \geq 0$.
Κατά τον προσδιορισμό όλων των παραπάνω διαδρομών χρησιμοποιείται ο αλγόριθμος Dijkstra και προκύπτει η κατανάλωση διαθέσιμων πόρων $R_{p,i} = \sum_p R_i(e)$ για κάθε διαδρομή διαδρομή.

Στη συνέχεια υπολογίζεται η βαθμωτή μεταβλητή :
$g_k = g_k\,(u^k)$ = b - R_p , όπου g_k, b, R_p μονοδιάστατοι πίνακες n × 1.

Εάν $g_{k,i}$ = 0, ∀ i =1,2,..,n , τότε το βέλτιστο έχει προσδιοριστεί και η διαδικασία τερματίζεται.

$t_k = \frac{\varphi_k(\ w^* - w(u^k)\ )}{\|g_k\|^2}$ και η τιμή $\widetilde{g_k}$ υπολογίζεται όπως και προηγουμένως.

Όπως έχει αναφερθεί ήδη το χρονικό κόστος αυτής της διαδικασίας είναι ο επανυπολογισμός του δικτύου για κάθε περιορισμό i=1,2,...,n με βάρη συνδέσμων $\bar{w}(e)$ = w(e) + u_i^k × R_i(e) σε κάθε επανάληψη. Στην ουσία δηλαδή το πρόβλημα από άποψη χρόνου εκτέλεσης είναι η ανάγκη για επίλυση όλων των υποπροβλημάτων σε κάθε επανάληψη. Στην εργασία των Zhao et al. (1999) προτάθηκε μία μέθοδος που οδηγεί σε συγκλίσεις επιλύοντας μόνο ένα υποπρόβλημα σε κάθε επανάληψη, ωστόσο απαιτείται μεγαλύτερος αριθμός επαναλήψεων.

5.7.3.ΜΕΘΟΔΟΣ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Η επόμενη μέθοδος έχει προταθεί από τον Desrochers (1988) και έχει μελετηθεί στη συνέχεια από τους Barnhart et al. (1998), Feillet et al. (2004), Righini and Salani (2005). Η μέθοδος αυτή βασίζεται στο δυναμικό προγραμματισμό, ωστόσο ο χρόνος εκτέλεσής της μπορεί να είναι ακόμα και εκθετικός στη χειρότερη περίπτωση. Στην ουσία οδηγεί στην απαρίθμηση όλων των εφικτών διαδρομών με δυναμικό τρόπο.

Αναλύοντάς την θεωρείται αρχικά μία διαδρομή μεταξύ δύο κόμβων r, i και συμβολίζεται ως X_{ri}. Όταν μέσω της διαδρομής αυτής το μεταφορικό μέσο φθάσει στον κόμβο i έχει καταναλώσει διαθέσιμους πόρους : $(R_i^1, R_i^2,\ldots, R_i^n)$, όπου n ο αριθμός των διαθέσιμων πόρων που υφίστανται περιορισμούς. Επίσης το κόστος διάνυσης της διαδρομής αυτής που προκύπτει από το κόστος διάνυσης των συνδέσμων από τους οποίους αποτελείται είναι : C_i. Για τη διαδρομή X_{ri} μπορεί να αποθηκευτούν σε μία μήτρα r_i οι ακόλουθες πληροφορίες :
$\lambda_i = (R_i^1, R_i^2,\ldots, R_i^n, s_i, k_i^1, k_i^2, \ldots, k_i^N, C_i)$, όπου s_i ο αριθμός των κόμβων από τους οποίους αποτελείται η διαδρομή X_{ri} και k_i^j μία δίτιμη μεταβλητή με τιμές 0,1 που δείχνει εάν ο κόμβος j ανήκει στη διαδρομή : $k_i^j = 1$ ή δεν ανήκει σε αυτήν : $k_i^j = 0$. Από τα παραπάνω είναι προφανές ότι $s_i = \sum_{z=1}^{N} k_i^z$.

Εάν υπάρχουν δύο διαφορετικές διαδρομές για τη μετακίνηση r→i : X_{ri}, X'_{ri} τότε η διαδρομή X_{ri} θεωρείται ότι επικρατεί της διαδρομής X'_{ri} εάν ισχύουν : $C_i < C'_i$ και $R_i^p < R_i^{p'}$ για κάθε pϵ{1,n}. Εάν κάποια από αυτές τις συνθήκες δεν ισχύει τότε δεν επικρατεί καμία διαδρομή. Κατά την εκτέλεση της μεθόδου χρησιμοποιούνται μόνο οι διαδρομές που επικρατούν ώστε να μειωθεί ο αριθμός των διαδρομών. Αυτό συμβαίνει διότι εάν μία διαδρομή επικρατεί κάποιας άλλης κατά τη μετακίνηση r→i θα συνεχίσει να επικρατεί και κατά την επέκταση της διαδρομής με κατεύθυνση τον κόμβο προορισμού. Αυτό μπορεί να αποδειχθεί θεωρώντας τυχαίο κόμβο j στον οποίο επεκτείνεται η διαδρομή X_{ri}. Αρχικά είναι προφανές ότι επειδή $R_i^p < R_i^{p'}$ για κάθε pϵ{1,n} ισχύει και $R_j^p = R_i^p + t_{i,j}^p < R_j^{p'} = R_i^{p'} + t_{i,j}^p$ για κάθε pϵ{1,n}, όπου $t_{i,j}^p$ η κατανάλωση του διαθέσιμου πόρου p κατά τη διάνυση του συνδέσμου i, j. Στη συνέχεια αφού $C_i < C'_i$ προκύπτει $C_i + w_{i,j} < C'_i + w_{i,j}$.

Εάν μία διαδρομή X_{ri}, τα στοιχεία της οποίας είναι αποθηκευμένα στη μήτρα r_i, επεκταθεί προς κάποιον άλλο κόμβο του δικτύου j είναι σημαντικό να υπάρχει πληροφόρηση σχετικά με το αν μπορεί να πραγματοποιηθεί αυτή η επέκταση. Αρχικά η διαδρομή X_{ri} μπορεί να επεκταθεί μόνο προς τους κόμβους που συνδέονται με τον κόμβο i μέσω συνδέσμου που εξέρχεται από αυτόν. Επίσης δε μπορεί ο επόμενος κόμβος από τον i να υπάρχει ήδη στη διαδρομή X_{ri} και η τελευταία προϋπόθεση είναι να πληρούνται όλοι οι περιορισμοί διαθέσιμων πόρων για τη νέα διαδρομή X_{rj}, δηλαδή :

$R_j^p = R_i^p + t_{i,j}^p < b^p$ για κάθε p={1,n}, όπου $t_{i,j}^p$ η κατανάλωση του διαθέσιμου πόρου p κατά τη διάνυση του συνδέσμου i,j και b^p ο περιορισμός του διαθέσιμου πόρου p. Έτσι μπορεί να θεωρηθεί μία νέα μήτρα για τα στοιχεία της διαδρομής X_{ri} :
$\lambda_i = (R_i^1, R_i^2, \ldots, R_i^n, s_i, k_i^1, k_i^2, \ldots, k_i^N, C_i)$, όπου s_i ο αριθμός των κόμβων στους οποίους δε μπορεί να επεκταθεί η διαδρομή X_{ri} και k_i^j μία δίτιμη μεταβλητή με τιμές 0,1 που δείχνει εάν ο κόμβος j είναι κόμβος προς τον οποίο μπορεί να επεκταθεί η διαδρομή X_{ri} : $k_i^j = 1$ ή όχι : $k_i^j = 0$. Είναι επίσης προφανές ότι $s_i = \sum_{z=1}^{N} k_i^z$.
Σύμφωνα με τα παραπάνω μπορεί να διατυπωθεί ο αλγόριθμος :

ΑΛΓΟΡΙΘΜΟΣ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΓΙΑ ΤΟ RCSPP

Βήμα 1(Αρχικοποιήσεις)
Έστω Λ_i μία λίστα με μήτρες λ_i στις οποίες είναι αποθηκευμένα τα στοιχεία διαφορετικών διαδρομών r→i για κάθε κόμβο i του δικτύου
Q = {r} , μία μήτρα στην οποία αποθηκεύονται οι κόμβοι του δικτύου από τους οποίους μπορεί να ξεκινήσει η επόμενη επανάληψη του αλγορίθμου
$\Lambda_r \leftarrow (0,0,0,\ldots,0)$
Επέκταση (λ_i , j) : μία υπορουτίνα που επεκτείνει μία από τις διαδρομές r → i , η οποία περιγράφεται από τη μήτρα $\lambda_i \in \Lambda_i$, σε έναν κόμβο j και είτε δεν επιστρέφει αποτέλεσμα εάν ο κόμβος j δεν είναι προσπελάσιμος μέσω αυτής της διαδρομής, είτε επιστρέφει τη μήτρα λ_j . Για το σκοπό αυτό υπολογίζεται το R_j^p για κάθε p={1,n}, το C_j και οι κόμβοι προς τους οποίους δε μπορεί να επεκταθεί η νέα διαδρομή r→j
E(i) : το σύνολο των κόμβων που συνδέονται άμεσα με τον κόμβο i
Επικράτηση (Λ_i) : μία υπορουτίνα που συγκρίνει τις διαδρομές r→i ώστε στη λίστα Λ_i να περιλαμβάνονται μόνο οι διαδρομές που έχουν επικρατήσει σε άλλες διαδρομές, διαγράφοντας τις υπόλοιπες μήτρες λ_i
F(i,j) : οι μήτρες λ_j που προκύπτουν από την επέκταση των αντίστοιχων λ_i

Βήμα 2(Βασική Επανάληψη)
Όσο Q ≠ ∅
 Για Κάθε i = 1,N
  Εάν i ≠ r
  Λ(i) ← ∅
  Τέλος Εάν
 Τέλος Για Κάθε
Επέλεξε τον επόμενο κόμβο i από τη μήτρα Q
  Για Κάθε j = E(i)
  F(i,j) ← ∅
    Για Κάθε $\lambda_i = (R_i^1, R_i^2, \ldots, R_i^n, s_i, k_i^1, k_i^2, \ldots, k_i^N, C_i) \in \Lambda_i$
      Εάν $k_i^j = 0$
      F(i,j) ← F(i,j) ∪ { Επέκταση (λ_i , j) }

Τέλος Εάν
$\Lambda_j \leftarrow$ Επικράτηση (Λ_j , F(i,j))
Τέλος Για Κάθε
Q ← Q ∪ { j }
Τέλος Για Κάθε
Q ← Q \ { i }
Τέλος Για Κάθε

5.7.4.ΜΕΘΟΔΟΣ ΔΙΑΓΡΑΦΗΣ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΔΙΚΤΥΟΥ ΠΟΥ ΔΕΝ ΥΠΑΚΟΥΟΥΝ ΣΕ ΠΕΡΙΟΡΙΣΜΟΥΣ

Στη συνέχεια αναπτύσσεται μία ακόμα μέθοδος που βασίζεται στη διαγραφή κόμβων και συνδέσμων από το δίκτυο, περιορίζοντας έτσι το μέγεθος του προβλήματος. Τα επιμέρους βήματα αυτής της μεθόδου είναι τα ακόλουθα :

Βήμα 1^{ο} : Διαγράφονται όλοι οι κόμβοι που δεν αποτελούν κόμβο προέλευσης για κανένα σύνδεσμο του δικτύου. Από τη διαγραφή αυτή εξαιρείται ο κόμβος προορισμού s. Η διαγραφή αυτή είναι αποδεκτή καθώς από κόμβους που δεν εξέρχεται κανένας σύνδεσμος δεν είναι δυνατό να υπάρχει διαδρομή προς τον κόμβο προορισμού.

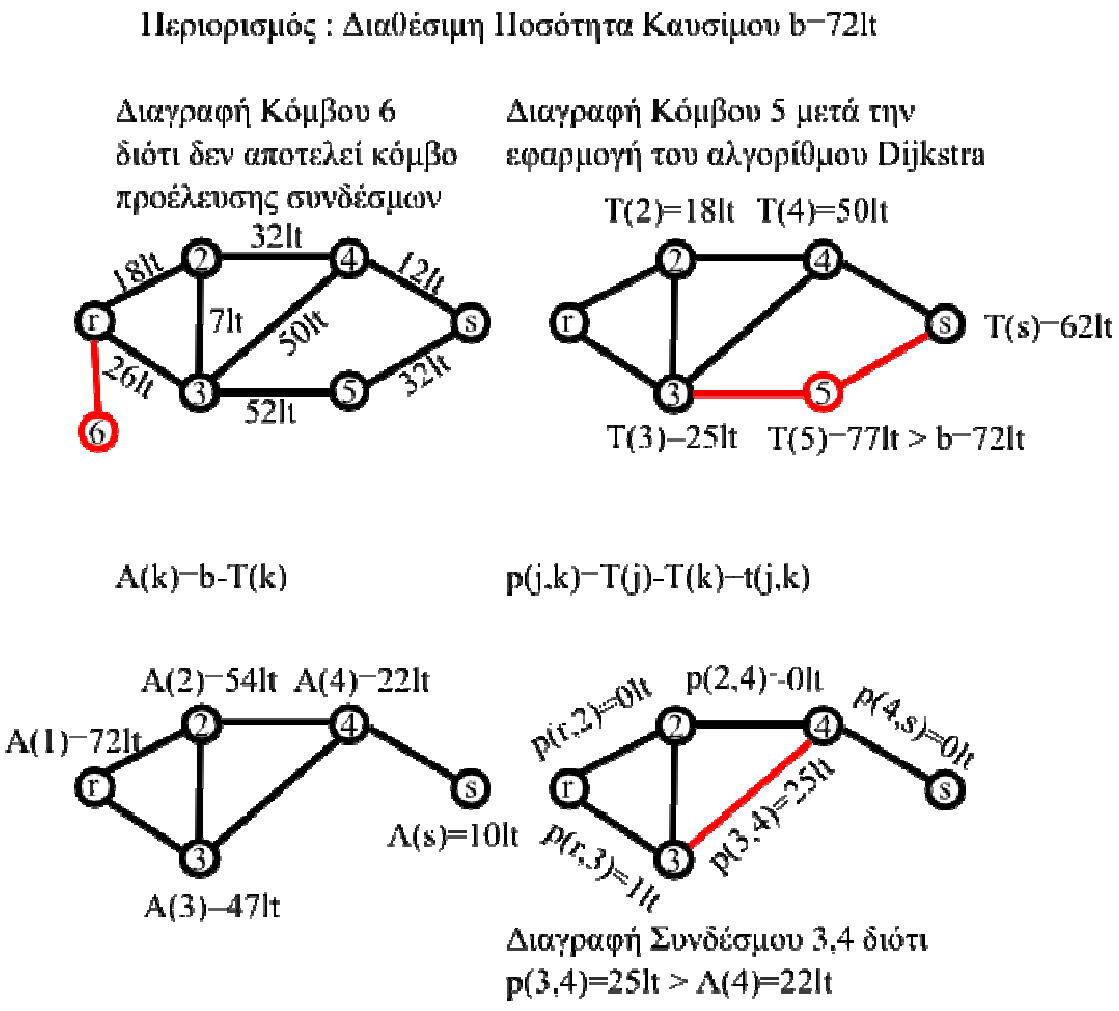
Βήμα 2^{ο} : Εφαρμόζεται ο αλγόριθμος Dijkstra τόσες φορές όσοι είναι και οι περιορισμοί που πρέπει να πληρούνται χρησιμοποιώντας κάθε φορά ως βάρη συνδέσμων την κατανάλωση διαθέσιμων πόρων από αυτούς. Για κάθε διαθέσιμο πόρο R_i υπάρχει μία ακραία τιμή b_i η οποία δεν πρέπει να παραβιάζεται. Έτσι, εφαρμόζοντας τον αλγόριθμο Dijkstra για τον περιορισμό i προκύπτει η ελάχιστη ποσότητα διαθέσιμου πόρου T_j^i που απαιτείται για τη μετακίνηση από τον κόμβο r σε κάθε κόμβο j του δικτύου. Προφανώς για όσους κόμβους ισχύει : $T_j^i > b_i$, διαγράφονται από το δίκτυο. Μαζί με τους κόμβους διαγράφονται και οι σύνδεσμοι που εισέρχονται ή εξέρχονται από αυτούς. Η διαγραφή αυτή δικαιολογείται καθώς εάν $T_j^i > b_i$, τότε δεν υπάρχει εναλλακτική που να ικανοποιεί τον περιορισμό i καθώς αυτός δεν ικανοποιείται ούτε από τη διαδρομή ελάχιστης κατανάλωσης του διαθέσιμου πόρου i.

Μετά την εφαρμογή του αλγορίθμου Dijkstra η ελάχιστη ποσότητα διαθέσιμου πόρου i που καταναλώνεται κατά τη μετακίνηση r→s προκύπτει ως : T_s^i
Είναι προφανές ότι εάν $T_s^i > b_i$ δεν υπάρχει καμία εφικτή διαδρομή που να ικανοποιεί τον περιορισμό διαθέσιμων πόρων i και το πρόβλημα δεν έχει επίλυση.

Βήμα 3^{ο} : Εάν $T_s^i < b_i$ τότε υπάρχει ποσότητα :
$A_j^i = b_i - T_s^i$ η οποία μπορεί να καταναλωθεί χωρίς να παραβιάζεται ο περιορισμός του διαθέσιμου πόρου i.

Έτσι για κάθε σύνδεσμο j, k που δεν έχει διαγραφεί από το δίκτυο υπολογίζεται η ποσότητα : $p^i_{j,k} = T^i_j - T^i_k + t^i_{j,k}$, όπου $t^i_{j,k}$ η ποσότητα διαθέσιμου πόρου i που καταναλώνεται κατά τη διάνυση του συνδέσμου j, k και $p^i_{j,k}$ η επιπλέον επιβάρυνση στην κατανάλωση του διαθέσιμου πόρου i εάν η διαδρομή διέλθει από το σύνδεσμο j, k. Εάν $p^i_{j,k} = 0$, τότε ο σύνδεσμος j, k ανήκει στη διαδρομή ελάχιστης κατανάλωσης του διαθέσιμου πόρου i. Αλλιώς $p^i_{j,k} > 0$ και εάν $p^i_{j,k} > A^i_k$, όπου A^i_k η επιπλέον ποσότητα διαθέσιμου πόρου i που μπορεί να καταναλωθεί, τότε ο σύνδεσμος αυτός διαγράφεται από το δίκτυο διότι κάθε διαδρομή που διέρχεται από αυτόν το σύνδεσμο παραβιάζει τον περιορισμό i.

Η παραπάνω διαδικασία παρουσιάζεται στο ακόλουθο συγκοινωνιακό δίκτυο του σχήματος στο οποίο εξετάζεται η περίπτωση ενός περιορισμού διαθέσιμου πόρου.



Σχήμα 5.17.Μείωση μεγέθους δικτύου μέσω της προτεινόμενης μεθόδου

Εκτός από τα προβλήματα σε δίκτυα με περιορισμούς διαθέσιμων πόρων υπάρχει και μία γενικότερη κατηγορία προβλημάτων, γνωστή και ως General Resource Constrained Shortest Path Problems (GRCSPP). Σε αυτή την κατηγορία οι περιορισμοί μπορεί να έχουν γενικότερη μορφή. Για παράδειγμα μπορεί να απαιτηθεί κανένας σύνδεσμος που ανήκει στην ελάχιστη διαδρομή να μην έχει τιμή που παραβιάζει την οριακή τιμή του περιορισμού U ή να απαιτηθεί ο μέσος όρος όλων των συνδέσμων της ελάχιστης διαδρομής να μην ξεπερνά την οριακή τιμή. Ακόμα μπορεί να απαιτηθεί ένα ποσοστό, π.χ. p% των συνδέσμων της διαδρομής να έχουν τιμή μικρότερη από την οριακή U κ.α.

Έτσι διαδρομή p θα πρέπει συγχρόνως :

1. Να είναι η ελάχιστη δυνατή από άποψη κόστους : $\sum_{e \in p} w(e)$ = minimum
2. Οι σύνδεσμοι της διαδρομής P να μην παραβιάζουν κανένα περιορισμό που μπορεί να αντιπροσωπευθεί με αριθμητικές τιμές : $\sum_{e \in p}(R_e)_i \leq b_i$,$\forall$ i $\in$ {1,n}
3. Οι σύνδεσμοι της διαδρομής P να πληρούν ανάλογα με τη φύση του προβλήματος κάποιον ή όλους τους περιορισμούς που δε μπορούν να αντιπροσωπευθούν με αριθμητικές τιμές, π.χ. $\sum_{e \in p} Z(e)$< p% $\times$ U_z

Εάν υπάρχει παραβίαση επιλέγεται η επόμενη διαδρομή ελαχίστου κόστους και γίνεται επανέλεγχος μέχρι να βρεθεί διαδρομή που θα ικανοποιεί τους περιορισμούς ή προκύψει τελικά ότι δεν υπάρχει τέτοια διαδρομή μεταξύ των ζητούμενων κόμβων.

5.8.ΣΥΓΚΡΙΣΕΙΣ ΜΕΘΟΔΩΝ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΣΥΓΚΟΙΝΩΝΙΑΚΩΝ ΔΙΚΤΥΩΝ

Όπως έχει αναφερθεί ήδη από την ενότητα 4.5. υπάρχουν δύο κύριοι τρόποι αναπαράστασης συγκοινωνιακών δικτύων και είναι η μέθοδος του Πίνακα γειτνίασης και η μέθοδος της Λίστας γειτνίασης. Στις ενότητες 5.1.-5.7. έχουν χρησιμοποιηθεί και οι δύο μέθοδοι. Για το λόγο αυτό θα εξεταστεί υπό ποιές συνθήκες μπορεί να χρησιμοποιηθεί η μία και υπό ποιές η άλλη ώστε να επιτευχθεί το βέλτιστο αποτέλεσμα από άποψη μείωσης των απαιτούμενων θέσεων μνήμης και του χρόνου εκτέλεσης ενός αλγορίθμου.

Με τη μέθοδο του πίνακα γειτνίασης θεωρείται ότι κάθε κόμβος i συνδέεται με κάθε κόμβο j του δικτύου. Έτσι η αντίσταση μετακίνησης από τον κόμβο i προς κάθε κόμβο j δίνεται ως w(i,j). Στην πραγματικότητα όμως ο κόμβος i δε συνδέεται με κάθε κόμβο j του δικτύου αλλά με έναν περιορισμένο αριθμό από αυτούς. Έτσι εάν δύο κόμβοι i, j δε συνδέονται μεταξύ τους θεωρείται ότι έχουν βάρος w(i,j) = +∞. Ως αποτέλεσμα για την αποθήκευση όλων των συνδέσμων ενός δικτύου απαιτούνται N^2 θέσεις μνήμης.

Με τη μέθοδο της Λίστας γειτνίασης αποθηκεύονται μόνο οι σύνδεσμοι που υπάρχουν στο δίκτυο. Για κάθε σύνδεσμο eϵE αποθηκεύεται ο κόμβος προέλευσής του S(e) και ο κόμβος προορισμού του E(e), επομένως απαιτούνται 3×E θέσεις μνήμης. Έτσι εάν χρησιμοποιηθεί η Λίστα Γειτνίασης σε ένα πολύ πυκνό δίκτυο που μπορεί να υπάρχουν μέχρι και N(N-1) σύνδεσμοι απαιτούνται 3(N^2-N) θέσεις μνήμης και είναι προτιμότερη η χρήση του Πίνακα Γειτνίασης. Εάν όμως το δίκτυο δεν είναι τόσο πυκνό και οι σύνδεσμοι του δικτύου E < $\frac{N^2}{3}$ είναι προτιμότερη η χρήση Λίστας Γειτνίασης. Γενικότερα σε συνήθη δίκτυα ισχύει ο παραπάνω περιορισμός και μάλιστα E <<< $\frac{N^2}{3}$, οπότε η χρήση Λίστας γειτνίασης είναι προτιμότερη από το Πίνακα γειτνίασης από άποψη θέσεων μνήμης.

Όσον αφορά το χρόνο εκτέλεσης ενός αλγορίθμου θα εξεταστεί μία επανάληψη που χρησιμοποιείται στους περισσότερους αλγορίθμους που παρουσιάστηκαν μέχρι τώρα. Η επανάληψη αυτή είναι : Εάν το κόστος μετακίνησης από τον κόμβο προέλευσης r προς τον κόμβο i είναι D(i), τότε για κάθε κόμβο j που συνδέεται με τον κόμβο i μέσω συνδέσμου να ελεγχθεί εάν το κόστος D(j) είναι μεγαλύτερο από το κόστος D(i)+[Βάρος Συνδέσμου i,j] και αν ισχύει αυτό να αναπροσαρμοστεί η τιμή D(j) ως D(j) = D(i)+[Βάρος Συνδέσμου i,j].

α) Αναπαράσταση με Πίνακα Γειτνίασης : Θεωρείται ότι από τον κόμβο i εξέρχονται N σύνδεσμοι προς κάθε κόμβο του δικτύου. Έτσι για να ολοκληρωθεί η παρακάτω επανάληψη απαιτούνται N συγκρίσεις :

Για Κάθε j = 1,N
Εάν D(j) > D(i) + w(i,j)
D(j) ← D(i) + w(i,j)

Τέλος Εάν
Τέλος Για Κάθε

β) Αναπαράσταση με Λίστα Γειτνίασης : Εξετάζονται όλοι οι σύνδεσμοι Ε που υπάρχου στο δίκτυο και γίνεται ο παραπάνω έλεγχος σε όσους από αυτούς έχουν κόμβο προέλευσης τον κόμβο i, οπότε απαιτούνται Ε συγκρίσεις :

Για Κάθε e = 1,E
Εάν D(E(e)) > D(S(e)) + w(e) & S(e) = i
D(E(e)) ← D(S(e)) + w(e)
Τέλος Εάν
Τέλος Για Κάθε

Για να είναι πιο αποδοτική από άποψη χρόνου εκτέλεσης η Λίστα γειτνίασης πρέπει ο αριθμός των συνδέσμων να είναι μικρότερος από τον αριθμό των κόμβων του δικτύου, κάτι που συμβαίνει σε σπάνιες περιπτώσεις. Ωστόσο ακόμα και αν αυτό δεν ισχύει χρησιμοποιείται συνήθως η Λίστα γειτνίασης καθώς η επιβάρυνση στο χρόνο εκτέλεσης είναι πολύ μικρή σε σχέση με την επιβάρυνση στον αριθμό των θέσεων μνήμης που απαιτούνται κατά την αποθήκευση του δικτύου με Πίνακα γειτνίασης, εκτός και αν το δίκτυο είναι πολύ πυκνό.

γ) Στη συνέχεια ερευνάται μία μέθοδος που βασίζεται στη Λίστα Γειτνίασης αλλά απαιτεί περισσότερες θέσεις μνήμης. Ωστόσο με τη μέθοδο αυτή απαιτείται ο θεωρητικά μικρότερος δυνατός χρόνος για την υλοποίηση της παραπάνω επανάληψης. Σκοπός της μεθόδου αυτής είναι για κάθε κόμβο προέλευσης i να εξετάζονται μόνο οι σύνδεσμοι που εξέρχονται από αυτόν και όχι όλοι οι σύνδεσμοι του δικτύου. Για το λόγο αυτό θεωρείται αρχικά ότι από κάθε σύνδεσμο $i \epsilon N$ δεν εξέρχεται κανένας κόμβος και εάν η συνάρτηση R(i) ισούται με τον αριθμό των συνδέσμων που εξέρχονται από τον κόμβο i, τότε $R(i) = 0, \forall i \in \{1,N\}$.

Εάν οι σύνδεσμοι του δικτύου είναι αποθηκευμένοι σε μία Λίστα Γειτνίασης, τότε για τον πρώτο σύνδεσμο $e \epsilon E$ που βρίσκεται στη Λίστα αποθηκεύεται ο κόμβος προέλευσής του S(e) και ο κόμβος προορισμού του E(e). Επίσης από τον κόμβο S(e) εξέρχεται πλέον ένας σύνδεσμος, άρα $R(S(e)) = R(S(e)) + 1 = 0 + 1 = 1$. Για να αποθηκευτεί ο πρώτος σύνδεσμος $e \epsilon E$ που εξέρχεται από τον κόμβο S(e) χρησιμοποιείται ένας πίνακας : r(S(e), R(S(e))) = e. Αυτή η σχέση σημαίνει ότι ο πρώτος σύνδεσμος R(S(e)) = 1 ο οποίος εξέρχεται από τον κόμβο S(e) είναι ο σύνδεσμος e. Η παραπάνω διαδικασία επαναλαμβάνεται για όλους τους συνδέσμους $e \epsilon E$, οπότε είναι γνωστός πλέον ο αριθμός των συνδέσμων που εξέρχονται από κάθε κόμβο i και ισούται με την τιμή R(i) αλλά και κάθε σύνδεσμος που εξέρχεται από κάθε κόμβο i και είναι ο r(i, j), όπου $j \epsilon \{1, R(i)\}$.

Έτσι με τη μέθοδο αυτή για την πραγματοποίηση της επανάληψης που εξετάζεται απαιτούνται R(i) συγκρίσεις, όσοι δηλαδή είναι και οι σύνδεσμοι που εξέρχονται από τον κόμβο i όπως παρουσιάζεται στη συνέχεια :

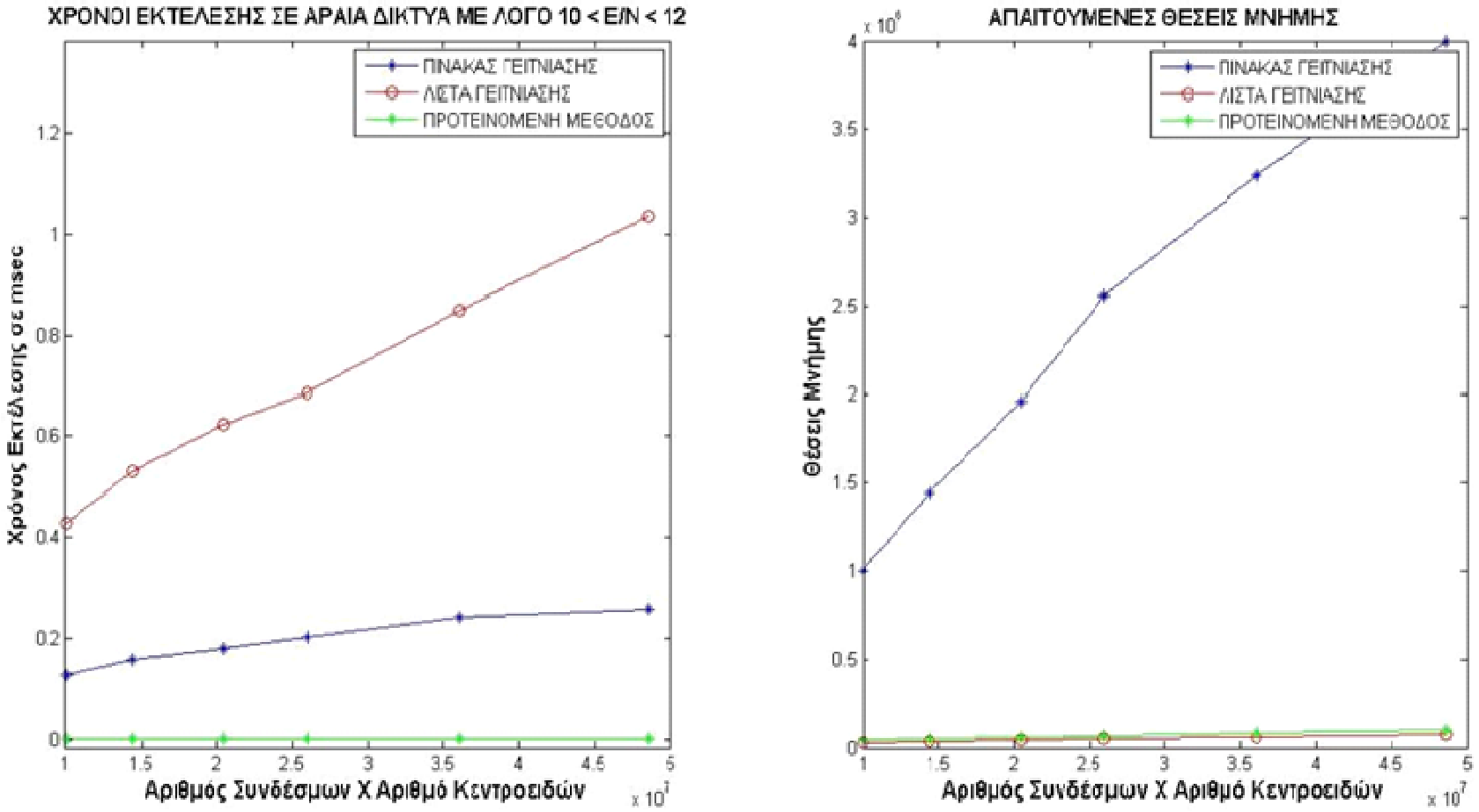
Για Κάθε j = 1,R(i) , όπου i ο κόμβος από τον οποίο εξέρχονται οι σύνδεσμοι :
Εάν D(E(r(i, j)) > D(S(r(i, j)) + w(r(i, j))
D(E(r(i, j)) ← D(S(r(i, j)) + w(r(i, j))
Τέλος Εάν
Τέλος Για Κάθε

Η παραπάνω μέθοδος οδηγεί στο μικρότερο θεωρητικό χρόνο εκτέλεσης της επανάληψης, ωστόσο οι απαιτήσεις σε θέσεις μνήμης είναι 4×E + N. Ωστόσο αν ο αριθμός των συνδέσμων στο δίκτυο είναι
$E \leq \frac{N(N-1)}{4}$, η μέθοδος αυτή απαιτεί λιγότερες θέσεις μνήμης από τη μέθοδο του Πίνακα Γειτνίασης και έχει θεωρητικά μικρότερο χρόνο εκτέλεσης. Αντίθετα σε καμία περίπτωση δεν αναμένεται να έχει μικρότερες απαιτήσεις σε θέσεις μνήμης σε σύγκριση με τη μέθοδο της Λίστας Γειτνίασης.

Στη συνέχεια πραγματοποιούνται εφαρμογές σε Η/Υ με επεξεργαστή Intel Core 2 Duo ταχύτητας 2.66 GHz και γίνονται συγκρίσεις στο χρόνο εκτέλεσης και στις απαιτούμενες θέσεις μνήμης των παραπάνω μεθόδων αναπαράστασης σε αραιά και πυκνά δίκτυα.

ΥΛΟΠΟΙΗΣΗ ΣΕ ΓΛΩΣΣΑ FORTRAN 95	ΠΙΝΑΚΑΣ ΓΕΙΤΝΙΑΣΗΣ	ΛΙΣΤΑ ΓΕΙΤΝΙΑΣΗΣ	ΠΡΟΤΕΙΝΟΜΕΝΗ ΜΕΘΟΔΟΣ	
ΔΙΚΤΥΟ Α:	0.1248	0.429	0.00059	→ *ΧΡΟΝΟΙ ΕΚΤΕΛΕΣΗΣ ΣΕ* ***msec***
ΚΕΝΤΡΟΕΙΔΗ = 1000	1000000	30030	41040	→ *ΑΠΑΙΤΟΥΜΕΝΕΣ ΘΕΣΕΙΣ ΜΝΗΜΗΣ*
ΣΥΝΔΕΣΜΟΙ = 10010	1000	10010	~10	→ *ΒΑΣΙΚΕΣ ΠΡΑΞΕΙΣ ΣΕ ΚΑΘΕ ΕΠΑΝΑΛΗΨΗ*
ΔΙΚΤΥΟ Β:	0.156001	0.530404	0.00059	
ΚΕΝΤΡΟΕΙΔΗ = 1200	1440000	36000	49200	
ΣΥΝΔΕΣΜΟΙ = 12000	1200	12000	~10	
ΔΙΚΤΥΟ Γ:	0.17940115	0.624	0.00059	
ΚΕΝΤΡΟΕΙΔΗ = 1400	1960000	43830	59840	
ΣΥΝΔΕΣΜΟΙ = 14610	1200	14610	~10	
ΔΙΚΤΥΟ Δ:	0.2028013	0.686405	0.00059	
ΚΕΝΤΡΟΕΙΔΗ = 1600	2560000	48630	66440	
ΣΥΝΔΕΣΜΟΙ = 16210	1600	16210	~10	
ΔΙΚΤΥΟ Ε:	0.2418015	0.8502	0.000624	
ΚΕΝΤΡΟΕΙΔΗ = 1800	3240000	60000	81800	
ΣΥΝΔΕΣΜΟΙ = 20000	1800	20000	~11	
ΔΙΚΤΥΟ ΣΤ:	0.25740165	1.0374067	0.000659	
ΚΕΝΤΡΟΕΙΔΗ = 2000	4000000	72870	99160	
ΣΥΝΔΕΣΜΟΙ = 24290	2000	24290	~12	

Πίνακας 5.3.Αποτελέσματα Εφαρμογής των τριών μεθόδων αναπαράστασης συγκοινωνιακών δικτύων σε αραιά δίκτυα



Διάγραμμα 5.1.Σύγκριση Χρόνων Εκτέλεσης και απαιτούμενων θέσεων μνήμης σε αραιά δίκτυα με E/N~10

Από τις παραπάνω εφαρμογές σε αραιά δίκτυα όπου ο αριθμός των συνδέσμων είναι περίπου δεκαπλάσιος από τον αριθμό των κόμβων προκύπτουν τα ακόλουθα συμπεράσματα :

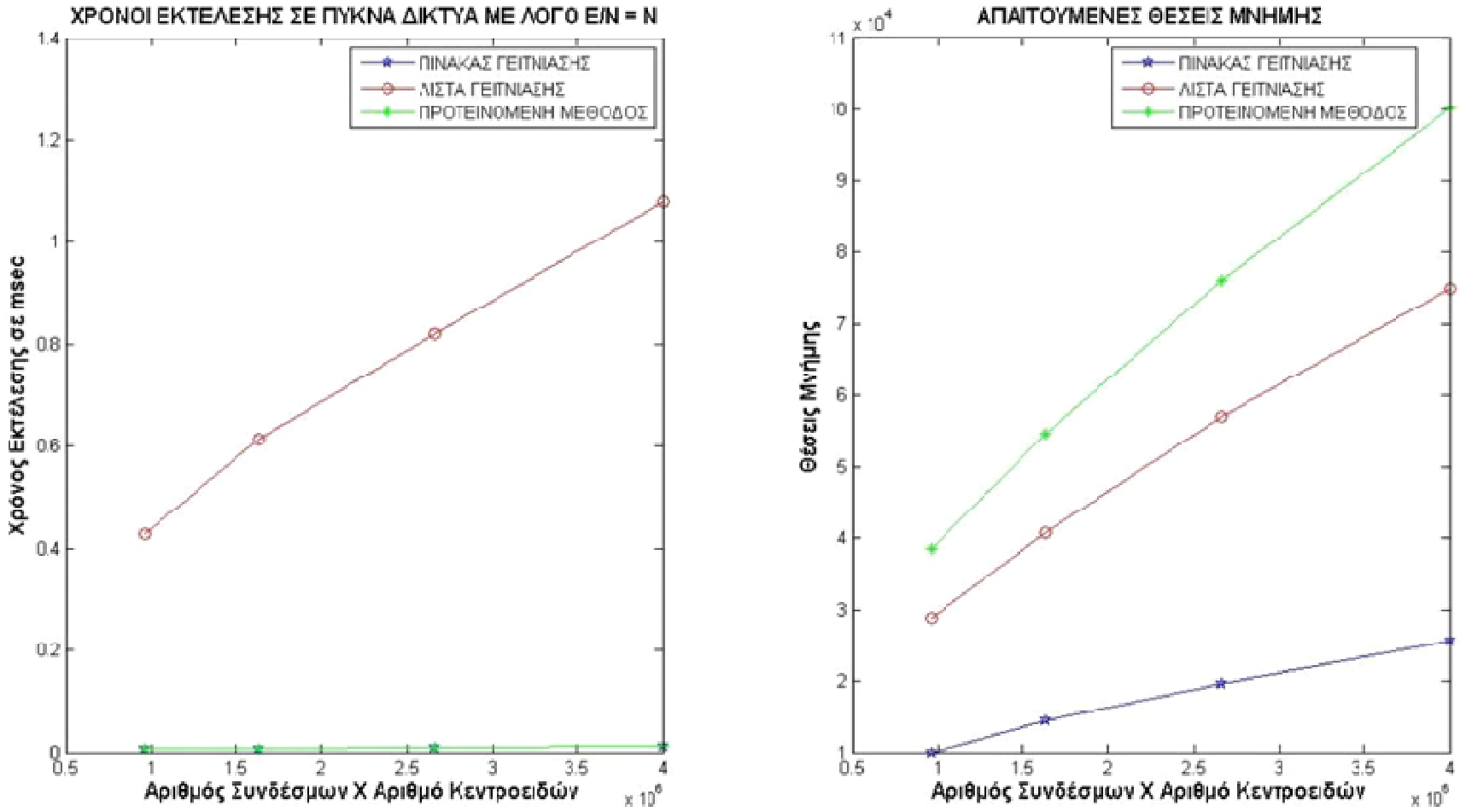
Είναι εμφανές ότι η προτεινόμενη μέθοδος αναπαράστασης των συγκοινωνιακών δικτύων προσφέρει πολύ μικρότερους χρόνους εκτέλεσης σε σχέση με τις άλλες δύο μεθόδους. Πιο συγκεκριμένα ο χρόνος εκτέλεσης ενός αλγορίθμου με χρήση αυτής της μεθόδου μπορεί να είναι μειωμένος έως και 300 φορές σε σύγκριση με την αναπαράσταση ενός δικτύου με Πίνακα Γειτνίασης. Η μέθοδος της Λίστας Γειτνίασης παρουσιάζει τους πιο αργούς χρόνους εκτέλεσης που μπορεί να είναι μέχρι και τέσσερις φορές μεγαλύτεροι από τους αντίστοιχους χρόνους μέσω της μεθόδου του Πίνακα Γειτνίασης. Ωστόσο η χρήση της Λίστας Γειτνίασης μπορεί να προτιμηθεί αντί του Πίνακα Γειτνίασης διότι δεσμεύει έως και πενήντα φορές λιγότερες θέσεις μνήμης.

Σε ένα σύνηθες συγκοινωνιακό δίκτυο ο αριθμός των συνδέσμων δεν υπερβαίνει το οκταπλάσιο του αριθμού των κόμβων. Για το λόγο αυτό είναι προτιμότερο να γίνεται αναπαράσταση των συγκοινωνιακών δικτύων σύμφωνα με την προτιμότερη μέθοδο ή με τη μέθοδο της Λίστας Γειτνίασης παρά τους επιβαρυμένους χρόνους εκτέλεσης που αυτή συνεπάγεται.

Στη συνέχεια παρουσιάζονται ενδεικτικά και τέσσερις εφαρμογές σε πολύ πυκνά συγκοινωνιακά δίκτυα όπου ο αριθμός των συνδέσμων προσεγγίζει το τετράγωνο του αριθμού των κόμβων. Τα δίκτυα αυτά βέβαια είναι ιδεατά και δε μπορούν να υλοποιηθούν στην πράξη. Τα αποτελέσματα δίνονται στον ακόλουθο πίνακα.

ΥΛΟΠΟΙΗΣΗ ΣΕ ΓΛΩΣΣΑ FORTRAN 95	ΠΙΝΑΚΑΣ ΓΕΙΤΝΙΑΣΗΣ	ΛΙΣΤΑ ΓΕΙΤΝΙΑΣΗΣ	ΠΡΟΤΕΙΝΟΜΕΝΗ ΜΕΘΟΔΟΣ	
ΔΙΚΤΥΟ Α:	0.005327	0.428	0.00532	→ *ΧΡΟΝΟΙ ΕΚΤΕΛΕΣΗΣ ΣΕ* ***msec***
ΚΕΝΤΡΟΕΙΔΗ = 100	10000	28800	38500	→ *ΑΠΑΙΤΟΥΜΕΝΕΣ ΘΕΣΕΙΣ ΜΝΗΜΗΣ*
ΣΥΝΔΕΣΜΟΙ = 9600	100	9600	~96	→ *ΒΑΣΙΚΕΣ ΠΡΑΞΕΙΣ ΣΕ ΚΑΘΕ ΕΠΑΝΑΛΗΨΗ*
ΔΙΚΤΥΟ Β:	0.00784	0.614	0.00781	
ΚΕΝΤΡΟΕΙΔΗ = 120	14400	40800	54520	
ΣΥΝΔΕΣΜΟΙ = 13600	120	13600	~113	
ΔΙΚΤΥΟ Γ:	0.009272	0.8213	0.009272	
ΚΕΝΤΡΟΕΙΔΗ = 140	19600	57000	76140	
ΣΥΝΔΕΣΜΟΙ = 19000	140	19000	~136	
ΔΙΚΤΥΟ Δ:	0.01113	1.0793	0.011129	
ΚΕΝΤΡΟΕΙΔΗ = 160	25600	75000	100160	
ΣΥΝΔΕΣΜΟΙ = 25000	160	25000	~156	

Πίνακας 5.4.Αποτελέσματα Εφαρμογής των τριών μεθόδων αναπαράστασης συγκοινωνιακών δικτύων σε πυκνά δίκτυα



Διάγραμμα 5.2.Σύγκριση Χρόνων Εκτέλεσης και απαιτούμενων θέσεων μνήμης σε πυκνά δίκτυα με Ε/Ν~Ν

5.9.ΜΕΘΟΔΟΣ ΚΑΤΑΚΕΡΜΑΤΙΣΜΟΥ ΔΙΚΤΥΩΝ

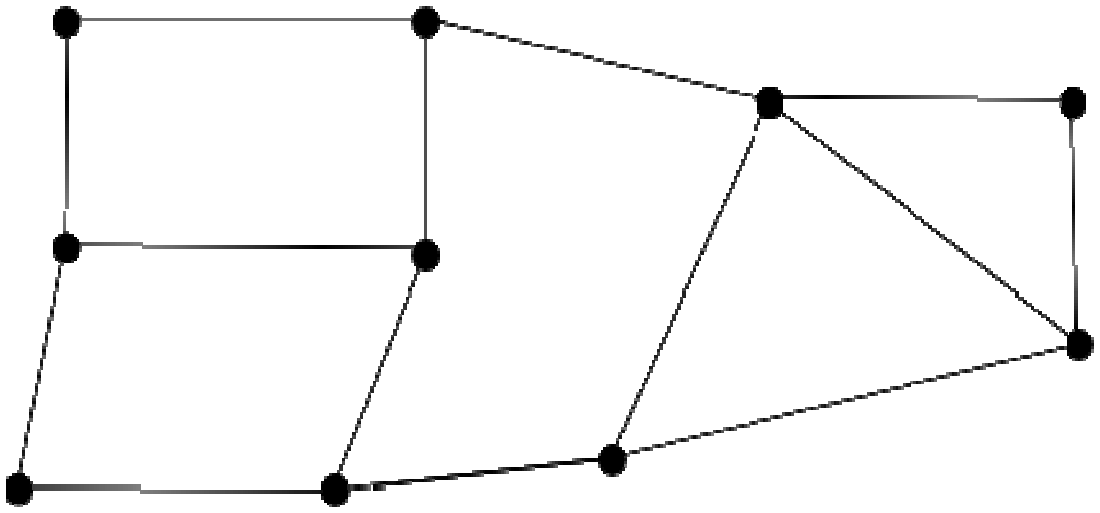
Η επιλογή της βέλτιστης διαδρομής σε ένα πραγματικό δίκτυο το οποίο περιλαμβάνει εκατομμύρια κόμβους και συνδέσμους μπορεί να απαιτεί σημαντικό χρονικό διάστημα αν υπολογιστεί με τους αλγορίθμους που παρουσιάστηκαν ήδη. Σε αυτή την ενότητα θα εξεταστεί η δυνατότητα περιορισμού του συγκοινωνιακού δικτύου ώστε να επικεντρώνεται η διαδικασία στο δίκτυο της περιοχής μελέτης που περιέχει τους κόμβους r, s. Για να γίνει αυτό πρέπει το δίκτυο να κατακερματιστεί σε μικρότερα δίκτυα και να παραμένει δυνατό να βρεθεί η βέλτιστη διαδρομή μεταξύ δύο κόμβων.

Ένα δίκτυο μπορεί να παρουσιαστεί ως γράφημα, έτσι από αυτό μπορούν να προκύψουν περισσότερα υπογραφήματα. Κάθε κόμβος του δικτύου πρέπει να περιέχεται σε ένα μόνο υπογράφημα. Όλοι οι σύνδεσμοι που βρίσκονται εντός ενός υπογραφήματος θα ονομάζονται εσωτερικοί σύνδεσμοι. Οι σύνδεσμοι που συνδέουν κόμβους από διαφορετικά υπογραφήματα θα καλούνται συνοριακοί σύνδεσμοι και οι κόμβοι που συνδέονται μέσω αυτών των συνδέσμων συνοριακοί κόμβοι. Όλοι οι υπόλοιποι κόμβοι καλούνται εσωτερικοί κόμβοι.

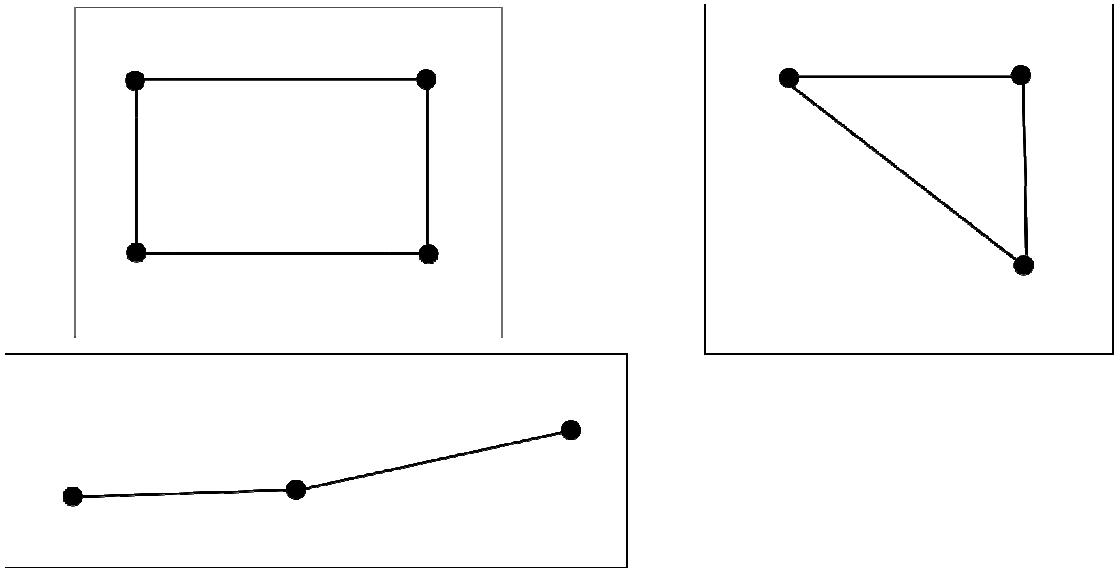
Σύμφωνα με τα παραπάνω εάν G = {N, E} είναι ένα γράφημα, τότε αυτό αποτελείται από υπογραφήματα $\{C_1, \dots, C_k\}$ όπου $C_i = \{N_i, E_i\}$, $\forall i \in \{1, k\}$.

Για τα υπογραφήματα ισχύει $N_i \cap N_j = \varnothing$, $\forall i, j \in \{1, k\}$ και $i \neq j$.
Επίσης ισχύει $\bigcup_{i=1}^{k} N_i = N$
Το σύνολο που περιέχει τους συνοριακούς συνδέσμους είναι : $E_b = \bigcup_{i=1}^{k} E_i - E$
Οι συνοριακοί κόμβοι ενός υπογραφήματος C_i είναι $N_{b,i} = \{u \in N_i \mid e \in E_b : u = S(e)$ ή $u = E(e)\}$

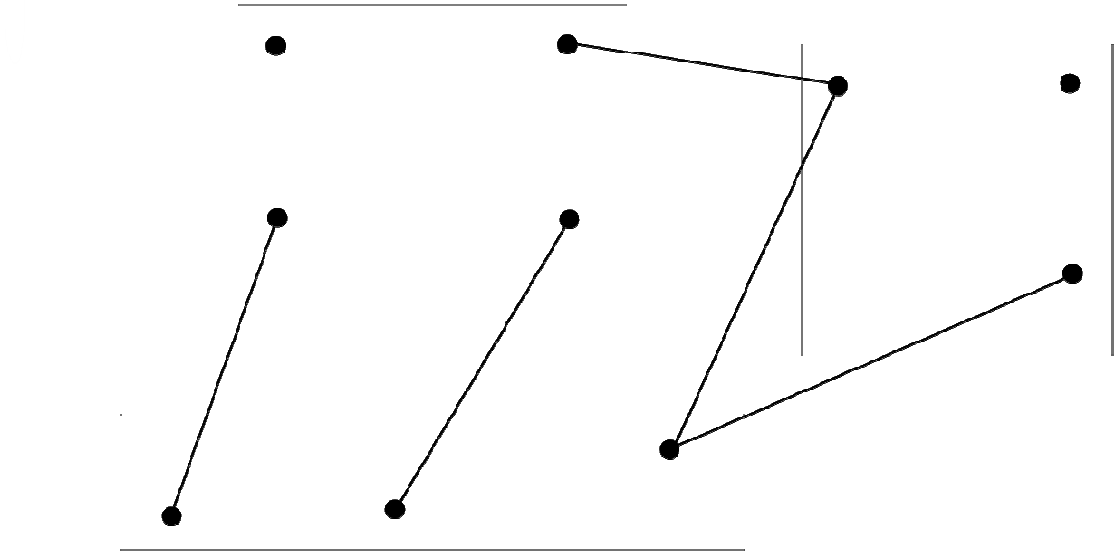
Τα παραπάνω γίνονται εμφανή στα ακόλουθα σχήματα.



Σχήμα 5.18.Γράφημα αναπαράστασης συγκοινωνιακού δικτύου



Σχήμα 5.19.Διαίρεση του γραφήματος σε υπογραφήματα



Σχήμα 5.20.Συνοριακοί Σύνδεσμοι και Συνοριακοί Κόμβοι υπογραφημάτων

Το πιο σύνηθες κριτήριο κατά τον κατακερματισμό του δικτύου είναι η δημιουργία υπογραφημάτων ώστε ο αριθμός των συνοριακών συνδέσμων : $E_b = \cup_{i=1}^{k} E_i$ – Ε να είναι ο ελάχιστος δυνατός. Το κριτήριο αυτό επιλέγεται ώστε να μειωθεί το κόστος επικοινωνίας μεταξύ των υπογραφημάτων. Ο δεύτερος στόχος είναι τα παραγόμενα υπογραφήματα να έχουν τον παρόμοιο αριθμό κόμβων. Στη συνέχεια παρουσιάζεται ένας αλγόριθμος υποδιαίρεσης του δικτύου που προτάθηκε από τους Kernigham and Lin (1970). Σκοπός του αλγορίθμου είναι η διαίρεση των κόμβων ενός δικτύου σε k υπογραφήματα με στόχο την ελαχιστοποίηση του αριθμού των συνοριακών κόμβων. Κατά την αρχικοποίηση είναι γνωστός ο αριθμός των κόμβων N και ο αριθμός των υπογραφημάτων k. Ο αλγόριθμος ξεκινά με την υποδιαίρεση του δικτύου σε δύο υπογραφήματα (cells) μέσω ενός τυχαίου κατακερματισμού του δικτύου. Σε κάθε βήμα επανάληψης ο αλγόριθμος ανταλλάσει υποσύνολα που περιέχουν των ίδιο αριθμό κόμβων μεταξύ των δύο υπογραφημάτων A, B ώστε να μειωθεί ο αριθμός των

συνοριακών συνδέσμων. Ο αλγόριθμος τερματίζεται όταν δεν είναι πλέον δυνατό να μειωθεί περαιτέρω ο αριθμός των συνοριακών συνδέσμων μέσω ανταλλαγής κόμβων μεταξύ των δύο υπογραφημάτων. Μπορεί να τερματιστεί επίσης και έπειτα από έναν προκαθορισμένο αριθμό επαναλήψεων.

Το δίκτυο μπορεί να υποδιαιρεθεί και σε περισσότερα από δύο υπογραφήματα εφαρμόζοντας επαναληπτικά την παραπάνω διαδικασία. Το ερώτημα είναι πως θα επιλεγούν οι κόμβοι που θα μεταφερθούν από το ένα υπογράφημα (A) στο άλλο (B). Όπως έχει αναφερθεί ήδη αυτή η επιλογή βασίζεται στο κέρδος που υπάρχει όταν ένας κόμβος u μεταφέρεται από το υπογράφημα A στο υπογράφημα B. Το κέρδος ορίζεται ως η μείωση του αριθμού των συνοριακών συνδέσμων κατά τη μεταφορά. Πιο αναλυτικά οι συνοριακοί σύνδεσμοι που συνέδεαν τον κόμβο u με τους κόμβους του υπογραφήματος B μετατρέπονται πλέον σε εσωτερικούς, ενώ οι εσωτερικοί σύνδεσμοι που συνέδεαν τον κόμβο u με τους κόμβους του υπογραφήματος A μετατρέπονται σε συνοριακούς. Εάν η παραπάνω διαφορά είναι θετική, τότε το κέρδος κατά τη μεταφορά του κόμβου u είναι θετικό και ο κόμβος μεταφέρεται στο υπογράφημα B. Στη συνέχεια από το υπογράφημα B αναζητείται ένας κόμβος ώστε να μεταφερθεί στο υπογράφημα A με την ίδια διαδικασία ώστε ο αριθμός κόμβων που περιέχει κάθε υπογράφημα να παραμένει σταθερός.

Μία σημαντική ιδέα στην παραπάνω μέθοδο είναι ότι ο αλγόριθμος μπορεί να μην τερματίσει ακόμα και εάν δεν υπάρχει κανένας σύνδεσμος με θετικό κέρδος μεταφοράς. Βέβαια η παρούσα κατάσταση των υπογραφημάτων αποθηκεύεται προσωρινά και οι επαναλήψεις συνεχίζονται μεταφέροντας κόμβους με αρνητικό κέρδος αναμένοντας ότι έπειτα από κάποιες επαναλήψεις θα προκύψουν κόμβοι με θετικό κέρδος μεταφοράς. Εάν δε συμβεί αυτό η μορφή των υπογραφημάτων επανέρχεται στην κατάσταση που είχε αποθηκευτεί προσωρινά. Ο χρόνος εκτέλεσης του παρπάνω αλγορίθμου είναι $O(N^2 \log N)$. Μία σημαντική τροποποίηση αυτού του αλγορίθμου προτάθηκε από τους Fiduccia and Mattheyses (1982). Στον αλγόριθμο που πρότειναν τα κέρδη από τη μεταφορά κάθε συνδέσμου αναβαθμίζονται σύμφωνα με τα νέα στοιχεία που προκύπτουν έπειτα από κάθε επανάληψη. Έτσι σε κάθε επανάληψη επιλέγονται οι κόμβοι με το μεγαλύτερο κέρδος μεταφοράς. Ο αλγόριθμος αυτός έχει χρόνος εκτέλεσης $O(E)$.

5.10.ΑΛΓΟΡΙΘΜΟΙ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ

5.10.1.Ο ΑΛΓΟΡΙΜΘΜΟΣ CHABINI

Όπως έχει αναφερθεί ήδη, αν και έχει δημοσιευθεί πλήθος εργασιών για την εύρεση της βέλτιστης διαδρομής σε δίκτυα με στατικά βάρη συνδέσμων, δεν έχει γίνει το ίδιο και για τα προβλήματα σε δίκτυα με δυναμικά βάρη. Μία από τις εργασίες πάνω σε αυτό το πρόβλημα είναι αυτή του Chabini (1998). Αν και υπάρχει εκτενής αναφορά στα δυναμικά δίκτυα σε επόμενο κεφάλαιο θα γίνει στη συνέχεια μία παρουσίαση αυτής της μεθόδου. Η μέθοδος αυτή είναι ένας αλγόριθμος για την εύρεση των ελάχιστων διαδρομών από όλους τους κόμβους προς όλους τους κόμβους (All Pairs Shortest Path problem) σε δίκτυα με δυναμικό βάρος συνδέσμων, το οποίο μεταβάλλεται με το χρόνο. Ο αλγόριθμος αυτός χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Floyd-Warshall ή κάποιον άλλο αλγόριθμο επίλυσης του APSPP. Η ονομασία του είναι DOT (Decreasing Order of Time) που αντικατοπτρίζει την κύρια ιδέα της εφαρμογής του. Θεωρώντας διακριτές τιμές του συνολικού χρόνου T κατά τις οποίες αλλάζουν τα βάρη συνδέσμων στο πρόβλημα : t ={1, ….., M-1} μπορούμε να υποθέσουμε ότι για χρόνο μεγαλύτερο από M-1 το πρόβλημα εκφυλίζεται σε πρόβλημα με στατικά βάρη συνδέσμων. Έτσι μπορεί να παρουσιαστεί ο αλγόριθμος :

ΑΛΓΟΡΙΘΜΟΣ CHABINI

Βήμα 1(Αρχικοποιήσεις)

Έστω T ο συνολικός χρόνος στον οποίο αναφέρεται το πρόβλημα και M ο χρόνος στον οποίο αλλάζουν τα δυναμικά βάρη συνδέσμων, τότε t={0, M, …., T}.

w(i)(j)(k) το βάρος μετακίνησης από τον κόμβο i → j με i, j ϵ N σε χρόνο k ϵ t, όπου w(i)(j)(k) ← +∞ εάν δεν υπάρχει σύνδεση μεταξύ των κόμβων.

Έστω D (i)(j)(k) η συντομότερη απόσταση από τον κόμβο i → j με i, j ϵ N σε χρόνο k ϵ t όπου κατά την αρχικοποίηση δηλώνεται ως :

D (i)(j)(k) ← w(i)(j)(k)

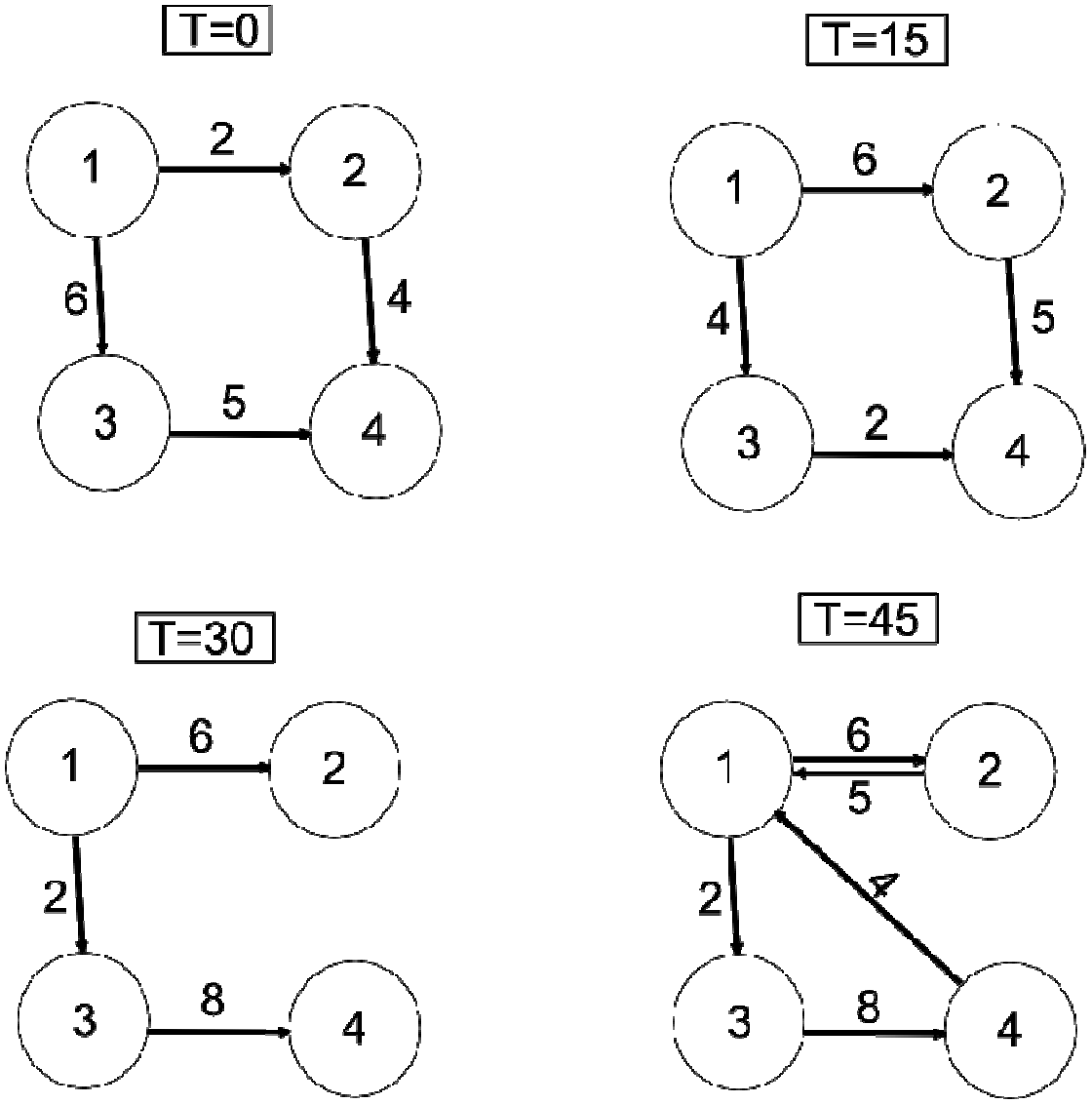
Βήμα 2(Κύριο Μέρος)

```
Για Κάθε k = T,0  // Χρήση Floyd - Warshall ή οποιουδήποτε άλλου APSPP
    Για Κάθε  b = 1,N
        Για Κάθε i = 1,N
            Για  Κάθε j = 1,N
               Εάν  D( i )(  j )( k ) > D( i )( b )( k ) + D( b )( j )( k )
               D( i )(  j )( k ) ← D( i )( b )( k ) + D( b )( j )( k )
               Τέλος Εάν
            Τέλος Για Κάθε
        Τέλος Για Κάθε
    Τέλος Για Κάθε
Τέλος Για Κάθε
```

Ονομασία αλγορίθμου	**Decreasing Order Time**
Λειτουργία του	Εύρεση ελάχιστων διαδρομών μεταξύ όλων των κόμβων του δικτύου (APSP)
Δίκτυο Εφαρμογής	Δυναμικοί Χρόνοι μετακίνησης μέσω συνδέσμων. Ικανοποιείται η συνθήκη FIFO
Αναμονή σε κόμβους	Επιτρεπτή

Πίνακας 5.5.Χαρακτηριστικά αλγορίθμου DOT

Έστω ένα παράδειγμα δικτύου που αποτελείται από τέσσερις ζώνες, όπου τα βάρη μετακίνησης των συνδέσμων αλλάζουν κάθε 15 λεπτά και σταθεροποιούνται μετά τα 45 λεπτά. Αυτό μπορεί να συμβαίνει για διάφορους λόγους, όπως κίνηση, έργα, περίοδοι αιχμής κ.α. Σχηματικά έχουμε :



Σχήμα 5.21.Δυναμικό Δίκτυο με αλλαγές στα βάρη των συνδέσμων

Στη συνέχεια προγραμματίζεται ο αλγόριθμος DOT σε γλώσσα Java :

```
/*ΚΑΤΕΥΘΥΝΣΗ ΣΥΓΚΟΙΝΩΝΙΟΛΟΓΟΥ ΜΗΧΑΝΙΚΟΥ
 *ΕΡΓΑΣΙΑ ΣΤΟ ΘΕΜΑ : ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΑΛΓΟΡΙΘΜΟΥ DOT
 *DYNAMIC NETWORKS
 *ALL PAIRS SHORTEST PATH PROBLEMS
 *ΗΜΕΡΟΜΗΝΙΑ : ΑΠΡΙΛΙΟΣ 2010
 *ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ : JAVA
 */
import java.util.Scanner;
public class DOT {
Scanner input = new Scanner (System.in);
int N=input.nextInt();//N:Ο ΑΡΙΘΜΟΣ ΤΩΝ ΚΟΜΒΩΝ
int T=input.nextInt();//T:ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΣΤΟΝ ΟΠΟΙΟ ΑΝΑΦΕΡΕΤΑΙ ΤΟ
ΠΡΟΒΛΗΜΑ
int M=input.nextInt();//M:ΧΡΟΝΙΚΟΣ ΔΙΑΧΩΡΙΣΜΟΣ ΟΠΟΥ ΑΛΛΑΖΟΥΝ ΤΑ ΒΑΡΗ
ΣΥΝΔΕΣΜΩΝ
double[][][] Weight=new double[N+1][N+1][T+1];//Weight[k][i][j]:ΤΟ ΒΑΡΟΣ
ΓΙΑ
//ΜΕΤΑΚΙΝΗΣΗ ΜΕΤΑΞΥ ΤΩΝ ΚΟΜΒΩΝ i,j ΤΗ ΧΡΟΝΙΚΗ ΣΤΙΓΜΗ k
double[][][] Distance=new double[N+1][N+1][T+1];//Distance[k][i][j]:Η
ΕΛΑΧΙΣΤΗ
//ΑΠΟΣΤΑΣΗ ΜΕΤΑΞΥ ΤΩΝ ΚΟΜΒΩΝ i,j ΤΗ ΧΡΟΝΙΚΗ ΣΤΙΓΜΗ k

protected void Initialization(){
   for(int k=0; k<=T; k=k+M)
   for (int i=1; i<=N; i++){
       for (int j=1;j<=N;j=j+1){
          Weight[i][j][k] = input.nextDouble();
          Distance[i][j][k] = Weight[i][j][k];
           }
    }
}//Initialization

public void DOT_Algorithm(){
  for  (int k=T;k>=0;k=k-M){
        System.out.println(" ΓΙΑ ΧΡΟΝΟ = "+k);
System.out.println("ΚΟΜΒΟΣ ΠΡΟΕΛΕΥΣΗΣ"+"  "+"ΚΟΜΒΟΣ ΠΡΟΟΡΙΣΜΟΥ"+"
"+"ΕΛΑΧΙΣΤΗ ΑΠΟΣΤΑΣΗ");
      for (int r=1;r<=N;r++){
            for (int i=1;i<=N;i++){
                  for (int j=1;j<=N;j++){
if(Distance[i][j][k] > Distance[i][r][k] + Distance[r][j][k]){
Distance[i][j][k] = Distance[i][r][k] + Distance[r][j][k];
            }
                  }}}
      for (int i=1;i<=N;i++){
            for (int j=1;j<=N;j++){
System.out.println("      "+i+"                    "+j+"
"+Distance[i][j]);}}
  }}//DOT_Algorithm

public static void main(String args[]){
      DOT dot = new DOT();
      dot.Initialization();
      dot.Initialization();
      }//main
}//DOT
```

--ΑΠΟΤΕΛΕΣΜΑΤΑ--

ΓΙΑ ΧΡΟΝΟ = 45

ΚΟΜΒΟΣ ΠΡΟΕΛΕΥΣΗΣ	ΚΟΜΒΟΣ ΠΡΟΟΡΙΣΜΟΥ	ΕΛΑΧΙΣΤΗ ΑΠΟΣΤΑΣΗ
1	1	0.0
1	2	6.0
1	3	3.0
1	4	8.0
2	1	5.0
2	2	0.0
2	3	8.0
2	4	13.0
3	1	9.0
3	2	15.0
3	3	0.0
3	4	5.0
4	1	4.0
4	2	10.0
4	3	7.0
4	4	0.0

ΓΙΑ ΧΡΟΝΟ = 30

ΚΟΜΒΟΣ ΠΡΟΕΛΕΥΣΗΣ	ΚΟΜΒΟΣ ΠΡΟΟΡΙΣΜΟΥ	ΕΛΑΧΙΣΤΗ ΑΠΟΣΤΑΣΗ
1	1	0.0
1	2	6.0
1	3	2.0
1	4	10.0
2	1	Infinity
2	2	0.0
2	3	Infinity
2	4	Infinity
3	1	Infinity
3	2	Infinity
3	3	0.0
3	4	8.0
4	1	Infinity
4	2	Infinity
4	3	Infinity
4	4	0.0

ΓΙΑ ΧΡΟΝΟ = 15

ΚΟΜΒΟΣ ΠΡΟΕΛΕΥΣΗΣ	ΚΟΜΒΟΣ ΠΡΟΟΡΙΣΜΟΥ	ΕΛΑΧΙΣΤΗ ΑΠΟΣΤΑΣΗ
1	1	0.0
1	2	6.0
1	3	4.0
1	4	6.0
2	1	Infinity
2	2	0.0
2	3	Infinity
2	4	5.0
3	1	Infinity
3	2	Infinity
3	3	0.0
3	4	2.0
4	1	Infinity
4	2	Infinity
4	3	Infinity
4	4	0.0

ΓΙΑ ΧΡΟΝΟ = 0

ΚΟΜΒΟΣ ΠΡΟΕΛΕΥΣΗΣ	ΚΟΜΒΟΣ ΠΡΟΟΡΙΣΜΟΥ	ΕΛΑΧΙΣΤΗ ΑΠΟΣΤΑΣΗ
1	1	0.0
1	2	2.0
1	3	6.0
1	4	6.0
2	1	Infinity
2	2	0.0
2	3	Infinity
2	4	4.0
3	1	Infinity
3	2	Infinity
3	3	0.0
3	4	5.0
4	1	Infinity
4	2	Infinity
4	3	Infinity
4	4	0.0

5.10.2.ΠΛΗΡΩΣ ΔΥΝΑΜΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΣΥΧΝΟΥΣ ΕΠΑΝΥΠΟΛΟΓΙΣΜΟΥΣ ΤΟΥ ΔΙΚΤΥΟΥ ΛΟΓΩ ΜΕΤΑΒΟΛΩΝ ΤΗΣ ΚΥΚΛΟΦΟΡΙΑΚΗΣ ΡΟΗΣ

Στις προηγούμενες ενότητες (5.2., 5.3.) έχουν αναπτύχθηκαν διάφορες μέθοδοι για τον υπολογισμό της βέλτιστης διαδρομής μεταξύ δύο κόμβων προέλευσης-προορισμού. Όταν διαδραματίζονται συνεχείς μεταβολές στα βάρη των συνδέσμων ενός δικτύου μπορούν να χρησιμοποιηθούν ορισμένες από αυτές τις μεθόδους για τον υπολογισμό της νέας βέλτιστης διαδρομής. Κατά τη διαδικασία αυτή επιτελείται επανυπολογισμός του δικτύου από μηδενική αφετηρία με νέα βάρη συνδέσμων. Ένα πρόσφατο παράδειγμα είναι η χρήση του αλγόριθμου Floyd-Warshall, ο οποίος εφαρμόζεται σε στατικά προβλήματα, ως υπορουτίνα για την αντιμετώπιση δυναμικών προβλημάτων. Αποτελεί ερώτημα όμως το κατά πόσο είναι αποδοτικό να χρησιμοποιούνται μέθοδοι που αντιμετωπίζουν στατικά προβλήματα σε δυναμικά δίκτυα καθώς η συνεχής ανάγκη για επανυπολογισμούς δε μπορεί να αντιμετωπιστεί από μεθόδους που δε λαμβάνουν υπόψη την παρούσα κατάσταση και επιλύουν το πρόβλημα από την αρχή. Στο σημείο αυτό αξίζει να σημειωθεί πως η έρευνα προς αυτή την κατεύθυνση είναι ελλιπής. Αυτό οφείλεται στη στατική αντιμετώπιση των δικτύων και στην έλλειψη επαρκούς πεδίου πληροφόρησης ώστε να επανατροφοδοτείται το σύστημα με νέα δεδομένα και να απαιτούνται συνεχείς επανυπολογισμοί. Στη συνέχεια προτείνεται ένας αλγόριθμος που προορίζεται για συνεχή επανυπολογισμό των διαδρομών.

Ο πρώτος πλήρως δυναμικός αλγόριθμος με προσανατολισμό τον συνεχή επανυπολογισμό διαδρομών προτάθηκε από τους Ramalingam and Reps (1996). Ο χρόνος εκτέλεσής του εξαρτάται από μία παράμετρο $\|\delta\|$, όπου :

$\|\delta\| = v + e$, όπου $v \epsilon N$ ο αριθμός των κόμβων που επηρεάζονται λόγω των μεταβολών στο δίκτυο και $e \epsilon E$ ο αριθμός των συνδέσμων για τους οποίους επηρεάζεται ο κόμβος προέλευσης ή προορισμού τους αντίστοιχα.

Πιο συγκεκριμένα ο αλγόριθμος διατηρεί ένα σύνολο SP(G) στο οποίο περιλαμβάνονται όλοι οι σύνδεσμοι του δικτύου G οι οποίοι ανήκουν σε τουλάχιστον μία διαδρομή ελαχίστου κόστους από τον κόμβο r προς όλους τους υπόλοιπους κόμβους του δικτύου. Εξετάζοντας αρχικά την εισαγωγή ενός νέου συνδέσμου (u,π) στο δίκτυο χρησιμοποιείται η μέθοδος Dijkstra στο υποσύνολο των κόμβων που επηρεάζεται από αυτήν τη μεταβολή. Εάν $D(u) + w(u,\pi) > D(\pi)$, τότε δε συντελείται καμία μεταβολή στο δίκτυο. Εάν όμως κάτι τέτοιο δεν ισχύει η επανάληψη συνεχίζεται από τον κόμβο π εξετάζοντας όλους τους συνδέσμους για τους οποίους αποτελεί κόμβο προέλευσης. Οι κόμβοι που επηρεάζονται διατάσσονται σε μία ουρά προτεραιότητας εκκινώντας από τον κόμβο με τη μικρότερη απόσταση από τον κόμβο w : $\{ x \mid D(x)-D(w) = \min \{D(i)-D(w)\}, \forall i \epsilon N\}$. Ο κόμβος x είναι πλέον ο κόμβος με την ελάχιστη απόσταση από τον κόμβο w. Στη συνέχεια ελέγχονται όλοι οι σύνδεσμοι με κόμβο προέλευσης τον κόμβο x : (x, y) και εάν το άθροισμα της απόσταση $r \rightarrow x$ με το βάρος του συνδέσμου (x, y) είναι μικρότερο από το D(y) :

απόσταση r → y, τότε ο κόμβος y είναι ένας από τους κόμβους που επηρεάζονται και αφενός αναβαθμίζεται το κόστος του σε D'(y) = D(x) + w(x,y) και αφετέρου εισέρχεται στην ουρά προτεραιότητας. Αυτή η διαδικασία εφαρμόζεται και εάν αντί για εισαγωγή νέου συνδέσμου συντελούταν αλλαγή στο βάρος ενός συνδέσμου που ανήκε ήδη στο δίκτυο.

Στη συνέχεια εξετάζεται η διαγραφή ενός συνδέσμου (u,π) από το δίκτυο. Σε αυτή την περίπτωση εντοπίζονται αρχικά οι κόμβοι που επηρεάζονται από τη διαγραφή αυτή. Οι κόμβοι αυτοί αποθηκεύονται σε ένα σύνολο Α. Ο πρώτος κόμβος προς εξέταση είναι ο κόμβος π. Ο κόμβος αυτός επηρεάζεται εάν και μόνο εάν ο σύνδεσμος (u,π) ανήκει στο σύνολο SP(G). Στη συνέχεια διαγράφονται από το σύνολο SP(G) όλοι οι σύνδεσμοι που έχουν κόμβο προέλευσης τον κόμβο u και οι κόμβοι προορισμού αυτών των συνδέσμων τοποθετούνται στο σύνολο Α. Στο τέλος αυτής της διαδικασίας έχουν δημιουργηθεί δύο ιδεατά υπογραφήματα και το πρόβλημα απαιτεί εφαρμογή του αλγορίθμου Dijkstra στο υπογράφημα Α. Στην ουσία δηλαδή δε λαμβάνεται υπόψη το υπογράφημα Β. Για το σκοπό αυτό θεωρείται νέος κόμβος προέλευσης r'. Συνεχίζοντας για κάθε σύνδεσμο (x, y) όπου x ϵ Α και y ϵ Β εισάγεται ιδεατός σύνδεσμος (r', y) με βάρος : w(r', y) = D(x) + w(x,y). Έτσι εφαρμόζεται ο αλγόριθμος Dijkstra όπως αναφέρθηκε προηγουμένως στο νέο υπογράφημα με νέα βάρη συνδέσμων και το πρόβλημα επιλύεται.

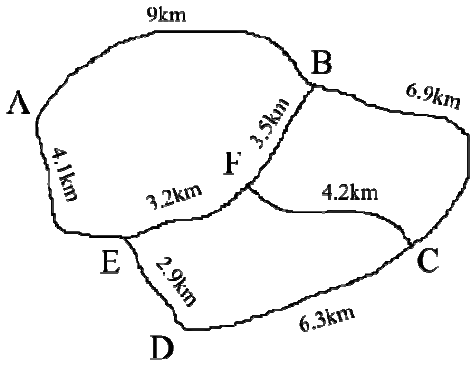
ΚΕΦΑΛΑΙΟ 6ο

ΚΑΤΑΜΕΡΙΣΜΟΣ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΣΤΑΤΙΚΑ ΔΙΚΤΥΑ ΜΕ ΧΡΗΣΗ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

6.1.ΔΡΟΜΟΛΟΓΗΣΗ ΣΕ ΣΤΑΤΙΚΑ ΔΙΚΤΥΑ

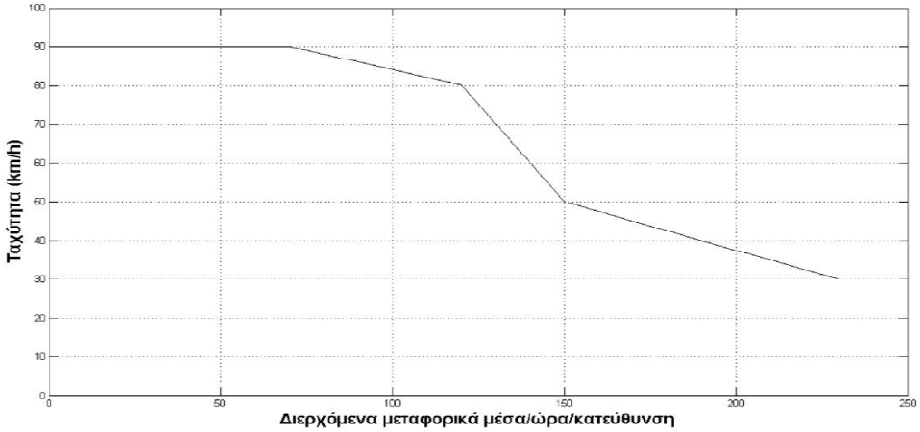
Σε πολλά συγκοινωνιακά δίκτυα η κυκλοφοριακή ροή είναι χαμηλή. Στα δίκτυα αυτά για να ισχύει η πρώτη αρχή του Wardrop (Ισορροπία χρήστη-User's equilibrium) απαιτείται όλοι οι χρόνοι διάνυσης που χρησιμοποιήθηκαν για τις μετακινήσεις ανάμεσα σε δύο σημεία προέλευσης-προορισμού να είναι ίσοι και μάλιστα μικρότεροι από οποιοδήποτε χρόνο οποιασδήποτε άλλης μη χρησιμοποιούμενης διαδρομής. Σε στατικά συγκοινωνιακά δίκτυα οι χρήστες καλούνται να επιλέξουν τη συντομότερη διαδρομή για να φθάσουν στον προορισμό τους ανάμεσα σε πολλές εναλλακτικές. Το πλεονέκτημα της δρομολόγησης σε στατικά δίκτυα είναι ότι δε λαμβάνεται υπόψη το φαινόμενο της κυκλοφοριακής συμφόρησης. Έτσι όσοι χρήστες κι αν ακολουθήσουν μία διαδρομή δεν αυξάνεται ο χρόνος διάνυσής της. Αυτό βέβαια αποτελεί απλοποιητική παραδοχή που καθιστά αυτό τον τύπο δρομολόγησης αναξιόπιστο και ευσταθεί μόνο βραχυπρόθεσμα, εκτός και αν λόγω χαμηλής φόρτισης δε μεταβάλλεται ουσιαστικά ο χρόνος διάνυσης των συνδέσμων.

Τα στατικά δίκτυα αποτελούν την απλούστερη περίπτωση για την υλοποίηση του καταμερισμού της ζήτησης. Σύμφωνα με τα παραπάνω εάν υπάρχει το μητρώο μετακινήσεων κάποιας χρονικής περιόδου, τότε οι μετακινήσεις κατανέμονται στο δίκτυο σε μόλις ένα στάδιο φόρτισης, αφού ο χρόνος διάνυσης των συνδέσμων παραμένει αμετάβλητος κατά τη φόρτισή τους. Η φόρτιση γίνεται με τη μέθοδο του όλα ή τίποτα με βάση την πρώτη αρχή του Wardrop. Έτσι όλοι οι χρήστες που μετακινούνται μεταξύ δύο σημείων προέλευσης προορισμού επιλέγουν τη συντομότερη διαδρομή για την υλοποίηση της μετακίνησής τους και οι σύνδεσμοι που αποτελούν αυτή τη διαδρομή παραλαμβάνουν το σύνολο της φόρτισης. Οι ροές που προκύπτουν στο δίκτυο με βάση την παραπάνω διαδικασία είναι γνωστές ως ροές ισορροπίας του χρήστη. Η παραπάνω διαδικασία παρουσιάζεται στα ακόλουθα σχήματα.



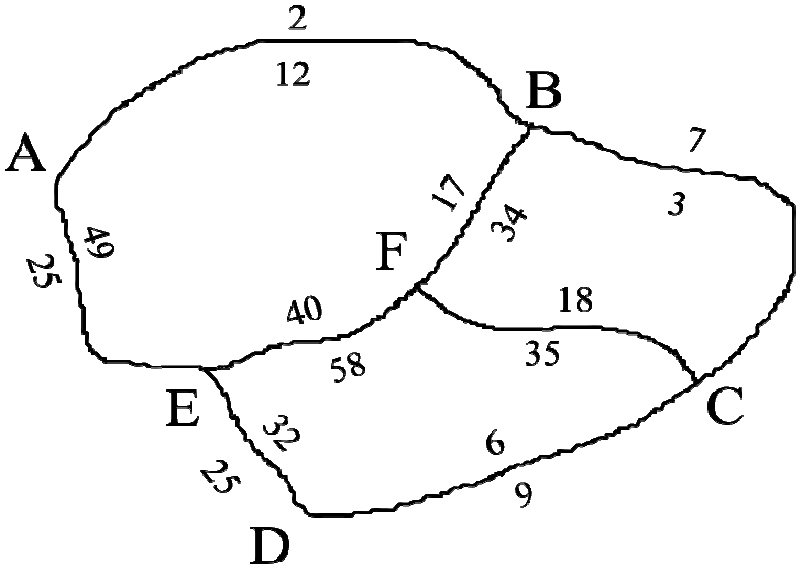
Μητρώο Μετακινήσεων

	A	B	C	D	E	F
A	-	12	7	9	6	3
B	2	-	3	4	8	5
C	6	7	-	6	7	5
D	8	12	9	-	4	8
E	31	9	17	7	-	2
F	4	13	11	5	6	-



Σχήμα 6.1.α) Χιλιομετρικές Αποστάσεις μεταξύ των κεντροειδών των ζωνών. β) Μητρώο Μετακινήσεων σε συγκεκριμένη χρονική περίοδο. γ) Μείωση της ταχύτητας διάνυσης συνδέσμων λόγω κυκλοφοριακής συμφόρησης.

Ο χρόνος διάνυσης για κάθε σύνδεσμο μεταβάλλεται εάν διέλθουν από αυτόν περισσότερα από 70 μεταφορικά μέσα/λωρίδα/ώρα όπως φαίνεται στο παραπάνω διάγραμμα. Εάν το δίκτυο φορτιστεί σύμφωνα με την 1η αρχή ισορροπίας του Wardrop δεν υπάρχει σύνδεσμος που να μεταβάλλεται ο χρόνος διάνυσής του και προκύπτουν οι φόρτοι συνδέσμων του σχήματος :



Σχήμα 6.2.Φόρτιση δικτύου σε κατάσταση ισορροπίας(Ροές Ισορροπίας Χρήστη)

Ο καταμερισμός της ζήτησης μπορεί να υλοποιηθεί με χρήση των συστημάτων πλοήγησης που παρέχουν πληροφόρηση στο χρήστη για τη συντομότερη διαδρομή που πρέπει να ακολουθήσει κατά την μετακίνησή του. Για το σκοπό αυτό απαιτούνται δεδομένα για το οδικό δίκτυο και τις γενικές συνθήκες κίνησης τα οποία παρέχονται συνήθως από μια βάση δεδομένων, που δεν ανανεώνεται συχνά.

Όπως είναι φανερό η χρησιμότητα των συστημάτων πλοήγησης στα στατικά δίκτυα είναι μεγάλη. Σε αντίθετη περίπτωση πολλοί χρήστες θα επιλέξουν εναλλακτικές διαδρομές με αυξημένο κόστος λόγω της έλλειψης πληροφόρησης, καθώς ο χρήστης δεν έχει τη δυνατότητα απαρίθμησης και σύγκρισης όλων των εναλλακτικών επιλογών του. Το πρόβλημα αυτό είναι εντονότερο σε πυκνά δίκτυα, στα οποία αυξάνονται οι διαθέσιμες εναλλακτικές επιλογές. Ως αποτέλεσμα καταναλώνεται επιπλέον ποσότητα ενέργειας και δε λαμβάνεται η βέλτιστη επιλογή από κάθε χρήστη. Στην ουσία δηλαδή σπαταλώνται ανώφελα χρηματικοί πόροι και επιβαρύνεται το περιβάλλον, ενώ μπορεί να δοθεί μία απλή και οικονομική λύση στο πρόβλημα. Υπενθυμίζεται βέβαια πως όλα τα παραπάνω αφορούν δίκτυα στα οποία δε μεταβάλλονται τα βάρη συνδέσμων ανεξαρτήτως των κυκλοφοριακών συνθηκών.

Αναλύοντας τη διαδικασία δρομολόγησης σε στατικά δίκτυα, μέσω ένος αλγόριθμου υπολογίζεται η συντομότερη διαδρομή και ο χρήστης μπορεί να ενημερωθεί για τη διαδρομή που πρέπει να ακολουθήσει. Την ενημέρωση αυτή θα τη λάβει με οπτικό ή ακουστικό τρόπο μέσω ενός συστήματος πλοήγησης. Κατά τη διαδικασία αυτή απαιτείται ο αλγόριθμος υπολογισμού της συντομότερης διαδρομής να επιλύσει το πρόβλημα όσο το δυνατόν ταχύτερα για δύο λόγους :
α) Σε δίκτυα με μεγάλο αριθμό κεντροειδών και συνδέσμων ο χρόνος εύρεσης της βέλτιστης διαδρομής είναι της τάξης των δευτερολέπτων και απαιτείται η διαδικασία να ολοκληρώνεται σύντομα ώστε να μην καθυστερεί η ενημέρωση του χρήστη.
β) Οι επεξεργαστές που διαθέτουν τα συστήματα πλοήγησης έχουν περιορισμένες δυνατότητες, που δε ξεπερνούν μέχρι στιγμής τα 700MHz .

Στη συνέχεια προτείνονται δύο νέοι αλγόριθμοι για την επίλυση του προβλήματος εύρεσης των συντομότερων διαδρομών από το κέντρο μίας ζώνης προς όλα τα υπόλοιπα κέντρα ζωνών του δικτύου. Έγινε προσπάθεια ώστε οι αλγόριθμοι αυτοί :
α) Να είναι απλοί σε μορφή ώστε να μπορούν να εκτελεστούν επιλύοντας προβλήματα σε οποιοδήποτε δίκτυο.
β) Να απαιτούν όσο το δυνατό λιγότερες βασικές πράξεις μέχρι τον τερματισμό τους ώστε να έχουν μικρό χρόνο εκτέλεσης.
γ) Να χρησιμοποιούν κατάλληλες δομές δεδομένων ώστε να έχουν βελτιωμένο χρόνο εκτέλεσης.

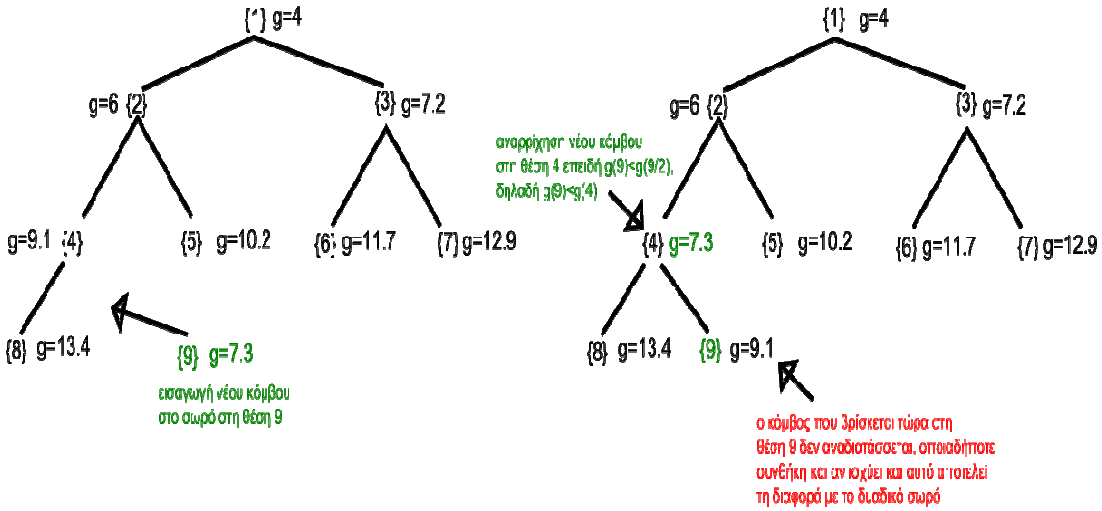
$1^{ος}$ αλγόριθμος : Ο αλγόριθμος έχει στοιχεία και από την κατηγορία των label setting και από την κατηγορία των label correcting αλγορίθμων καθώς δεν επιλέγει σε κάθε επανάληψη τον κόμβο που βρίσκεται πιο κοντά στον αρχικό ώστε να τον θέσει κόμβο ‘οδηγό’, διότι κάτι τέτοιο απαιτεί αρκετό χρόνο ώστε να προσδιοριστεί. Για το λόγο αυτό δεν ανήκει στην κατηγορία των label setting αλγορίθμων. Από την άλλη

όμως δεν επιλέγει τελείως τυχαία τον επόμενο κόμβο οδηγό, κάτι που εφαρμόζεται στους αλγορίθμους της κατηγορίας label correcting αλλά χρησιμοποιώντας μία νέα δομή δεδομένων που θα αναπτυχθεί στη συνέχεια και βασίζεται στο δυαδικό σωρό ταξινομεί τους κόμβους σε σειρά προτεραιότητας με βάση την απόστασή τους από τον κόμβο προέλευσης. Η νέα δομή δεδομένων μπορεί να καλείται και υβριδικός δυαδικός σωρός διότι δεν ικανοποιεί τη δομή του δυαδικού σωρού αλλά έχει κάποια στοιχεία της. Στη συνέχεια γίνεται πιο αναλυτική παρουσίαση αυτής της μεθόδου:

ΜΟΡΦΗ 1^{ου} ΑΛΓΟΡΙΘΜΟΥ

Σε κάθε βήμα επανάληψης υπάρχει ένας νέος κόμβος 'οδηγός' $\boldsymbol{\pi}$.
Κόμβος 'οδηγός' ονομάζεται ο κόμβος που βρίσκεται στην αρχή της λίστας R.
Βήμα 1(Αρχικοποιήσεις)
Κατά την αρχικοποίηση η λίστα R περιέχει μόνο τον κόμβο προέλευσης της μετακίνησης, ο οποίος αναγκαστικά είναι ο κόμβος 'οδηγός' κατά την πρώτη επανάληψη του αλγορίθμου.
Θεώρησε ότι όλοι οι κόμβοι απέχουν $d(w) \leftarrow +\infty$ από τον κόμβο προέλευσης και ότι $d(\boldsymbol{\pi}) \leftarrow 0$.
Βήμα 2(Κύριο Μέρος)
Για κάθε σύνδεσμο $(\boldsymbol{\pi},w)$:
Εάν $D(w) > D(\boldsymbol{\pi}) + w(\boldsymbol{\pi},w)$, τότε
$D(w) \leftarrow D(\boldsymbol{\pi}) + w(\boldsymbol{\pi},w)$, όπου $w(\boldsymbol{\pi},w)$ ο χρόνος διάνυσης του συνδέσμου $(\boldsymbol{\pi},w)$
Τέλος Εάν
Τέλος Για Κάθε
Εισήγαγε τους νέους κόμβους w που πληρούν την παραπάνω ιδιότητα στη λίστα R, η οποία αποτελεί έναν υβριδικό δυαδικό σωρό δεδομένων. Όσο λιγότερο απέχει κάθε κόμβος w από τον κόμβο προέλευσης τόσο υψηλότερα θα ανέβει στο σωρό, χωρίς ωστόσο να υπακούει ακριβώς στη δομή του δυαδικού σωρού ώστε να μην απαιτείται χρόνος για πλήρη αναδιάταξη.
Αφαίρεσε τον κόμβο 'οδηγό' $\boldsymbol{\pi}$ από το σωρό R. Ο νέος κόμβος που θα ανέβει στην αρχή του σωρού είναι ο κόμβος w ο οποίος έχει πιθανότατα τη μικρότερη τιμή D(w).
Αν δεν υπάρχουν άλλοι κόμβοι στο σωρό R ή αν ο κόμβος προορισμού έχει ανέβει στην πρώτη θέση του σωρού τερμάτισε.
Αλλιώς επανέλαβε τη διαδικασία από την αρχή.

Ο υβριδικός δυαδικός σωρός [R] είναι μία δομή δεδομένων στην οποία εισέρχονται κόμβοι βάσει της απόστασής τους από τον αρχικό κόμβο προορισμού. Ο χρόνος για την αφαίρεση ενός κόμβου είναι *O(1)* καθώς αφαιρείται το πρώτο στοιχείο του σωρού R(1). Κατά την εισαγωγή ενός επιπλέον κόμβου στο σωρό ακολουθείται η διαδικασία του σχήματος 6.5.



Σχήμα 6.3.Εισαγωγή κόμβου στον υβριδικό σωρό

Λόγω της δομής του υβριδικού σωρού ο επόμενος κόμβος που επιλέγεται ως κόμβος 'οδηγός' δεν είναι κατ'ανάγκη ο επόμενος κόμβος με τη μικρότερη απόσταση από τον αρχικό r. Έτσι ο ίδιος κόμβος μπορεί να εισέλθει ξανά στο σωρό ακόμα και αν έχει εισέλθει στο παρελθόν σε αυτόν. Το πλεονέκτημα αυτής της μεθόδου είναι ότι αναδιατάσσεται μόνο ο κόμβος που εισάγεται και όχι όλοι οι κόμβοι, κάτι που απαιτεί πολύ λιγότερο χρόνο.

2^{ος} αλγόριθμος – Dijkstra με Τριχοτομημένη Λίστα : Ο αλγόριθμος αυτός είναι πιο αποδοτικός και ανήκει στην κατηγορία των άπληστων αλγορίθμων (label setting). Στην ουσία πρόκειται για μία νέα δομή δεδομένων που μπορεί να χρησιμοποιηθεί στον αλγόριθμο Dijkstra. Η νέα προτεινόμενη μορφή έχει στόχο να βελτιώσει το χρόνο εκτέλεσης του αλγορίθμου Dijkstra με χρήση δυαδικού σωρού, χρησιμοποιώντας μία αρκετά πιο απλή δομή δεδομένων από το δυαδικό σωρό που μπορεί να υλοποιηθεί πιο ευκολότερα. Στο γενικό βήμα επανάληψης του αλγορίθμου Dijkstra με χρήση λίστας αναμονής συγκρίνονται οι αποστάσεις όλων των κόμβων που δεν έχουν επιλεγεί ακόμα με τον αρχικό κόμβο και επιλέγεται αυτός με τη μικρότερη απόσταση. Η νέα δομή δεδομένων μπορεί να περιορίσει δραστικά το πεδίο συγκρίσεων (πολύ λιγότερες βασικές πράξεις) ως εξής :

Όλοι οι κόμβοι αρχικά θεωρείται ότι απέχουν από τον κόμβο προέλευσης r άπειρη απόσταση $D(i) \leftarrow +\infty$, $\forall\, i \in \{1,N\}-r$. Στο γενικό βήμα επανάληψης η απόσταση αυτή αναπροσαρμόζεται. Εάν λοιπόν ο κόμβος προέλευσης συνδέεται άμεσα για παράδειγμα με επτά κόμβους (a,b,c,d,e,f,g), τότε στο πρώτο βήμα επανάληψης η απόσταση των κόμβων αυτών από τον αρχικό αναπροσαρμόζεται. Για την επιλογή του νέου πλησιέστερου κόμβου στον αρχικό σύμφωνα με τη θεωρία των άπληστων αλγορίθμων συγκρίνονται οι τιμές των επτά κόμβων και επιλέγεται, για παράδειγμα ο κόμβος c, για τον οποίο ισχύει $D(c) \leftarrow \text{Minimum}[D(i)]$, $\forall\, i \in \{a,b,c,d,e,f,g\}$. Για το σκοπό αυτό απαιτούνται επτά συγκρίσεις και εάν ο κόμβος προέλευσης συνδεόταν με επτά χιλιάδες κόμβους θα απαιτούνταν επτά χιλιάδες συγκρίσεις. Ακόμα και αν χρησιμοποιηθούν αποδοτικές μέθοδοι αναπαράστασης των δικτύων όπως η τρίτη

μέθοδος που αναπτύσσεται στην ενότητα 5.8. ο αριθμός των συγκρίσεων παραμένει ο ίδιος καθώς ελέγχονται και οι αποστάσεις των κόμβων που έχουν προκύψει από προηγούμενα βήματα.

Η μέθοδος που προτείνεται είναι η χρησιμοποίηση δύο σταθερών αριθμών που μπορούν να ονομαστούν ως CoefficientA και CoefficientB με την εξής χρήση : Κάθε κόμβος i που συνδέεται με τον αρχικό κόμβο εάν έχει κόστος μετακίνησης από τον αρχικό : D (i) < CoefficientA κατατάσσεται σε ένα σύνολο Q, εάν έχει : CoefficientA < D(i) < CoefficientB κατατάσσεται σε ένα δεύτερο σύνολο R και εάν έχει : D(i) > CoefficientB κατατάσσεται σε ένα τρίτο σύνολο S. Έτσι αν από τους επτά κόμβους οι δύο ανήκουν στο πρώτο σύνολο οι τρεις στο δεύτερο και οι υπόλοιποι στο τρίτο για την επιλογή του επόμενου πλησιέστερου κόμβου προς τον αρχικό θα συγκριθούν μόνο οι κόμβοι του πρώτου συνόλου, που έχουν και μικρότερο κόστος σε σχέση με τους υπόλοιπους γιατί κάποιος από αυτούς θα είναι ο επόμενος κόμβος με το ελάχιστο κόστος από τον αρχικό. Για να γίνει αυτό βέβαια απαιτούνται μόνο δύο συγκρίσεις. Καθώς η διαδικασία επαναλαμβάνεται κάποια στιγμή δε θα υπάρχουν άλλοι κόμβοι στο πρώτο σύνολο και τότε θα συγκρίνονται μεταξύ τους οι κόμβοι του δεύτερου συνόλου και στο τέλος του τρίτου συνόλου.

Η ταχύτητα της μεθόδου εξαρτάται από τις σταθερές CoefficientA και CoefficientB που θα επιλεγούν ώστε να ισομοιράζονται οι κόμβοι στα τρία σύνολα και να γίνονται όσο το δυνατό λιγότερες συγκρίσεις. Μία λύση που προτείνεται είναι κάθε φορά που προκύπτει ένας νέος κόμβος i με $D(i) < +\infty$ να προστίθεται η τιμή του σε μία σταθερά $T \leftarrow T + D(i)$ και να χρησιμοποιείται άλλη μία σταθερά που να προσμετρά των κόμβο ως μονάδα $d \leftarrow d+1$. Έτσι ο λόγος T/d μπορεί να θεωρηθεί ότι προσεγγίζει τον τρέχον μέσο όρο των αποστάσεων όλων των κόμβων από τον αρχικό κόμβο προέλευσης r. Κάθε φορά που κάποιος κόμβος ***u*** θα αποχωρεί από το σύνολο επειδή θα είναι πλέον ο τρέχον πλησιέστερος κόμβος στον αρχικό θα αναπροσαρμόζεται και η τιμή του τρέχον μέσου όρου ως $T \leftarrow T-D(\boldsymbol{u})$ και $d \leftarrow d-1$. Έτσι μπορούν να δοθούν τρέχουσες τιμές στις σταθερές :
$CoefficientA \leftarrow T/d \times 0.6666$ και $CoefficientB \leftarrow T/d \times 1.3333$.

DIJKSTRA ΜΕ ΤΡΙΧΟΤΟΜΗΜΕΝΗ ΛΙΣΤΑ ΩΣ ΔΟΜΗ ΔΕΔΟΜΕΝΩΝ

Βήμα 1(Αρχικοποιήσεις)
Θεώρησε ότι όλοι οι κόμβοι απέχουν $D(i) \leftarrow +\infty$ από τον κόμβο προέλευσης και ότι $D(\boldsymbol{u}) \leftarrow 0$, όπου $u \leftarrow r$. Επίσης θεώρησε τρείς σταθερές $A,B,C \leftarrow 0$.
Βήμα 2(Κύριο Μέρος)
$ΜΕΤΡΗΤΗΣ \leftarrow 0$
Όσο $ΜΕΤΡΗΤΗΣ \neq N$, όπου N ο αριθμός των κόμβων του δικτύου
Για Κάθε e = 1,E
    Εάν S(e) = u & D(E(e)) > D(S(e)) + w(e)
    D(E(e)) ← D(S(e)) + w(e)

```
        Εάν D( E(e) ) < CoefficientA
        A ← A+1
        Q(A) ← D( E(e) )
        QQ(A) ← D( E(e) )
        Αλλιώς Εάν D( E(e) ) < CoefficientB
        B ← B+1
        R(B) ← D( E(e) )
        RR(B) ← E(e)
        Αλλιώς
        C ← C+1
        S(C) ← d(w)
        SS(C) ← w
        Τέλος Εάν
    Τέλος Εάν
Τέλος Για Κάθε
Για Κάθε i = 1,A
Εάν QQ(k) ο κόμβος για τον οποίο Q( k ) = Minimum[Q( i )], ∀ iϵ{1,A}
u ← QQ(k)
Q(k) ← +∞
Πήγαινε στη Συνέχεια3
Αλλιώς πήγαινε στη Συνέχεια1
Τέλος Εάν
Τέλος για κάθε
Συνέχεια1
Για Κάθε i = 1,B
Εάν RR(k) ο κόμβος για τον οποίο R( k ) = Minimum[R( i )], ∀ iϵ{1,B}
u ← RR(k)
R(k) ← +∞
Πήγαινε στη Συνέχεια3
Αλλιώς πήγαινε στη Συνέχεια2
Τέλος Εάν
Τέλος Για Κάθε
Συνέχεια2
Για Κάθε i = 1,C
Εάν SS(k) ο κόμβος για τον οποίο S( k ) = Minimum[S( i )], ∀ iϵ{1,C}
u ← SS(k)
S(k) ← +∞
Τέλος Εάν
Τέλος για κάθε
Συνέχεια3
ΜΕΤΡΗΤΗΣ ← ΜΕΤΡΗΤΗΣ+1
Τέλος Όσο
```

6.2.ΡΟΕΣ ΧΡΗΣΤΗ ΣΕ ΣΤΑΤΙΚΑ ΔΙΚΤΥΑ

Στην παραπάνω ενότητα και στις ενότητες 5.2., 5.4. παρουσιάστηκαν ορισμένοι αλγόριθμοι για την επιλογή της συντομότερης διαδρομής από ένα χρήστη κατά τη μετακίνησή του μεταξύ ενός ζεύγους σημείων προέλευσης προορισμού r, s. Εάν το συγκοινωνιακό δίκτυο έχει περιορισμένη έκταση μπορεί η ζήτηση για μετακινήσεις που προκύπτει από ένα μητρώο προέλευσης-προορισμού να καταμεριστεί σε αυτό, όπως και στο Σχήμα 6.1. Αν το δίκτυο όμως είναι μεγάλο ο καταμερισμός της ζήτησης μπορεί να υλοποιηθεί μόνο μέσω της χρήσης ειδικού αλγορίθμου. Στη συνέχεια παρουσιάζεται ένα μαθηματικό μοντέλο για τον καταμερισμό της ζήτησης σε στατικά δίκτυα :

q_a : ο κυκλοφοριακός φόρτος του συνδέσμου α

t_a : ο χρόνος διάνυσης του συνδέσμου α, ο οποίος δεν εξαρτάται από τον κυκλοφοριακό φόρτο q_a στα στατικά δίκτυα αλλά παραμένει αμετάβλητος.

$\sum_k f_k^{rs} = q^{rs}$, δηλαδή η συνολική φόρτιση μεταξύ δύο κόμβων προέλευσης προορισμού r→s που συμβολίζεται ως q^{rs} πρέπει να καταμερίζεται σε όλες τις εναλλακτικές διαδρομές k που έχουν κόμβο προέλευσης τον r και κόμβο προορισμού τον s : f_k^{rs}

$f_k^{rs} \geq 0$, δηλαδή κάθε διαδρομή k μεταξύ δύο κόμβων r, s μπορεί να παραλαμβάνει κυκλοφοριακό φόρτο ή να μη φορτίζεται.

$q_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_k^{rs}$, ∀α. Αυτός ο περιορισμός υπονοεί ότι η φόρτιση κάθε συνδέσμου α ισούται με το άθροισμα της φόρτισης που παραλαμβάνουν όλες οι διαδρομές k με κόμβο προέλευσης τον τυχαίο κόμβο r και κόμβο προορισμού τον τυχαίο κόμβο s για τις οποίες ισχύει ότι ο σύνδεσμος α είναι μέλος της διαδρομής, κάτι που συνεπάγεται ότι $\delta_k^{rs} = 1$, αλλιώς $\delta_k^{rs} = 0$.

$P_z = \begin{cases} 1, & \text{εάν } t_z = minimum\,(t_i), i \epsilon k \\ 0 & \end{cases}$. Δηλαδή η πιθανότητα να επιλεγεί η διαδρομή z για την μετακίνηση μεταξύ των κεντροειδών r, s ισούται με τη μονάδα εάν ο χρόνος διάνυσης της διαδρομής z := t_z είναι μικρότερος από οποιονδήποτε χρόνο διάνυσης κάθε άλλης εναλλακτικής διαδρομής k που υλοποιεί αυτή τη μετακίνηση.

$f_k^{rs} = P_k \times q^{rs}$, δηλαδή η φόρτιση κάθε διαδρομής k μεταξύ δύο κόμβων προέλευσης προορισμού r, s είναι ίση με τη συνολική ζήτηση για τη μετακίνηση αυτή (q^{rs}) ή ίση με μηδέν. Από αυτή τη σχέση προκύπτει η ονομασία για τον καταμερισμού της ζήτησης σε στατικά δίκτυα ως μέθοδος του όλα ή τίποτα.

Με βάση το παραπάνω μαθηματικό μοντέλο προτείνεται ο αλγόριθμος :

ΑΛΓΟΡΙΘΜΟΣ : User's Equilibrium – Ροές Χρήστη σε Στατικά Δίκτυα

Μεταφορικό δίκτυο G={N,E}
Διάβασε από ένα αρχείο τη ζήτηση για μετακινήσεις στο δικτύο μεταξύ κάθε ζεύγους κόμβων προέλευσης-προορισμού i, j : Demand(i, j).
Διάβασε από ένα αρχείο τους χρόνους διάνυσης των συνδέσμων, t(e), $\forall$ e$\in${1,E}.
Για Κάθε i = 1,N
Χρησιμοποίησε ως υπορουτίνα έναν αλγόριθμο εύρεσης της συντομότερης διαδρομής από έναν κόμβο i προς όλους τους υπόλοιπους κόμβους του δικτύου (SSSPP).
Ο χρόνος διάνυσης κάθε συντομότερης διαδρομής από τον κόμβο i προς οποιοδήποτε άλλο κόμβο j$\in${1,N} υπολογίστηκε ως D(j), όπου D(i) = 0.
Από τον αλγόριθμο της Ενότητας 5.3., από τον οποίο προκύπτουν τα δένδρα των συντομότερων διαδρομών, προσδιόρισε τους συνδέσμους a$\in${1,E} από τους οποίους αποτελείται κάθε συντομότερη διαδρομή από τον κόμβο i προς οποιοδήποτε άλλο κόμβο.

    Για Κάθε j = 1,N
    Φόρτισε όλους τους συνδέσμους που αποτελούν τη συντομότερη διαδρομή i, j με το συνολικό φόρτο της μετακίνησης = Demand(i,j), j$\in${1,N}
    Τέλος Για Κάθε

Τέλος Για Κάθε

ΚΕΦΑΛΑΙΟ 7^{ο}

ΚΑΤΑΜΕΡΙΣΜΟΣ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ ΜΕ ΧΡΗΣΗ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

7.1.ΕΙΣΑΓΩΓΗ

Μία πιο ενδιαφέρουσα κατηγορία δικτύων, για την οποία αποτελεί αναπάντητο ερώτημα το κατά πόσο μπορούν να χρησιμεύσουν τα συστήματα πλοήγησης είναι τα δυναμικά δίκτυα. Στα δίκτυα αυτά το κόστος διάνυσης των συνδέσμων μεταβάλλεται όταν μεταβληθούν και οι κυκλοφοριακές συνθήκες. Είναι προφανές ότι σε αυτές τις κατηγορίες δικτύων δε μπορεί να χρησιμοποιηθεί ο τύπος δρομολόγησης που εφαρμόστηκε στα στατικά δίκτυα. Για το λόγο αυτό πρέπει να προταθεί ένας διαφορετικός τύπος δρομολόγησης ο οποίος μπορεί να αντιμετωπίζει τα αντίστοιχα ζητήματα που θέτει η δυναμική φύση των δικτύων. Ωστόσο, μέχρι σήμερα, χρησιμοποιούνται κατά μεγάλη πλειονότητα απλές συσκευές οι οποίες οδηγούν σε στατική καθοδήγηση και λειτουργούν με πληροφορίες σχετικά με τις κυκλοφοριακές συνθήκες οι οποίες δεν αναβαθμίζονται συχνά. Η πλειονότητα των συστημάτων πλοήγησης που διατίθενται χρησιμοποιεί χρόνους διάνυσης των συνδέσμων που προκύπτουν μέσω συντελεστών βαρύτητας οι οποίοι έχουν αμετάβλητες τιμές. Στην ουσία δηλαδή τα δίκτυα αντιμετωπίζονται με μία στατική θεώρηση από τα και δε λαμβάνονται υπόψη οι συνεχείς μεταβολές στην κυκλοφοριακή ροή.

Ο παραπάνω τύπος δρομολόγησης είναι γνωστός ως αυτόνομη αρχιτεκτονική και όπως αναφέρθηκε ήδη έχει τον περιορισμό της λειτουργίας με χρόνους διάνυσης συνδέσμων που δεν ανανεώνονται, καθώς δεν υπάρχει διαθέσιμη τεχνολογία που να αναβαθμίζει τον ψηφιακό χάρτη δεδομένων με νέα στοιχεία. Έτσι, ενώ η μαζική χρήση των συστημάτων πλοήγησης μπορεί να οδηγήσει σε βέλτιστο καταμερισμό της ζήτησης σε στατικά δίκτυα δεν έχει το ίδιο αποτέλεσμα και σε δυναμικά δίκτυα. Στην πραγματικότητα μάλιστα οι διαδρομές που προτείνονται διαφέρουν αρκετά από τις βέλτιστες. Με βάση αυτή την αρχιτεκτονική η χρήση τους σε δυναμικά δίκτυα μπορεί να έχει αξία μόνο για όσους δε γνωρίζουν την περιοχή της μετακίνησής τους.

Μία πρώτη απάντηση λοιπόν σχετικά με το εάν τα συστήματα πλοήγησης μπορούν να οδηγήσουν σε βέλτιστο καταμερισμό της ζήτησης σε δυναμικά δίκτυα είναι αρνητική. Δε μπορεί όμως να εγκαταληφθεί η προσπάθεια βελτίωσης των λειτουργιών τους καθώς αποτελούν ένα μέσο χαμηλού κόστους μέσω του οποίο μπορεί να βελτιωθεί ο καταμερισμός της ζήτησης στα δίκτυα και να προκύψουν μεγάλα οικονομικά και περιβαλλοντικά οφέλη. Δεν είναι υπερβολή να αναφερθεί ότι λόγω προβληματικού καταμερισμού της ζήτησης στα συγκοινωνιακά δίκτυα δαπανώνται άσκοπα τεράστια χρηματικά ποσά και ενεργειακοί πόροι. Λόγω της κυκλοφοριακής συμφόρησης μόνο στις Η.Π.Α. κατά το έτος 2000-2001 δαπανήθηκαν

άσκοπα 67,5 δισεκατομμύρια δολάρια, 5,7 δισεκατομμύρια γαλόνια βενζίνης και προέκυψαν 3,6 δισεκατομμύρια ώρες καθυστερήσεων σύμφωνα με έρευνα του Texas Transport Institute (2002). Επίσης κατά το έτος 2005, μόνο στο Los Angeles δαπανήθηκαν άσκοπα 9,325 εκατομμύρια δολάρια σύμφωνα πάλι με στοιχεία του Texas Transport Institute. Είναι προφανές λοιπόν ότι ακόμα και μικρές βελτιώσεις του καταμερισμού της ζήτησης μέσω των συστημάτων πλοήγησης σε δυναμικά δίκτυα μπορούν να οδηγήσουν σε μεγάλα οικονομικά και περιβαλλοντικά οφέλη. Για να μπορέσει όμως να συμβεί αυτό πρέπει αρχικά να επανεξεταστεί και να βελτιωθεί ο τύπος δρομολόγησης των συστημάτων πλοήγησης. Σε γενικές γραμμές πρέπει να προταθεί ένας τύπος δρομολόγησης ώστε να λαμβάνονται υπόψη οι μεταβολές στα κόστη διάνυσης των συνδέσμων σε δυναμικά δίκτυα. Αυτός ο τύπος δρομολόγησης θα έχει τα ακόλουθα πλεονεκτήματα συγκριτικά με τον στατικό τύπο δρομολόγησης :

α) Μπορεί να εφαρμοστεί σε δίκτυα με κυκλοφοριακή συμφόρηση, τα οποία αποτελούν την πλειονότητα των συγκοινωνιακών δικτύων.

β) Ο χρήστης ενημερώνεται για τη διαδρομή που πρέπει να ακολουθήσει με βάση τις παρούσες κυκλοφοριακές συνθήκες.

γ) Μπορεί να μειώσει δραστικά το μέσο χρόνο μετακινήσεων.

δ) Μπορεί να αυξήσει υπό προϋποθέσεις τη μέση ταχύτητα κυκλοφορίας των μεταφορικών μέσων.

στ) Ο χρόνος επεξεργασίας των δεδομένων δε διαφέρει από αυτόν στα στατικά δίκτυα, καθώς υπολογίζονται διαδρομές με βάση τις παρούσες συνθήκες συμφόρησης που επικρατούν κατά τον υπολογισμό.

ε) Ενημερώνει το χρήστη για καθημερινά γεγονότα, νέα σημεία ενδιαφέροντος, καθυστερήσεις λόγω ατυχημάτων κ.α.

Αυτός ο τύπος δρομολόγησης όμως είναι δύσκολα εφαρμόσιμος καθώς προσκρούει σε τεχνικά ζητήματα τα οποία δε μπορούν να αντιμετωπιστούν εύκολα όπως :

α) Απαιτεί συνεχώς νέα δεδομένα για την εφαρμογή του. Κάτι τέτοιο όμως δεν είναι εύκολα εφικτό λόγω του αυξημένου κόστους. Ο τρόπος συλλογής δεδομένων περιλαμβάνει μία σειρά από συγκοινωνιακά συστήματα μέτρησης που θα αναλυθούν στη συνέχεια. Ωστόσο τα συστήματα αυτά χρησιμοποιούνται κυρίως από κέντρα διαχείρισης της κυκλοφορίας και δε χρησιμοποιούνται από τα συστήματα πλοήγησης.

β) Ακόμα και στην περίπτωση της δυναμικής πληροφόρησης η διαδρομή που προτείνεται στο χρήστη υπολογίζεται με βάση τις συνθήκες κυκλοφοριακής συμφόρησης που επικρατούν στο μεταφορικό δίκτυο κατά τον υπολογισμό της. Έτσι η διαδρομή αυτή μπορεί να μην παραμένει η συντομότερη στη συνέχεια της μετακίνησης και όσο μεγαλύτερη είναι η μετακίνηση η πιθανότητα να συμβεί κάτι

τέτοιο αυξάνεται. Αυτό είναι απόρροια της δυσκολίας πρόβλεψης τυχόν μεταβολών στις κυκλοφοριακές συνθήκες μετά τον υπολογισμό της προτεινόμενης διαδρομής.

γ) Δε μπορούν να αποτιμηθούν με βεβαιότητα τα αποτελέσματα αυτού του τύπου δρομολόγησης στην εξισορρόπηση των συγκοινωνιακών δικτύων, καθώς το συνολικό κόστος μετακίνησης μπορεί να μειωθεί αλλά σε ορισμένες περιπτώσεις μπορεί και να αυξηθεί.

δ) Εάν οι συνθήκες στο δίκτυο μεταβληθούν κατά τη διάρκεια της διαδρομής, κάτι που είναι σύνηθες, δεν ισχύει ο χρόνος άφιξης στον προορισμό που έχει προβλεφθεί κατά τον υπολογισμό της διαδρομής.

Στις ενότητες που ακολουθούν προτείνονται διάφορες μέθοδοι, συστήματα μέτρησης κυκλοφοριακών συνθηκών και στοχαστικοί αλγόριθμοι. Μέσα από μία σύνθεση των παραπάνω πρακτικών επιχειρείται η υλοποίηση νέων τύπων δρομολόγησης που ξεφεύγουν από τις πρακτικές της αυτόνομης δρομολόγησης με απώτερο στόχο τη βελτίωση στην απόδοση των δυναμικών δικτύων.

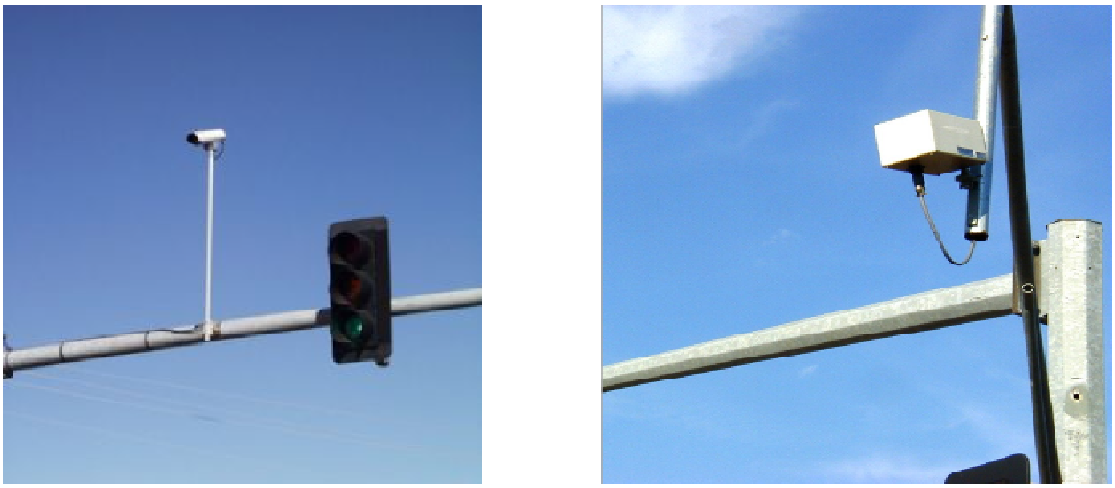
7.2.ΤΕΧΝΟΛΟΓΙΕΣ ΜΕΤΡΗΣΗΣ ΚΥΚΛΟΦΟΡΙΑΚΩΝ ΜΕΓΕΘΩΝ

Ένα από τα σημαντικότερα προβλήματα της προτεινόμενης δρομολόγησης είναι η ανάγκη για συνεχή συλλογή δεδομένων για το μεταφορικό δίκτυο, την αξιοποίησή τους και τον επανυπολογισμό του χρόνου διάνυσης κάθε συνδέσμου λόγω μεταβολής των κυκλοφοριακών συνθηκών. Για το λόγο αυτό παρουσιάζονται διάφορα μέσα συλλογής κυκλοφορικών δεδομένων που χρησιμεύουν και στον υπολογισμό της κυκλοφοριακής ροής.

α) Αισθητήρες επί του οδοστρώματος ή εντός του οδοστρώματος (in-roadway sensors) : Οι αισθητήρες αυτοί μπορεί να είναι ενσωματωμένοι ή στερεωμένοι στο οδόστρωμα. Το πιο απλό δείγμα αυτής της κατηγορίας είναι οι μετρητές με πεπιεσμένο αέρα (pneumatic road tube). Οι μετρητές αυτοί αποτελούνται από έναν ελαστικό σωλήνα που τοποθετείται στο οδόστρωμα είτε μόνιμα είτε προσωρινά. Το ένα άκρο του σωλήνα είναι κλειστό και το άλλο συνδέεται με ειδικό τύμπανο μετρητή που λειτουργεί με πεπιεσμένο αέρα. Όταν ένα μέσο μεταφοράς διέλθει από το σωλήνα, ωθεί την ποσότητα αέρα μέσα στο σωλήνα, ενεργοποιεί το μετρητή και καταγράφεται μία μονάδα. Μία άλλη τεχνική είναι η χρήση αγώγιμων καλωδίων ανίχνευσης. Οι αγωγοί ανίχνευσης αποτελούνται από ένα ή περισσότερα καλώδια που ενσωματώνονται στο οδόστρωμα, συνδέονται με ένα σύστημα ελέγχου και διαπερνώνται από ένα σήμα συχνότητας από 10 έως 200 kHz. Όταν ένα μεταφορικό μέσο διέλθει πάνω από το καλώδιο, η αγωγιμότητά του μειώνεται και έτσι εντοπίζεται η διέλευση του οχήματος. Η τεχνική των αγωγών ανίχνευσης χρησιμοποιήθηκε ευρέως για τη συλλογή δεδομένων, αφού επιτρέπει τη μέτρηση της κυκλοφοριακής ροής αλλά και την εκτίμηση της ταχύτητας των οχημάτων μέσω της επεξεργασίας διάφορων παραμέτρων. Επίσης έχει αποδεκτή απόδοση σε όλες τις κυκλοφοριακές και καιρικές συνθήκες. Τα μειονεκτήματά της είναι ότι πρέπει να γίνει πολύ προσεκτική εγκατάσταση αλλιώς τα αποτελέσματα των μετρήσεων μπορεί να μην είναι αποδεκτά και ότι απαιτείται η εγκατάσταση πολλών αγωγών, κάτι που αυξάνει σημαντικά το κόστος. Για το λόγο αυτό ο αυτόματος μετρητής με πεπιεσμένο αέρα παραμένει η πιο διαδεδομένη μορφή μέτρησης μέχρι σήμερα. Μια άλλη συσκευή που χρησιμοποιείται για τη συλλογή δεδομένων είναι οι μετρητές μαγνητικού πεδίου που ενεργοποιούνται και καταγράφουν τη διέλευση ενός μεταφορικού μέσου ανιχνεύοντας μεταβολές του μαγνητικού πεδίου, καθώς οι μεταλλικός σκελετός του μέσου διέρχεται πάνω από το σημείο τοποθέτησής τους.

Εικόνα 7.1.Μετρητής με πεπιεσμένο αέρα

β) Αισθητήρες έξω (παραπλεύρως ή πάνω) από το οδόστρωμα (over-roadway sensors or non-intrusive sensors) : Παραδείγματα αυτών των αισθητήρων είναι τα μικροκυματικά ραντάρ (microwave radar), οι ανιχνευτές υπερύθρων, υπερήχων, τα φωτοκύτταρα κ.α. Αυτοί οι ανιχνευτές τοποθετούνται δίπλα στο δρόμο και μπορούν να εγκατασταθούν εύκολα χωρίς να χρειάζεται να γίνουν εργασίες στο οδόστρωμα και να διακοπεί η κυκλοφορία. Σε αυτή την κατηγορία ανήκουν και οι ανιχνευτές επεξεργασίας εικόνας από κάμερα (video detectors). Τα συστήματα επεξεργασίας εικόνας χρησιμοποιούν την τεχνολογία ανάλυσης εικόνας για την αυτόματη ανίχνευση των δεδομένων για την κυκλοφορία από βίντεο που συλλέγονται από κλειστό κύκλωμα τηλεόρασης. Παρόλο που η τεχνολογία υπάρχει εδώ και κάποια χρόνια, μόλις πρόσφατα η μείωση στο κόστος αυτής της μεθόδου επέτρεψε τη διάδοσή της. Υπάρχουν συστήματα επεξεργασίας εικόνας που απαιτούν την ανθρώπινη παρακολούθηση (αναλογικά) για τη μέτρηση των χαρακτηριστικών της κυκλοφορίας όπως είναι τα κλειστά αναλογικά κυκλώματα τηλεόρασης σε κέντρα διαχείρισης της κυκλοφορίας. Μια νέα διάσταση σε αυτό τον τύπο μέτρησης όμως δίνεται μέσω των ψηφιακών ανιχνευτών όπου η εικόνα που μεταδίδεται μπορεί να υποστεί επεξεργασία μέσω ειδικών αλγορίθμων και να προκύψουν τα στοιχεία διάφορων κυκλοφοριακών μεγεθών. Τα κυκλοφοριακά μεγέθη που μπορούν να μετρηθούν είναι ο όγκος της κυκλοφορίας, η ταχύτητα και η ταξινόμηση των οχημάτων. Η αξιοπιστία των μετρήσεων εξαρτάται από τις συνθήκες φωτισμού, τις καιρικές συνθήκες αλλά και τη γωνία της κάμερας και τη θέση της.



Εικόνα 7.2.Ανιχνευτές τύπου κάμερας και μικροκυματικού ραντάρ

γ) Συστήματα μέτρησης που βασίζονται στα μεταφορικά μέσα : Οι πιο ραγδαία αναπτυσσόμενες τεχνολογίες καταγραφής μεγεθών κυκλοφοριακής ροής είναι αυτές που χρησιμοποιούν το μεταφορικό μέσο ως κινούμενο αισθητήρα μέσω ειδικού εξοπλισμού που εγκαθίσταται σε αυτό. Έτσι το μεταφορικό μέσο που λειτουργεί ως κινούμενος αισθητήρας μπορεί να μεταδίδει σε έναν κεντρικό υπολογιστή ή να αποθηκεύει πληροφορίες για την ταχύτητα και το χρόνο διαδρομής ή και τη θέση του ανά τακτά χρονικά διαστήματα. Το πλεονέκτημα των τεχνολογιών αυτών είναι ότι δίνουν τη δυνατότητα παρακολούθησης της κυκλοφορίας (ταχύτητα σημείου ή διαδρομής, χρόνος διαδρομής από και προς επιλεγμένα σημεία, δεδομένα προέλευσης-προορισμού κ.α.) σε ένα ευρύτερο μεταφορικό δίκτυο. Οι τεχνολογίες που χρησιμοποιούνται σήμερα είναι η αναγνώριση οχημάτων με τη βοήθεια καρτών (κάρτες RFID) που λειτουργούν ως αναμεταδότες (transponders, tags) κάθε φορά που το εξοπλισμένο όχημα διέρχεται από συγκεκριμένα σημεία ελέγχου (Automatic Vehicle Identification-AVI) στα οποία έχει εγκατασταθεί ειδικός εξοπλισμός (RFID readers). Αυτή η μέθοδος είναι γνωστή και ως τεχνική παθητικής αναγνώρισης με πολύ χαμηλότερο κόστος από την ενεργητική αλλά και πολύ περιορισμένα δεδομένα. Η τεχνική ενεργητικής αναγνώρισης χρησιμοποιεί ένα σύστημα εντοπισμού θέσης (GPS) με στόχο την αυτόματη αναγνώριση της θέσης του οχήματος (Automatic Vehicle Locator-AVL). Η ενεργητική τεχνική πλεονεκτεί στο ότι η παρακολούθηση μπορεί να γίνει ανεξαρτήτως σταθερών σημείων σε όλο το δίκτυο, αλλά απαιτείται μεγάλος αριθμός εξοπλισμένων οχημάτων για να επιτευχθεί κάτι τέτοιο. Περισσότερες πληροφορίες παρουσιάζονται στο βιβλίο Φραντζεσκάκης και συν. (1997).

7.3.ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ ΧΡΟΝΟΥ ΔΙΑΝΥΣΗΣ ΣΥΝΔΕΣΜΩΝ ΣΕ ΔΙΚΤΥΑ ΜΕ ΚΥΚΛΟΦΟΡΙΑΚΗ ΣΥΜΦΟΡΗΣΗ

Όπως έχει αναφερθεί ήδη τα συστήματα πλοήγησης χρησιμοποιούν συντελεστές βαρύτητας για τον υπολογισμό του χρόνου διάνυσης των συνδέσμων ενός οδικού δικτύου. Μέχρι σήμερα δεν έχει γίνει προσπάθεια απευθείας καταγραφής των χρόνων διάνυσης των συνδέσμων ενός δικτύου. Κάτι τέτοιο δε θα είχε μεγάλη αξία άλλωστε καθώς οι χρόνοι αυτοί μεταβάλλονται συνεχώς κυρίως λόγω των μεταβολών στην κυκλοφοριακή ροή. Για τον υπολογισμό του χρόνου διάνυσης ενός συνδέσμου (π.χ. μίας αστικής οδού) απαιτείται ο προσδιορισμός της μέσης ταχύτητας κίνησης σε αυτή την οδό. Η ταχύτητα των μέσων μεταφοράς στις αστικές οδούς εξαρτάται από τρεις κύριους παράγοντες :

α) Το περιβάλλον της οδού που περιλαμβάνει τα γεωμετρικά χαρακτηριστικά, το χαρακτήρα των παρόδιων δραστηριοτήτων και τις γειτονικές χρήσεις γης.

β) Την αλληλεπίδραση μεταξύ οχημάτων που ορίζεται από την πυκνότητα της κυκλοφορίας, την αναλογία φορτηγών και λεωφορείων και τις στρέφουσες κινήσεις.

γ) Τον έλεγχο της κυκλοφορίας μέσω φωτεινής σηματοδότησης ή σήμανσης που αναγκάζει ένα τμήμα των οχημάτων να επιβραδύνει ή να σταματήσει για να εξασφαλιστεί η προτεραιότητα των εμπλεκόμενων οχημάτων.

Πριν συνεχιστεί η ανάλυση δίνονται ορισμένοι ορισμοί που αφορούν τα σημαντικότερα κυκλοφοριακά μεγέθη :

Κυκλοφοριακός Φόρτος : Ο αριθμός των οχημάτων που διέρχονται από μία ιδεατή διατομή κατά το χρονικό διάστημα Τ. Ο κυκλοφοριακός φόρτος έχει μονάδα μέτρησης (οχήματα ανά χρονικό διάστημα).

Πυκνότητα : Ο αριθμός των οχημάτων που βρίσκονται σε τμήμα συνολικού μήκους L τη χρονική στιγμή t. Η πυκνότητα έχει μονάδα μέτρησης (οχήματα ανά μονάδα μήκους).

Ταχύτητα σημείου : Η ταχύτητα που έχει ένα όχημα όταν περνά από δεδομένο σημείο.

Ταχύτητα διαδρομής : Η μέση ταχύτητα με την οποία κινήθηκε ένα όχημα από το σημείο προέλευσης στο σημείο προορισμού, υπολογιζόμενων και των καθυστερήσεων λόγω στάσεων.

Μέση ταχύτητα χρόνου : Ο αριθμητικός μέσος όρος των ταχυτήτων σημείου των οχημάτων που περνούν μπροστά από μια διατομή οδού σε μια δεδομένη χρονική περίοδο.

Μέση ταχύτητα χώρου : Ο αριθμητικός μέσος όρος των ταχυτήτων σημείου, που έχουν, σε μια ορισμένη χρονική περίοδο όλα τα οχήματα που βρίσκονται σε ένα δεδομένο τμήμα της οδού. Αναφέρεται στο χρόνο που απαιτείται για να διανυθεί ένα οδικό τμήμα.

Από τους παραπάνω ορισμούς προκύπτει η θεμελιώδης σχέση της κυκλοφοριακής ροής : Κυκλοφοριακός Φόρτος (q) = Πυκνότητα (k) x Μέση Ταχύτητα Χώρου (u).
Αυτή η σχέση μπορεί να αποδειχθεί ως εξής :
Δεχόμενοι ένα μικρό οδικό τμήμα μήκους L, το οποίο διασχίζουν N(T) οχήματα (της ίδιας κατεύθυνσης) κατά τη διάρκεια ενός χρονικού διαστήματος T, η μέση πυκνότητα κατά το χρονικό διάστημα T είναι :

$$k = \frac{Μέσος\ Αριθμος\ Οχημάτων\ στο\ εξεταζόμενο\ οδικό\ τμήμα}{Μήκος\ οδικού\ τμήματος}$$

Αν το όχημα i διασχίζει το οδικό τμήμα μήκους L σε χρόνο t_i η πιθανότητα P_i το όχημα i να βρίσκεται μέσα στο εξεταζόμενο οδικό τμήμα σε κάποια χρονική στιγμή εντός του διαστήματος T δίνεται από τη σχέση : $P_i = \frac{t_i}{T}$
Άρα ο μέσος αριθμός οχημάτων στο εξεταζόμενο τμήμα μήκους L προκύπτει ίσος με : $\frac{\sum_i^n t_i}{T}$
Από τα παραπάνω η μέση πυκνότητα κατά το χρονικό διάστημα T ισούται με :

$$k = \frac{\frac{\sum_{i=1}^{N(T)} t_i}{T}}{L} = \frac{\frac{N(T)}{T}}{\frac{N(T)L}{\sum_{i=1}^{N(T)} t_i}}$$

Επειδή το μήκος του οδικού τμήματος L είναι μικρό, είναι δυνατό να θεωρηθεί ότι ο αριθμός των οχημάτων N(T) που διασχίζουν το οδικό τμήμα στη χρονική περίοδο T είναι ίδιος με τον αριθμό των οχημάτων που περνούν από μία διατομή στη θέση x του οδικού τμήματος L. Επομένως η παραπάνω σχέση γίνεται : k = q/u

Με βάση αυτήν τη σχέση εάν είναι γνωστά τα δύο μεγέθη προκύπτει το τρίτο. Έχουν γίνει πολλές ερευνητικές προσπάθειες ώστε αυτά τα τρία μεγέθη να συνδεθούν μεταξύ τους. Από τις έρευνες αυτές έχουν προκύψει προσεγγιστικές σχέσεις που συνδέουν την πυκνότητα και τη μέση ταχύτητα χώρου.

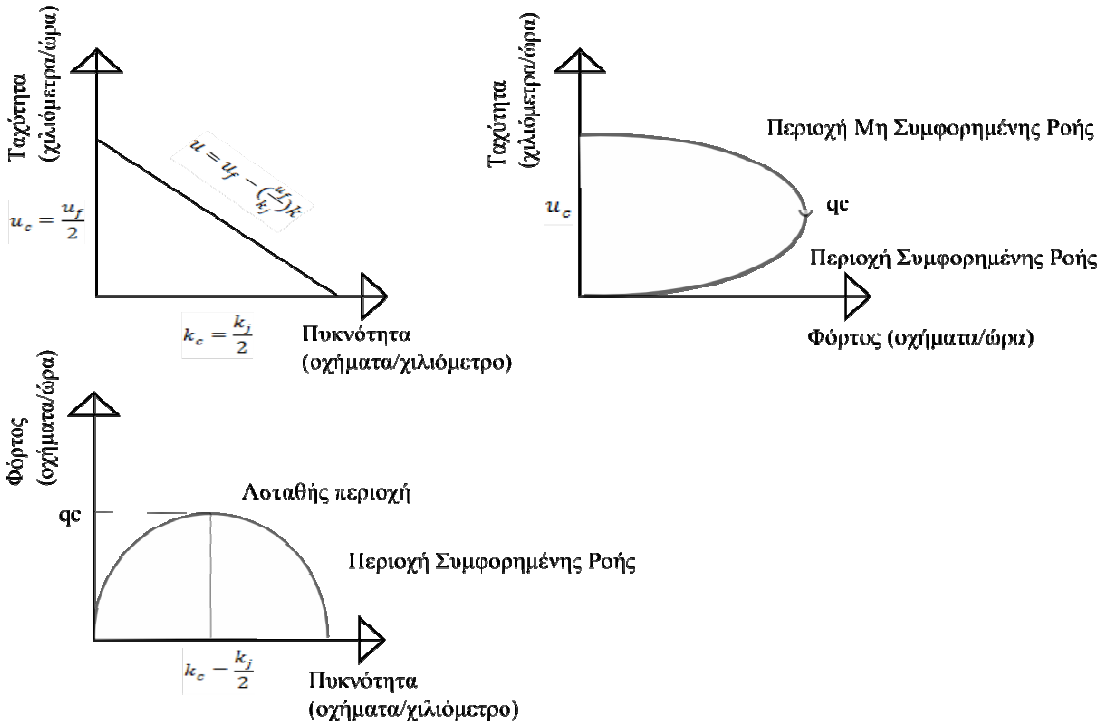
Μία από τις πρώτες έρευνες σε αυτό το θέμα ήταν του Greenshields (1935). Πιο συγκεκριμένα πρότεινε μία γραμμική σχέση που συνδέει τη μέση ταχύτητα χώρου με τη μέση πυκνότητα : u = a – b × k , όπου a, bϵR^+.

Όταν η πυκνότητα τείνει προς το μηδέν η ταχύτητα προσεγγίζει την ταχύτητα ελεύθερης ροής u→u_s. Όταν η πυκνότητα φθάσει στη μέγιστη τιμή της k→k_j, τότε αυτή η κατάσταση αντιστοιχεί σε ακινητοποίηση των οχημάτων u→0. Οπότε η ταχύτητα συνδέεται με τη μέση πυκνότητα μέσω της σχέσης :

$$u = u_f - (\frac{u_f}{k_j})k$$

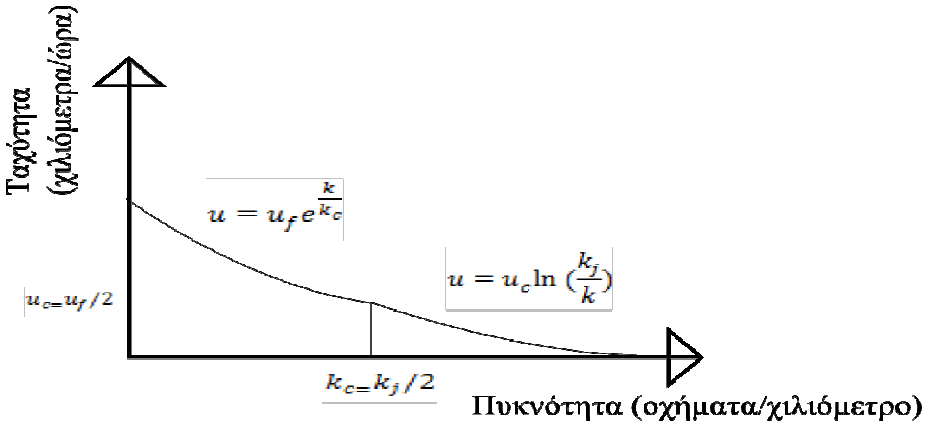
και ο φόρτος αντίστοιχα : $q = u_f k - (\frac{u_f}{k_j})k^2$

Από τις σχέσεις αυτές προκύπτουν τα ακόλουθα διαγράμματα :



Διάγραμμα 7.1.Θεμελιώδη διαγράμματα κυκλοφοριακής ροής κατά Greenshield

Από τη σχέση φόρτου-πυκνότητας φαίνεται ότι όταν η πυκνότητα τείνει στο μηδέν και ο φόρτος τείνει στο μηδέν. Ο φόρτος όμως τείνει στο μηδέν και στην περίπτωση που η πυκνότητα τείνει στην πυκνότητα κορεσμού, επειδή τα οχήματα παραμένουν ακινητοποιημένα επί της οδού. Τα παραπάνω ισχύουν με βάση τη θεώρηση ότι η ταχύτητα και η πυκνότητα συνδέονται με γραμμική σχέση. Ωστόσο σε μία συνολική προσπάθεια που έγινε για την αξιολόγηση της μορφής της σχέσης πυκνότητας-πιθανότητας από τους Drake et al. (1967) προέκυψε ότι η σχέση της ταχύτητας-πυκνότητας μπορεί να προσεγγιστεί με μία τμηματική εκθετική μορφή. Το πρώτο τμηματικά εκθετικό πρότυπο παρουσιάστηκε από τον Edie (1961).



Διάγραμμα 7.2.Σχέση ταχύτητας-πυκνότητας κατά Edie

Μέχρι σήμερα έχουν παρουσιαστεί και άλλα διαγράμματα που προσεγγίζουν ικανοποιητικά τη σχέση ταχύτητας-πυκνότητας, ταχύτητας-φόρτου, φόρτου-πυκνότητας. Ωστόσο έχει δειχθεί ότι με τις εμπειρικές σχέσεις δεν περιγράφεται επαρκώς όλο το εύρος των κυκλοφοριακών καταστάσεων. Αυτό οφείλεται στη μεταβλητότητα της κυκλοφοριακής ροής, κυρίως στην περιοχή κοντά στη μέγιστη τιμή του φόρτου. Έτσι δε μπορούν να ληφθούν με βεβαιότητα συμπεράσματα για το μέσο χρόνο διάνυσης ενός οδικού τμήματος, παρά μόνο προσεγγιστικά.

Μία άλλη μέθοδος που μπορεί να έχει πιο αποδοτικά αποτελέσματα στον υπολογισμό της μέσης ταχύτητας διάνυσης ενός οδικού τμήματος είναι η μέθοδος προσδιορισμού της στάθμης εξυπηρέτησης στην οποία λειτουργεί κάθε οδικό στοιχείο κάτω από ένα συγκεκριμένο φόρτο. Ο προσδιορισμός της στάθμης εξυπηρέτησης προϋποθέτει αρχικά τη γνώση της κυκλοφοριακής ικανότητας κάθε οδικού στοιχείου. Η κυκλοφοριακή ικανότητα εκφράζει το μέγιστο ωριαίο αριθμό ροής οχημάτων ή προσώπων, κατά μία διεύθυνση ή και κατά τις δύο διευθύνσεις, κατά τη διάρκεια μίας χρονικής περιόδου, υπό τις οδικές και κυκλοφοριακές συνθήκες καθώς και τις συνθήκες ελέγχου της κυκλοφορίας που επικρατούν. Η στάθμη εξυπηρέτησης εξαρτάται από περισσότερους παράγοντες, καθώς αποτελεί ποιοτικό μέγεθος που εκφράζει τις συνθήκες λειτουργίας σε ένα ρεύμα κυκλοφορίας όπως τις αντιλαμβάνονται οι χρήστες τους. Ο προσδιορισμός της στάθμης εξυπηρέτησης γίνεται μέσω των δεικτών εξυπηρέτησης (ταχύτητα, χρόνος μετακίνησης, πυκνότητα, καθυστερήσεις). Η κυκλοφοριακή ικανότητα και η στάθμη εξυπηρετησης ενός οδικού στοιχείου εξαρτάται συνήθως από :

α) Τις οδικές συνθήκες που περιλαμβάνουν τα γεωμετρικά χαρακτηριστικά του οδικού στοιχείου (αριθμός λωρίδων κυκλοφορίας, τύπος οδού, ταχύτητα μελέτης κ.α.)
β) Τις κυκλοφοριακές συνθήκες, καθώς όπως έχει αναλυθεί ήδη αύξηση της μέσης πυκνότητας οδηγεί σε μείωση της μέσης ταχύτητας χώρου.
γ) Τις συνθήκες ελέγχου και κυρίως τη σηματορύθμιση αλλά και τις άλλες μορφές σηματοδότησης.
δ) Την εφαρμογή των νέων τεχνολογιών που μπορούν να αυξήσουν τη χωρητικότητα του μεταφορικού δικτύου και να βελτιώσουν ορισμένα κυκλοφοριακά μεγέθη.

Το πρόβλημα αυτής της μεθόδου έγκειται στο μεγάλο υπολογιστικό φόρτο, καθώς κάθε οδικό στοιχείο πρέπει να εξετάζεται ξεχωριστά. Σύμφωνα με το τελευταίο εγχειρίδιο κυκλοφοριακής ικανότητας των Η.Π.Α. : Highway Capacity Manual (2000) καλύπτεται ο υπολογισμός δεκατριών διαφορετικών οδικών στοιχείων. Για τον υπολογισμό κάθε οδικού στοιχείου δίνεται αρχικά μία σειρά συμβατικών μεγεθών, γνωστών και ως βασικές συνθήκες, τις οποίες πληροί το πρότυπο οδικό στοιχείο που ανήκει στην ίδια κατηγορία με αυτό που εξετάζεται. Στη συνέχεια χρησιμοποιείται μία σειρά συντελεστών για την προσαρμογή στις πραγματικές συνθήκες του υπό εξέταση οδικού στοιχείου. Τα οδικά στοιχεία τα οποία μπορεί εξεταστούν σύμφωνα με τα στοιχεία του εγχειρίδιου είναι :

Αστικές Οδοί, Σηματοδοτούμενοι Κόμβοι, Μη Σηματοδοτούμενοι Κόμβοι, Πεζοί, Ποδήλατα, Αρτηρίες Δύο Λωρίδων, Αρτηρίες με Τέσσερις ή περισσότερες Λωρίδες, Ελεύθερες Λεωφόροι, Βασικά Τμήματα Ελεύθερων Λεωφόρων, Περιοχές Πλέξης Ελεύθερων Λεωφόρων, Ράμπες και συνδέσεις Ραμπών, Καταλήξεις Ραμπών Ανισόπεδου Κόμβου, Μαζικά Μέσα Μεταφοράς.

Όπως είναι φανερό ορισμένα από αυτά, όπως τα μέσα μαζικής μεταφοράς, δεν αποτελούν οδικά στοιχεία, όμως μπορεί να προσδιοριστεί η στάθμη εξυπηρέτησής τους μέσω κάποιων ειδικών δεικτών εξυπηρέτησης.

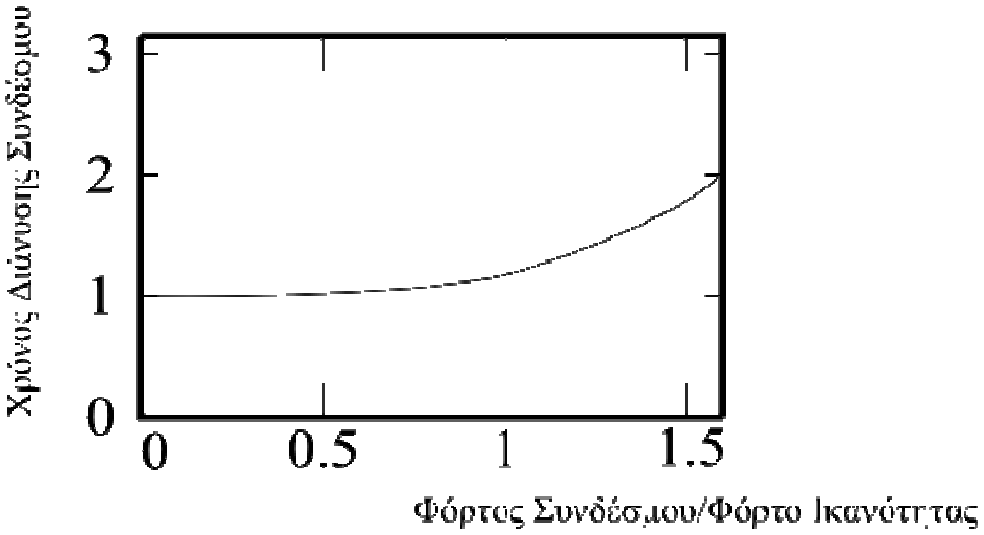
Για τη χρήση της μεθόδου για τον υπολογισμό της μέσης ταχύτητας διάνυσης των οδικών τμημάτων μιας περιοχής μελέτης πρέπει να ληφθούν υπόψη οι εξής παράμετροι :
α) Η μέθοδος βασίζεται σε στοιχεία από πίνακες που έχουν προκύψει από διάφορες έρευνες και τα αποτελέσματα που προκύπτουν (π.χ. μέση ταχύτητα διάνυσης οδικού τμήματος) μπορεί να διαφέρουν από την πραγματικότητα. Για το λόγο αυτό πρέπει να ερευνηθεί εάν ο βαθμός απόκλισης των αποτελεσμάτων είναι αποδεκτός.
β) Απαιτείται η καταγραφή μίας σειράς στοιχείων των οδικών τμημάτων της περιοχής. Στην κλίμακα της περιοχής μελέτης είναι υλοποιήσιμο αλλά σε μεγαλύτερη κλίμακα παρουσιάζονται έντονα προβλήματα καθώς απαιτείται η καταγραφή συγκεκριμένων στοιχείων όπως κλίση οδού, στοιχεία χάραξης, πλάτος ερεισμάτων, ύπαρξη εμποδίων, μορφή σηματοδότησης κ.α.
γ) Απαιτείται συλλογή δεδομένων για πλήθος δυναμικών στοιχείων. Αυτό είναι ένα από τα σημαντικότερα μειονεκτήματα της μεθόδου καθώς σε ορισμένες περιπτώσεις απαιτούνται πολύ αναλυτικά στοιχεία όπως ποσοστό στρεφόντων οχημάτων προς κάποια κατεύθυνση, ποσοστό βαρέων οχημάτων κ.α.
δ) Η επεξεργασία όλων αυτών των στοιχείων είναι ιδιαίτερα χρονοβόρα και δεν είναι εύκολο να υπάρχει δυναμική ροή αποτελεσμάτων. Στα δυναμικά δίκτυα όμως αυτό είναι απαραίτητο ώστε να ενημερώνεται ο χρήστης έγκαιρα για τις μεταβολές στο δίκτυο.

Μία πιο εύχρηστη πρακτική για τον υπολογισμό του χρόνου διάνυσης ενός μεταφορικού συνδέσμου αποτελεί μία συνάρτηση γνωστή με τα αρχικά BPR (*U.S. Bureau of Public Roads, Traffic Assignment Manual, Dept. Of Commerce, Washigton D.C., 1964*) η οποία μοντελοποιεί με αρκετή ακρίβεια τη σχέση ανάμεσα στο χρόνο διάνυσης ενός κυκλοφοριακού συνδέσμου και τις συνθήκες κυκλοφορίας σε αυτόν. Η μορφή της συνάρτησης είναι η ακόλουθη :

$t_a(q_a) = t_a^0 [\ 1+ \alpha\ (q_a\ /\ q_{a,cap})^{\beta}]$, όπου α, β σταθερές που μεταβάλλονται με το είδος και την ιεράρχηση του οδικού συνδέσμου.

Ο χρόνος t_a^0 είναι ο χρόνος ελεύθερης ροής στην οριακή συνθήκη όπου ο πραγματικός φόρτος του συνδέσμου προς τον φόρτο ικανότητάς του τείνει στο μηδέν. Επίσης με $q_{a,cap}$ συμβολίζεται ο φόρτος ικανότητας του συνδέσμου α.

Σύμφωνα με την παραπάνω συνάρτηση ο χρόνος διάσχυσης του οδικού τμήματος είναι ελάχιστος όσο ο λόγος του πραγματικού φόρτου προς το φόρτο ικανότητας είναι μικρός. Κατά την αύξηση του φόρτου ο χρόνος διάνυσης αυξάνεται με ιδιαίτερα αργό ρυθμό μέχρι το σημείο που ο πραγματικός φόρτος ισούται με το φόρτο ικανότητας του συνδέσμου. Από το σημείο αυτό ο χρόνος διάνυσης αυξάνεται εκθετικά. Τα παραπάνω παρουσιάζονται στο ακόλουθο σχήμα για α = 0,15 και β = 4.



Σχήμα 7.1.Σχέση χρόνου διάνυσης και λόγου φόρτου/ικανότητας

Η συνάρτηση BPR έχει χρησιμοποιηθεί περισσότερο από κάθε άλλη για τον υπολογισμό του χρόνου καθυστέρησης σε σχέση με τις κυκλοφοριακές συνθήκες. Ωστόσο κατά το παρελθόν χρησιμοποιήθηκε κυρίως για την κατανομή της ημερήσιας ζήτησης και ως εκ τούτου δεν ενδείκνυται για την επίλυση της φόρτισης των δικτύων ιδίως σε συνθήκες αιχμής. Για το λόγο αυτό μπορεί να μειωθεί η κυκλοφοριακή ικανότητα του συνδέσμου $q_{a,cap}$ ειδικά σε αστικά κέντρα πολλαπλασιαζόμενη με ένα μειωτικό συντελεστή, $q_{a,cap} = 0.7 * q_{a,cap}$. Η νέα αυτή διατύπωση καθιστά τη σχέση φόρτου/ικανότητας – χρόνου διάνυσης δυσμενέστερη αλλά και πιο ρεαλιστική.

7.4.ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΤΟΥ ΚΑΤΑΜΕΡΙΣΜΟΥ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ ΜΕΣΩ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

7.4.1.ΕΙΣΑΓΩΓΙΚΑ ΣΤΟΙΧΕΙΑ

Όπως έχει αναφερθεί ήδη η βελτιστοποίηση του καταμερισμού της ζήτησης σε δίκτυα όπου το βάρος των συνδέσμων μεταβάλλεται είναι ζωτικής σημασίας τόσο για την ανάπτυξη των πόλεων όσο και για το περιβάλλον. Για το λόγο αυτό στη συνέχεια προτείνονται νέοι τύποι δρομολόγησης ώστε να προτείνεται συνεχώς στο χρήστη μία διαδρομή η οποία προσεγγίζει τη βέλτιστη σε δυναμικά δίκτυα. Αρχικά εξετάζονται τρόποι δρομολόγησης θεωρώντας ότι ένα ποσοστό των χρηστών διαθέτει σύστηματα πλοήγησης, κάτι που αποτελεί και τη ρεαλιστική συνθήκη. Υπό αυτή τη συνθήκη εξετάζεται ποιές μέθοδοι μπορεί να εφαρμοστούν ώστε να προτείνονται στους χρήστες που διαθέτουν συστήματα πλοήγησης διαδρομές που προσεγγίζουν τις βέλτιστες. Στη συνέχεια εξετάζονται τρόποι δρομολόγησης θεωρώντας ότι όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης, κάτι που αποτελεί την ιδεατή συνθήκη. Υπό αυτή τη συνθήκη παρουσιάζονται μέθοδοι ώστε η δρομολόγηση μέσω των συστημάτων πλοήγησης να οδηγεί στο βέλτιστο καταμερισμό της ζήτησης στο δίκτυο.

7.4.2.ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ, ΟΤΑΝ ΜΙΚΡΟ ΠΟΣΟΣΤΟ ΤΩΝ ΧΡΗΣΤΩΝ ΔΙΑΘΕΤΕΙ ΣΥΣΤΗΜΑΤΑ ΠΛΟΗΓΗΣΗΣ

Επειδή τα συστήματα πλοήγησης σχεδόν στο σύνολό τους υπολογίζουν διαδρομές με βάση την ισορροπία του χρήστη, δηλαδή προτείνουν τη συντομότερη διαδρομή μεταξύ δύο σημείων προέλευσης-προορισμού σε κάθε χρήστη θα εξεταστεί αυτή η περίπτωση. Σε επόμενο κεφάλαιο ελέγχεται εάν αυτή η πρακτική είναι σωστή και τι μπορεί να γίνει ώστε να βελτιωθεί, προς το παρόν όμως θεωρείται ότι ο αντικειμενικός στόχος είναι η ισορροπία του χρήστη.

Σε συγκοινωνιακά δίκτυα όπου μόνο ένα μικρό ποσοστό των χρηστών διαθέτει συστήματα πλοήγησης μπορεί να θεωρηθεί ότι η επίδραση του ποσοστού αυτού στον καταμερισμό της ζήτησης είναι αμελητέα. Ο καταμερισμός της ζήτησης σε αυτά τα δίκτυα δεν είναι εύκολα προβλέψιμος διότι έχει προκύψει από τις επιμέρους επιλογές κάθε χρήστη υπό συνθήκες έλλειψης πληροφόρησης. Μία προσέγγιση είναι να θεωρηθεί ότι οι χρήστες, με βάση την προσωπική τους εμπειρία, ακολουθούν διαδρομές που προσεγγίζουν τις βέλτιστες, όπως αναφέρεται σχετικά και στη στοχαστική ισορροπία του χρήστη.

Στη συνέχεια παρουσιάζονται τέσσερις μέθοδοι για τον υπολογισμό της βέλτιστης διαδρομής σε δίκτυα όπου ένα ποσοστό των χρηστών διαθέτει συστήματα πλοήγησης. Οι μέθοδοι αυτοί έχουν ιδιαίτερη σημασία καθώς μπορούν να

χρησιμοποιηθούν από τα συστήματα πλοήγησης ώστε να προταθούν στους χρήστες διαδρομές οι οποίες προσεγγίζουν τις βέλτιστες σε δυναμικά δίκτυα υπό συνθήκες αβεβαιότητας.

7.4.2.1.ΒΕΛΤΙΩΣΕΙΣ ΤΗΣ ΑΥΤΟΝΟΜΗΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΓΙΑ ΤΗ ΧΡΗΣΗ ΤΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ

Όπως έχει αναφερθεί ήδη τα συστήματα πλοήγησης ακολουθούν στην πλειονότητά τους την αυτόνομη δρομολόγηση στην οποία οι χρόνοι διάνυσης των συνδέσμων θεωρούνται σταθεροί. Η προτεινόμενη μέθοδος είναι μία βελτίωση της αυτόνομης δρομολόγησης που βασίζεται στην επίγνωση της κυκλοφοριακής κατάστασης στην τρέχουσα θέση που βρίσκεται το μεταφορικό μέσο. Αυτό μπορεί να επιτευχθεί μέσω της διαδραστικής επικοινωνίας χρήστη-συστήματος πλοήγησης ή μέσω κάποιας κάμερας καταγραφής της κυκλοφοριακής συμφόρησης στα οδικά τμήματα πλησίον του μεταφορικού μέσου. Η μέθοδος αυτή στηρίζεται στην αρχή ότι λόγω της αβεβαιότητας σχετικά με τις κυκλοφοριακές συνθήκες δεν είναι αποδοτικό να γίνεται αναζήτηση για τη βέλτιστη διαδρομή από την αρχή του προβλήματος. Αντίθετα, καλύτερα αποτελέσματα έχει η δημιουργία στρατηγικών επίλυσης του προβλήματος οι οποίες βασίζονται σε τοπικά δεδομένα. Σε πολλές περιπτώσεις τα ρεαλιστικά δεδομένα για το δίκτυο μπορεί να είναι ελάχιστα ή να μην αντιπροσωπεύουν τις παρούσες συνθήκες και το σύστημα πλοήγησης να ενημερώνεται για το χρόνο διάνυσης των συνδέσμων μόλις το μεταφορικό μέσο φτάσει σε έναν κόμβο που συνδέεται άμεσα με αυτούς.

Το πρόβλημα αυτό παρουσιάστηκε αρχικά από τον Croucher (1978) και επεκτάθηκε στη συνέχεια από τους Andreatta and Romeo (1988). Κατά την ανάλυσή του θεωρείται δίκτυο G={N,M} και w(i, j) ο χρόνος διάνυσης του συνδέσμου i, j που μπορεί να είναι μία τυχαία άγνωστη μεταβλητή η οποία ακολουθεί μία πιθανοτική κατανομή. Επίσης θεωρούνται z = {1,…, Z} διαφορετικά σενάρια σχετικά με τα κόστη διάνυσης των συνδέσμων. Για κάθε σενάριο y ϵ z υπάρχουν αποθηκευμένες τιμές σχετικά με τα βάρη συνδέσμων. Κατά την αρχικοποίηση του προβλήματος επιλέγεται ένα σενάριο c ϵ z και όλοι οι σύνδεσμοι θεωρείται πλέον ότι έχουν κόστος w_c (i, j). Με αυτή τη θεώρηση τα κόστη συνδέσμων αντιμετωπίζονται ως εξαρτημένες τυχαίες μεταβλητές. Θα μπορούσαν να αντιμετωπιστούν και ως ανεξάρτητες μεταβλητές χωρίς πρόβλημα, αν και στην πραγματικότητα η αύξηση της συμφόρησης σε ένα σύνδεσμο συνήθως επηρεάζει και τους γειτονικούς του συνδέσμους. Καθώς το σύστημα πλοήγησης αναζητά τη βέλτιστη διαδρομή από έναν κόμβο r σε έναν κόμβο s, αυτή υπολογίζεται με χρόνους διάνυσης συνδέσμων σύμφωνα με το σενάριο c που επιλέχθηκε πρώτο. Καθώς το μεταφορικό μέσο συνεχίζει την επιλεγμένη διαδρομή, φθάνοντας στο δεύτερο κόμβο πληροφορείται για τους πραγματικούς χρόνους διάνυσης των συνδέσμων που συνδέονται με αυτόν. Η μέθοδος αυτή αντιμετωπίζει με στοχαστικό τρόπο τους χρόνους διάνυσης των συνδέσμων και οι πληροφορίες που λαμβάνονται εξαρτώνται από τη διαδρομή που επιλέγεται. Για το λόγο αυτό η σωστή αρχική επιλογή ενός σεναρίου c={1,..,Z} είναι

κομβικής σημασίας για τον προσδιορισμό της βέλτιστης διαδρομής. Αν σε κάποιο βήμα διαπιστωθεί ότι το σενάριο c που επιλέχθηκε έχει χρόνους διάνυσης αρκετά διαφορετικούς από αυτούς που διαπιστώνονται στην πράξη μπορεί να γίνει επιλογή άλλου σεναρίου z' για το υπόλοιπο της διαδρομής. Έτσι επανυπολογίζεται η βέλτιστη διαδρομή από το σημείο που βρίσκεται πλέον το μεταφορικό μέσο προς τον κόμβο τελικού προορισμού με βάρη συνδέσμου $w_{z'}$. Για το λόγο αυτό καλό είναι τα διαφορετικά σενάρια να αποθηκεύονται ταξινομημένα σε σειρά από αυτά με τους μικρότερους χρόνους διαδρομής προς αυτά με τους μεγαλύτερους. Ένας αλγόριθμος που υλοποιεί την προτεινόμενη μέθοδο δρομολόγησης έχει τη μορφή :

ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΗΝ ΕΥΡΕΣΗ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΥΠΟ ΣΥΝΘΗΚΕΣ ΑΒΕΒΑΙΟΤΗΤΑΣ

Έστω μία λίστα με {1,Z} διαφορετικά σενάρια σχετικά με τους χρόνους διάνυσης των συνδέσμων.
Ονόμασε κόμβο 'οδηγό' τον κόμβο προέλευσης u ← r
T ← 0
$T_1 \leftarrow 0$, $T_2 \leftarrow 0$,$T_3 \leftarrow 0$, ……., $T_Z \leftarrow 0$
Q(u) ← 0
ΑΡΧΗ
Διάβασε τους πραγματικούς χρόνους διάνυσης των συνδέσμων που έχουν κόμβο προέλευσης τον u.
Άθροισε τους πραγματικούς χρόνους διάνυσης των συνδέσμων που έχουν κόμβο προέλευσης τον κόμβο u : T ← $\sum$ t(u, j) + T
Επανέλαβε την ίδια διαδικασία για τους χρόνους διάνυσης των συνδέσμων που έχουν κόμβο προέλευσης τον κόμβο u σε όλα τα σενάρια :
$T_1 \leftarrow \sum t_1(u,j) + T_1$, $T_2 \leftarrow \sum t_2(u,j) + T_2$, ……. , $T_Z \leftarrow \sum t_Z(u,j) + T_Z$
Επέλεξε το σενάριο k για το οποίο : T-$T_k \leq$ T-T_i , ∀ i ϵ{1,Z}
Για Κάθε I = 1, N όπου N ο αριθμός των κόμβων στο δίκτυο
  Εάν Q(I) ≠ 0 τότε :
    Για κάθε J = 1, N
    t(i, j) ← t_k(i, j)
    Τέλος για κάθε
  Τέλος Εάν
Τέλος Για κάθε
Χρησιμοποίησε έναν αλγόριθμο επίλυσης του προβλήματος εύρεση της βέλτιστης διαδρομής από έναν κόμβο προέλευσης σε έναν κόμβο προορισμού με βάση το τρέχον επιλεγμένο σενάριο k και επέλεξε τον επόμενο κόμβο C πού συνδέεται με τον u και ανήκει στη βέλτιστη διαδρομή.
u ← C
Q(u) ← 0
Εάν u ≠ s όπου s ο κόμβος προορισμού επέστρεψε στην ΑΡΧΗ
Τερμάτισε τη διαδικασία

7.4.2.2.ΑΥΤΟΝΟΜΗ ΔΡΟΜΟΛΟΓΗΣΗ ΜΕ ΧΡΗΣΗ ΣΤΟΧΑΣΤΙΚΩΝ ΜΕΘΟΔΩΝ ΕΠΙΛΟΓΗΣ ΔΙΑΔΡΟΜΗΣ

Η μέθοδος αυτή καλείται να αντιμετωπίσει το πρόβλημα της επιλογής της συντομότερης διαδρομής υπό συνθήκες αβεβαιότητας (Optimal Route Planning Under Uncertainty) καθώς υπάρχει ελλιπής ενημέρωση για τις κυκλοφοριακές συνθήκες που επικρατούν στο δίκτυο. Οι χρόνοι διάνυσης των συνδέσμων δε θεωρούνται γνωστοί καθώς όταν χρησιμοποιείται η αυτόνομη δρομολόγηση δεν αντιπροσωπεύουν την πραγματικότητα. Για το λόγο αυτό οι χρόνοι διάνυσης θεωρούνται πλέον ως τυχαίες μεταβλητές που ακολουθούν κάποια κατανομή. Στοιχεία για τους χρόνους αυτούς ώστε να επιλεγεί η κατανομή που ακολουθούν, η μέση τιμή και η διασπορά τους μπορεί να ληφθούν με χρήση ιστορικών στοιχείων για το δίκτυο. Η επιτυχία αυτής της μεθόδου εξαρτάται από την επιτυχία του γενικότερου μοντέλου προβλέψεων. Μέχρι σήμερα έχει γίνει μικρή έρευνα σχετικά με τη δημιουργία θεωρητικών μοντέλων που αντιμετωπίζουν ευθέως την αβεβαιότητα αυτή και υπολογίζουν μία βέλτιστη διαδρομή με μοναδικές πληροφορίες τους στοχαστικούς πλέον χρόνους διάνυσης των συνδέσμων. Εάν θεωρηθεί ότι σε ένα αρχείο δεδομένων είναι αποθηκευμένα ιστορικά στοιχεία των χρόνων διάνυσης των συνδέσμων, τότε για δίκτυο G={N,E} μπορεί για κάθε σύνδεσμό e ϵ {1,E} ο χρόνος διάνυσής του να ακολουθεί μία κατανομή με συνάρτηση πυκνότητας πιθανότητας, $f_e(x) \geqslant 0$, ∀xϵR όπου $\int_{-\infty}^{+\infty} f_e(x)\,dx = 1$.

Για κάθε σύνδεσμο μπορεί να χρησιμοποιείται διαφορετική κατανομή, όμως υπάρχουν κάποιες γενικές αρχές που θα πρέπει να ληφθούν υπόψη καθώς η πλήρης κατανόηση των όρων που διέπουν το στοχαστικό σύστημα είναι απαραίτητη προϋπόθεση ώστε να ληφθεί το κατάλληλο μοντέλο που θα περιγράψει τη συμπεριφορά του. Για να επιλεγεί λοιπόν η κατανομή που ακολουθεί ο χρόνος διάνυσης κάθε συνδέσμου πρέπει να επιλεγεί αρχικά μία κατανομή η μορφή της οποίας να είναι παρόμοια με την κατανομή που φαίνεται να ακολουθούν τα δεδομένα. Στη συνέχεια προσδιορίζονται οι τιμές των παραμέτρων της κατανομής που επιλέχθηκε ώστε να εξασφαλίζεται όσο το δυνατόν καλύτερη σύμπτωση των τιμών αυτής της κατανομής και των δεδομένων χρόνων διάνυσης. Στο τέλος εξετάζεται μέσω ελέγχου υποθέσεων εάν η κατανομή που επιλέχθηκε είναι ικανοποιητική σε κάποιο επίπεδο σημαντικότητας. Η πιο εύκολη μέθοδος προσαρμογής της κατανομής σε μετρήσεις είναι η μέθοδος των ροπών, από την οποία προκύπτουν οι τιμές των παραμέτρων της κατανομής. Μία άλλη μέθοδος που μπορεί να χρησιμοποιηθεί επίσης είναι η μέθοδος της μέγιστης πιθανοφάνειας. Στο τέλος γίνεται έλεγχος καλής προσαρμογής εφαρμόζοντας το κριτήριο χ^2, όπου :
$\chi^2 = \sum_i^c \frac{F_{i\delta} - F_{i\theta}}{F_{i\theta}}$, όπου c ο αριθμός των διαφορετικών κλάσεων, $F_{i\delta}$ η παρατηρηθήσα συχνότητα από τα δεδομένα που βρίσκονται στην ομάδα i και $F_{i\theta}$ η θεωρητική συχνότητα που προκύπτει από την κατανομή. Η παραπάνω τιμή χ^2 έχει βρεθεί ότι ακολουθεί την κατανομή χ^2 με βαθμούς ελευθερίας ν = c-1-A, όπου A ο αριθμός των παραμέτρων της θεωρητικής κατανομής που υπολογίστηκαν από τις μετρήσεις. Εάν η

τιμή του χ^2 είναι μικρότερη από την τιμή που προκύπτει από τους πίνακες της κατανομής χ^2 για ν βαθμούς ελευθερίας σύμφωνα και με το επίπεδο σημαντικότητας που έχει επιλεγεί, τότε γίνεται η κατανομή που προτείνεται είναι αποδεκτή ως προς τον έλεγχο καλής προσαρμογής.

Θεωρείται αρχικά πως για κάθε σύνδεσμό e ϵ {1,E} ο χρόνος διάνυσής του ακολουθεί μία κατανομή με συνάρτηση πυκνότητας πιθανότητας, $f_e(x)$ και οι χρόνοι διάνυσης διαφορετικών συνδέσμων είναι μεταβλητές ανεξάρτητες μεταξύ τους. Εάν X, Y τυχαίες μεταβλητές των χρόνων διάνυσης δύο συνδέσμων, επειδή θεωρούνται ανεξάρτητες :

P(A∩B) = P(XϵA)×P(YϵB) για όλα τα διαστήματα A, B ϵ R και εάν X_i τυχαίες μεταβλητές των χρόνων διάνυσης όλων των συνδέσμων με i=1,..,E τότε :

P[$\bigcap_{i=1}^{E}(x_i \epsilon B_i)$] = $\prod_{i=1}^{E} P(X_i \epsilon B_i)$ για όλα τα διαστήματα B_i, i=1,2,..,E ϵ R

Εάν μεταξύ δύο κόμβων r, s υπάρχει ένα σύνολο εναλλακτικών διαδρομών P(r,s), τότε κάθε εναλλακτική διαδρομή pϵP(r,s). Σύμφωνα με τη θεωρεία αποφάσεων δεν είναι δόκιμη η επιλογή της συντομότερης διαδρομής διότι θα πρέπει να ληφθεί υπόψη και η αξιοπιστία της. Ορίζεται πλέον ως υποθετικά συντομότερη διαδρομή η διαδρομή p_* για την οποία :

p_* ϵ p | $\sum_{e \epsilon p_*} \mu(e) = \min \sum_{e \epsilon p} \mu(e)$, για ∀ p ϵ P(r, s)

Ως υποθετικά συντομότερη δηλαδή θεωρείται πλέον η διαδρομή για την οποία το άθροισμα της μέσης τιμής των χρόνων διάνυσης των συνδέσμων της είναι το ελάχιστο. Στην ακραία περίπτωση που επιδιώκεται η επιλογή της πιο αξιόπιστης διαδρομής p_{**}, τότε αυτή προσδιορίζεται από την επίλυση του προβλήματος :

p_{**} ϵ p | $\sum_{e \epsilon p_{**}} \sigma^2(e) = \min \sum_{e \epsilon p} \sigma^2(e)$, για ∀ p ϵ P(r, s)

Το πρόβλημα πλέον είναι η δημιουργία μίας πρακτικής ώστε να επιλεγεί μία διαδρομή που να είναι ταυτόχρονα σύντομη και αξιόπιστη. Αυτό εξαρτάται από το πόσο σύντομη και πόσο αξιόπιστη επιθυμούμε να είναι. Στην ουσία δηλαδή η προτεινόμενη διαδρομή πρέπει να υπακούει κάποιο κριτήριο. Για το λόγο αυτό μπορεί να προταθεί ο συνολικός χρόνος διάνυσής της να είναι με βεβαιότητα μικρότερος από κάποια χρονική τιμή K ϵ R^+.

Εάν μία εναλλακτική διαδρομή αποτελείται από έναν σύνδεσμο, τότε η πιθανότητα να φθάσει ο χρήστης στον κόμβο προορισμού σε χρόνο μικρότερου του K είναι :

F(K) = P[X≤K] = $\int_{-\infty}^{K} f(x)\,dx$

Εάν η διαδρομή αυτή αποτελείται από τους συνδέσμους x_i, i=1,2,…,n των οποίων τα κόστη διάνυσης είναι ανεξάρτητες τυχαίες μεταβλητές με συναρτήσεις πυκνότητας

πιθανότητας $f_i(x_i)$, τότε η πιθανότητα αυτής της διαδρομής να φθάσει στο προορισμό σε χρόνο K είναι :

F(K) = $\int_{-\infty}^{k_1} \int_{-\infty}^{k2} \dots \int_{-\infty}^{k_n} f_1(x_1) f_2(x_2) \dots . f_n(x_n)\, dx_1 dx_2 \dots dx_n$

Ορισμένες φορές οι χρόνοι διάνυσης των συνδέσμων (X) θεωρείται ότι ακολουθούν την κανονική κατανομή με παραμέτρους μ, σ όπου (-∞<μ<+∞, σ>0) και γράφεται X~N(μ,σ^2). Σε αυτή την περίπτωση η συνάρτηση πυκνότητας πιθανότητας είναι :

$f_e(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, όπου $\int_{-\infty}^{+\infty} f_e(x)\, dx = 1$

και στο διάστημα (μ-3σ, μ+3σ) περιέχονται όλες σχεδόν οι τιμές της τυχαίας μεταβλητής X (χρόνος διάνυσης συνδέσμων). Το πλεονέκτημα κατά τη χρήση της κανονικής κατανομής είναι ότι εάν τα κόστη διάνυσης όλων των συνδέσμων ακολουθούν την κανονική κατανομή, τότε και το κόστος διάνυσης των διαδρομών από το σημείο προέλευσης στο σημείο προορισμού ακολουθεί και αυτό την κανονική κατανομή με μέση τιμή $\mu_r = \sum_{e\,\epsilon r} \mu_e$, όπου r η διαδρομή. Ωστόσο, δεν ισχύει το ίδιο και για τη διασπορά. Έτσι γνωρίζοντας τη μέση τιμή των εναλλακτικών διαδρομών μπορεί να επιλεγεί η κατάλληλη διαδρομή.

Ωστόσο πρακτικά αν και χρησιμοποιείται ευρύτατα η κανονική κατανομή δε μπορεί να περιγράψει μία μεταβλητή όπως ο χρόνος διάνυσης των συνδέσμων, καθώς ο χρόνος είναι μέγεθος που παίρνει μόνο θετικές τιμές. Για το λόγο αυτό και εάν γίνει αποδεκτό ότι σε συνθήκες ελεύθερης ροής οι αφίξεις των οχημάτων είναι τυχαίες (ακολουθούν την κατανομή Poisson) μπορεί να χρησιμοποιηθεί για την περιγραφή του χρόνου διάνυσης των συνδέσμων η κατανομή Γάμμα. Η κατανομή αυτή έχει χρησιμοποιηθεί για τον ίδιο λόγο από τους Fan et al. (2005).

Μπορεί να γραφεί t_e~ γ (α_e, b_e) που σημαίνει ότι ο χρόνος διάνυσης του συνδέσμου e ακολουθεί την κατανομή Γάμμα με α_e, b_e θετικές παραμέτρους. Για να ακολουθεί μία τυχαία μεταβλητή την κατανομής Γάμμα η συνάρτηση πυκνότητας πιθανότητας πρέπει να είναι :

$f(t_e) = \frac{b_e^{a_e}}{\Gamma(\alpha_e)} {t_e}^{\alpha_e - 1} e^{-\lambda t}$, t ≥ 0 , όπου $\Gamma(\alpha_e) = \int_0^{+\infty} y^{\alpha_e - 1} e^{-y}\, dy$ και $f(t_e) = 0$, t< 0.

Μέση Τιμή $\mu_e = \int_{-\infty}^{+\infty} t_e\, f(t_e) dt = E(T) = \frac{b_e}{\alpha_e}$, Διασπορά $\sigma_e^2 = E(T^2) - \mu_e^2 = \frac{b_e}{\alpha_e^2}$

Εάν m και s^2 ο μέσος όρος και η διακύμανση των χρόνων διάνυσης του συνδέσμου που προκύπτουν από τις μετρήσεις, τότε οι παράμετροι της κατανομής προσδιορίζονται από τις σχέσεις : $\alpha_e = \frac{m^2}{s^2}$, $b_e = \frac{m}{s^2}$

Η τιμή της συνάρτησης κατανομής πιθανότητας F(t) = P(T≤t) δε μπορεί να υπολογιστεί αναλυτικά παρά μόνο με χρήση H/Y ή μεθόδους αριθμητικής ανάλυσης καθώς δε μπορεί να υπολογιστεί το ολοκλήρωμα F(t) = $\int_0^t f(x)\, dx$. Βέβαια όπως

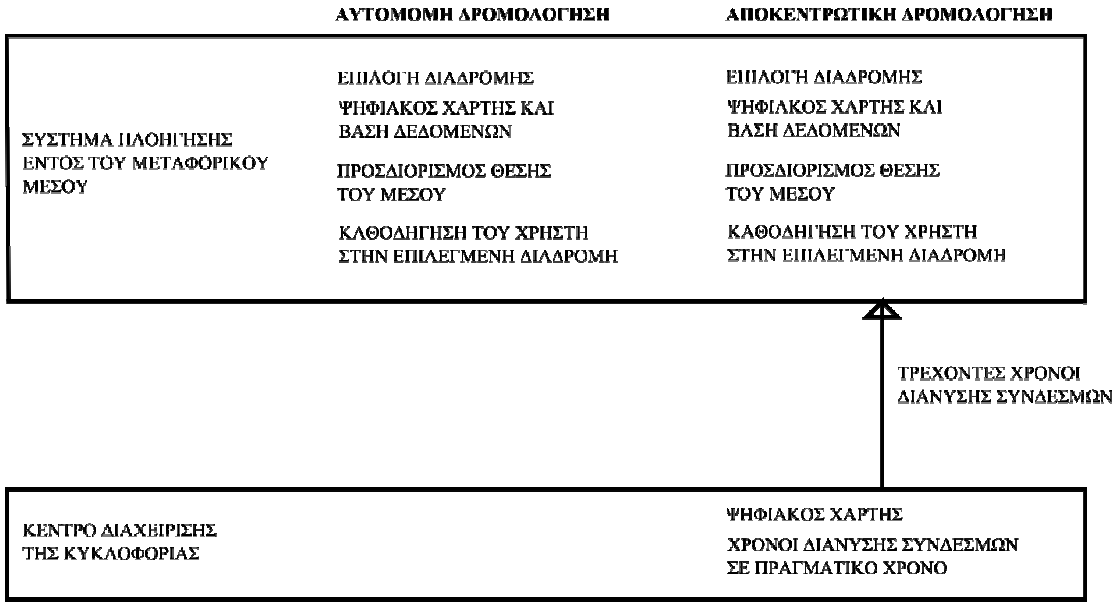
αναφέρθηκε ήδη δεν υπάρχει πιθανότητα οι χρόνοι διάνυσης να πάρουν αρνητική τιμή καθώς $P(T \leq 0) = F(0) = \int_{-\infty}^{0} f(x)\,dx = \int_{-\infty}^{0} 0\,dx = 0$.

Το ολοκλήρωμα αυτό είναι δυνατό να υπολογιστεί αναλυτικά όταν η σταθερά α_e είναι ακέραιος αριθμός, οπότε $\Gamma(y) = (y-1)!$. Η ειδική αυτή κατανομή είναι γνωστή ως κατανομή Erlang, οπότε προκύπτει

$$F(t_e) = 1 - \sum_{i=1}^{a_e - 1} \frac{e^{-b_e x}(b_e x)^i}{i!}$$

7.4.2.3.ΑΠΟΚΕΝΤΡΩΤΙΚΗ ΔΡΟΜΟΛΟΓΗΣΗ - Η ΑΝΤΙΔΡΑΣΤΙΚΗ ΜΕΘΟΔΟΣ ΤΩΝ ΣΥΝΕΧΟΜΕΝΩΝ ΕΠΑΝΥΠΟΛΟΓΙΣΜΩΝ

Μέχρι στιγμής το πρόβλημα αντιμετωπίστηκε υπό το πρίσμα της αυτόνομης δρομολόγησης. Σε αυτό τον τύπο δρομολόγησης θεωρείται ότι το σύστημα πλοήγησης δεν ανταλλάσει πληροφορίες με κανένα άλλο μέσο. Αυτή η θεώρηση όμως είναι συνυφασμένη με πολλές παθογένειες και περιορισμούς που δε μπορούν να αντιμετωπιστούν. Στη συνέχεια προτείνεται ένας νέος τύπος δρομολόγησης, γνωστός ως αποκεντρωτική αρχιτεκτονική, στην οποία το σύστημα πλοήγησης ανταλλάσει πληροφορίες με ένα άλλο μέσο. Το μέσο αυτό μπορεί να είναι κάποιο κέντρο διαχείρισης της κυκλοφορίας ή μία βάση που θα παρέχει αναβαθμισμένα δεδομένα κυκλοφοριακών στοιχείων. Στο εξής αυτό το μέσο θα ονομάζεται κεντρικός διαχειριστής. Στο ακόλουθο σχήμα παρουσιάζονται οι διαφορές των δύο παραπάνω τύπων δρομολόγησης.



Σχήμα 7.2.Χαρακτηριστικά Αυτόνομης και Αποκεντρωτικής δρομολόγησης

Ο παραπάνω τύπος δρομολόγησης βασίζεται στη συνεχή λήψη πληροφοριών σχετικά με τις κυκλοφοριακές συνθήκες που επικρατούν στο δίκτυο από το κέντρο διαχείρισης της κυκλοφορίας. Η μέθοδος αυτή χρησιμοποιεί αλγορίθμους με μικρούς

χρόνους εκτέλεσης, όπως τους αλγορίθμους εύρεσης της βέλτιστης διαδρομής μεταξύ δύο κόμβων που αναπτύχθηκαν στις Ενότητες 5.2., 5.4. ή τον αλγόριθμο των Ramalingham, Reps (1996) που αναπτύχθηκε στην Ενότητα 5.10.2. και ενδείκνυται για συνεχείς επανυπολογισμούς των δικτύων. Η μέθοδος βασίζεται στη συνεχόμενη ροή κυκλοφοριακών δεδομένων, κάτι που στα αρχικά της στάδια αποτελούσε και το μεγαλύτερο μειονέκτημά της : Friesz et al. (1993). Επειδή όμως στηρίζεται στη συνεχή ενημέρωση έχει τη δυνατότητα να χρησιμοποιεί αλγορίθμους με μικρό υπολογιστικό κόστος καθώς δεν υπάρχει μεγάλη ανάγκη για πρόβλεψη των μελλοντικών κυκλοφοριακών συνθηκών.

Στην ιδεατή περίπτωση η παρεχόμενη ενημέρωση είναι συνεχής, κάτι τέτοιο όμως δεν είναι δυνατό να επιτευχθεί. Για το λόγο αυτό θεωρείται ότι τα κυκλοφοριακά δεδομένα αναβαθμίζονται σε χρόνο k ϵR^+ , δηλαδή ανά διακριτά χρονικά διαστήματα. Εάν ο χρόνος διάνυσης κάθε συνδέσμου προκύπτει από τη συνάρτηση που προτάθηκε στην Ενότητα 7.3. ως :

$t_a(q_{a,t}) = t_a^0$ [1 + 0,15 $(q_{a,t} / q_{a,cap})^4$] , όπου $q_{a,t}$ ο φόρτος του συνδέσμου α κατά τη χρονική στιγμή t που ξεκινά η μετακίνηση και t_a^0 το κόστος διάνυσης του συνδέσμου α σε συνθήκες ελεύθερης ροής, τότε μπορεί να δοθεί η μαθηματική μορφή του προβλήματος ως εξής :

r, s : τα δύο άκρα της μετακίνησης

P(r,s) : το σύνολο των εναλλακτικών διαδρομών p μεταξύ των κόμβων r, s

Z = Min $\sum_{a \epsilon p} t(q_{a,t})$: Υπολογισμός της προτεινόμενης διαδρομής p_*

Εάν t = t_0 , όπου t_0 ο χρόνος στον οποίο αλλάζουν τα κυκλοφοριακά δεδομένα, τότε

r^* η τρέχουσα θέση του μεταφορικού μέσου τη χρονική στιγμή t_0

Για κάθε σύνδεσμο α : $q_{a,t} = q_{a,t_0}$

Για κάθε σύνδεσμο α : $t_a(q_{a,t}) = t_a^0$ [1 + 0,15 $(q_{a,t} / q_{a,cap})^4$], ο νέος χρόνος διάνυσης του συνδέσμου α

P(r^*,s) : το σύνολο των εναλλακτικών διαδρομών p μεταξύ των κόμβων r^*, s

Z = Min $\sum_{a \epsilon p} t(q_{a,t})$: Υπολογισμός της νέας προτεινόμενης διαδρομής p_*

Το παραπάνω πρόβλημα επιλύεται με **φ** επαναλήψεις ενός αλγορίθμου εύρεσης της βέλτιστης διαδρομής μεταξύ δύο κόμβων, όπου **φ** ο αριθμός των αναβαθμίσεων στα κυκλοφοριακά δεδομένα μέχρι την ολοκλήρωση της διαδρομής. Μία ανάλυση της παραπάνω μεθόδου παρουσιάστηκε από τους Kaufman and Smith (1993).

Με βάση το μαθηματικό μοντέλο απαιτούνται τα εξής στάδια υπολογισμού : Υπολογισμός της συντομότερης διαδρομής με βάση τα αρχικά δεδομένα, λήψη των αναβαθμισμένων χρόνων διάνυσης των συνδέσμων, υπολογισμός της νέας συντομότερης διαδρομής με κόμβο προέλευσης τον τρέχον κόμβο στον οποίο

βρίσκεται πλέον το μεταφορικό μέσο, επανάληψη των δύο τελευταίων βημάτων κάθε φορά που αναβαθμίζονται τα δεδομένα μέχρι να ολοκληρωθεί η διαδρομή. Η παραπάνω μέθοδος έχει έναν αντιδραστικό χαρακτήρα στις μεταβολές του συστήματος, καθώς οι αλγόριθμοι που χρησιμοποιεί απαιτούν μικρό χρόνο εκτέλεσης και μπορούν να εφαρμοστούν συνεχείς επανυπολογισμοί.

7.4.2.4.Η ΜΕΘΟΔΟΣ ΤΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΕΚΤΑΚΤΩΝ ΣΥΝΘΗΚΩΝ

Με αυτό τον όρο χαρακτηρίζεται ένας τύπος δρομολόγησης που βασίζεται σε στατικά δεδομένα σχετικά με τις κυκλοφοριακές συνθήκες. Ωστόσο, παρέχεται συνεχής ροή κυκλοφοριακών δεδομένων για τους κεντρικούς οδικούς άξονες και για έκτακτα περιστατικά όπως πορείες, ατυχήματα κ.α. Με αυτό τον τρόπο δρομολόγησης δε μπορεί σε καμία περίπτωση να βελτιστοποιηθεί συνολικά ο καταμερισμός της ζήτησης, μπορούν όμως να αντιμετωπιστούν περιπτώσεις που οδηγούν σε συμφόρηση και καθυστερήσεις λόγω έλλειψης ενημέρωσης. Αυτές οι περιπτώσεις όμως δεν πρέπει να υποτιμηθούν διότι οδηγούν σε μεγάλη οικονομική και περιβαλλοντική επιβάρυνση ενώ θα μπορούσαν πολλοί εύκολα να είχαν αποφευχθεί. Επίσης, το γεγονός ότι η μέθοδος αυτή δεν απαιτεί συνεχή ροή κυκλοφοριακών δεδομένων για όλο το δίκτυο την καθιστά ιδιαίτερα ελκυστική καθώς οδηγεί σε πολύ μικρότερα κόστη μετακινήσεων σε σύγκριση με τη στατική δρομολόγηση χωρίς να καταβληθεί ιδιαίτερη προσπάθεια για τη συλλογή δεδομένων. Για να πραγματοποιηθεί αυτός ο τύπος δρομολόγησης πρέπει να υπολογιστούν τα νέα βάρη συνδέσμων για τους οποίους υπάρχουν αναβαθμίσεις.

Στο σημείο αυτό πρέπει να αναφερθεί ότι η αξία της μεθόδου διαχείρισης έκτακτων συνθηκών μεγιστοποιείται σε συγκοινωνιακά δίκτυα που δεν προσφέρουν πολλές εναλλακτικές επιλογές. Εάν για παράδειγμα μία μετακίνηση r→s μπορεί να πραγματοποιηθεί μέσω τριών κεντρικών αρτηριών και λόγω ενός συμβάντος σε μία από αυτές απομείνουν δύο εναλλακτικές επιλογές, ακόμα και αν τα δεδομένα για τα κόστη συνδέσμων είναι στατικά και δεν προσδιορίζουν την παρούσα κατάσταση υπάρχει μεγάλη πιθανότητα να επιλεγεί η βέλτιστη διαδρομή. Σε αντίθετη περίπτωση, εάν υπάρχει για παράδειγμα πλήθος εναλλακτικών επιλογών τα στατικά βάρη των συνδέσμων θα οδηγήσουν σε επιλογή μίας διαδρομής που δεν είναι βέλτιστη, καθώς τα βάρη αυτά δεν αντιπροσωπεύουν τις παρούσες συνθήκες στο δίκτυο.

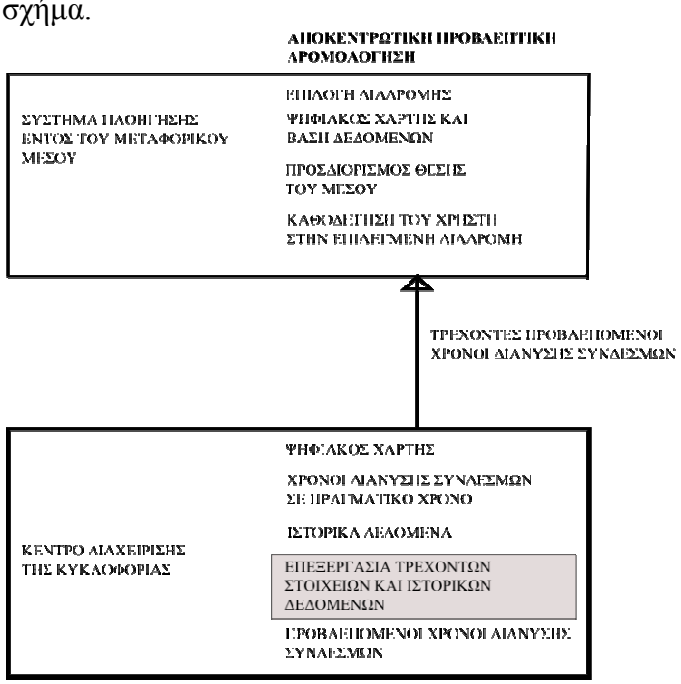
7.4.2.5.ΑΠΟΚΕΝΤΡΩΤΙΚΗ ΔΡΟΜΟΛΟΓΗΣΗ - ΠΡΟΒΛΕΠΤΙΚΕΣ ΜΕΘΟΔΟΙ

Αυτή η μέθοδος συνδυάζει δεδομένα σχετικά με τις κυκλοφοριακές συνθήκες που λαμβάνονται σε πραγματικό χρόνο και δεδομένα από άλλες πηγές ώστε να γίνουν βραχυπρόθεσμες προβλέψεις σχετικά με την κυκλοφοριακή κατάσταση του συγκοινωνιακού δικτύου. Αυτές οι προβλέψεις οδηγούν σε αναδιατυπωμένες τιμές

των χρόνων διάνυσης των συνδέσμων με τις οποίες ενημερώνονται τα συστήματα πλοήγησης.

Αυτός ο τύπος δρομολόγησης φαίνεται να είναι πιο αποδοτικός σε σύγκριση με τη δρομολόγηση μέσω της συνεχούς παροχής πληροφοριών ή την αυτόνομη δρομολόγηση με χρήση ιστορικών δεδομένων. Αυτό συμβαίνει διότι γίνεται προσπάθεια ώστε να ληφθούν υπόψη οι βραχυπρόθεσμες μεταβολές των κυκλοφοριακών συνθηκών στο δίκτυο. Έτσι προσπαθούν να προσεγγιστούν οι συνθήκες που θα αντιμετωπίσει ο χρήστης μελλοντικά όταν βρεθεί σε κάποιο σημείο του δικτύου.

Μέχρι σήμερα έχουν προταθεί ορισμένα πακέτα προσομοίωσης τα οποία χρησιμοποιούν αλγορίθμους και συνδυάζουν πραγματικά δεδομένα και ιστορικά στοιχεία με στόχο την πρόβλεψη των μελλοντικών συνθηκών και των χρόνων διάνυσης των συνδέσμων. Οι Ben-Akiva et al. (1997), (1998) πρότειναν ένα πακέτο γνωστό ως DynaMIT (Dynamic Network Assignment for the Management of Information to Travelers). Επίσης οι Kaufman et al. (1991) είχαν αναπτύξει ήδη ένα άλλο πακέτο γνωστό ως SAVaNT (Simulation of Anticipatory Vehicle Network Traffic). Εδώ πρέπει να σημειωθεί ότι και στα δύο πακέτα γίνεται προσπάθεια ώστε να δοθεί καθοδήγηση προς του χρήστες και τις διαδρομές που θα επιλέξουν, αλλά σε μικρότερο βαθμό. Ο κύριος ρόλος τους είναι η πρόβλεψη των μελλοντικών κυκλοφοριακών συνθηκών στο δίκτυο και η ενημέρωση των συστημάτων πλοήγησης με τα νέα στοιχεία. Τα πακέτα αυτά χρησιμοποιούνται από τον κεντρικό διαχειριστή, ο οποίος μέχρι τώρα αποτελούσε μία πηγή πληροφόρησης που απλώς μετέδιδε τα αναβαθμισμένα κυκλοφοριακά δεδομένα στα συστήματα πλοήγησης. Η προβλεπτική πλέον αποκεντρωτική δρομολόγηση λειτουργεί όπως παρουσιάζεται στο παρακάτω σχήμα.



Σχήμα 7.3.Χαρακτηριστικά Αποκεντρωτικής Προβλεπτικής δρομολόγησης

Τα πακέτα αυτά χρησιμοποιούν μεθόδους προσομοίωσης σχετικά με την κυκλοφοριακή ροή ώστε να προβλεφθούν οι χρόνοι διάνυσης των συνδέσμων. Σε κάθε επανάληψη του αλγορίθμου που χρησιμοποιούν αυτά τα πακέτα δε δίνονται οι τρέχοντες χρόνοι διάνυσης των συνδέσμων τ_e, e ϵ E όπως συμβαίνει στην αντιδραστική αποκεντρωτική δρομολόγηση. Αντίθετα εάν θεωρηθεί ότι ο αλγόριθμος τερματίζει την εκτέλεσή του σε χρόνο t = 0 και θα επανεκτελεστεί σε χρόνο t = T, τότε κάθε σύνδεσμος e ϵ E έχει χρόνο διάνυσης τ_e(t), tϵ{0,T}. Στην ουσία δηλαδή οι χρόνοι διάνυσης των συνδέσμων δεν παραμένουν σταθεροί μέχρι να αναβαθμιστούν ξανά τα στοιχεία του δικτύου αλλά μεταβάλλονται εντός αυτής της περιόδου. Το πρόβλημα είναι να προβλεφθούν αυτοί οι χρόνοι :
$\tau = \tau_e$(t), ∀e ϵ E, t ϵ{0,T}

Μέσω της προσομοίωσης μπορούν να δοθούν οι χρόνοι διάνυσης των συνδέσμων όταν τα μεταφορικά μέσα διέλθουν από αυτούς. Στο πακέτο SAVaNT για παράδειγμα προτάθηκε η ακόλουθη μέθοδος.

Έστω τ_e^i (t), ο χρόνος που καταγράφηκε από ένα μεταφορικό μέσο κατά τη διάνυση του συνδέσμου που βρίσκεται πριν από τον σύνδεσμο e και είναι το ιοστό που αναχωρεί από τον σύνδεσμο e σε χρόνο t. Έστω επίσης T_e^i (t) ο χρόνος προσομοίωσης όταν καταγράφηκε η τιμή τ_e^i (t)
Ο χρόνος διάνυσης ενός συνδέσμου e σε χρόνο t μεταβάλλεται κάθε φορά που ένα επιπλέον μεταφορικό μέσο αναχωρεί από το σύνδεσμο αυτό σε χρόνο t και αναβαθμίζεται σύμφωνα με τη συνάρτηση : τ_e(t) = 0.4 τ_e(t) + 0.6 τ_e^i (t)

Ο χρόνος τ_e(t) αντιπροσωπεύει το χρόνο διάνυσης του συνδέσμου, ο οποίος απαιτείται εάν ένα μεταφορικό μέσο ξεκινήσει να τον διανύει σε χρόνο t. Όμως σύμφωνα με την παραπάνω καταγραφή ο χρόνος τ_e^i (t) υπολογίζεται όταν το μεταφορικό μέσο βρίσκεται στην αρχή του συνδέσμου e, δηλαδή αφορά τον προηγούμενο σύνδεσμο του e. Για το λόγο αυτό και για να υπάρχει μεγαλύτερη ακρίβεια δε πρέπει να λαμβάνεται υπόψη ο χρόνος τ_e^i (t) για τον σύνδεσμο e, αφού αφορά προγενέστερο αλλά ο χρόνος τ_e^i (t) = T_z^i (t') - τ_z^i (t'), όπου z ο κόμβος προορισμού του συνδέσμου e. Κάτι τέτοιο όμως δεν προτείνεται από την παραπάνω μέθοδο και για το λόγο αυτό στα πρώτα στάδια της εφαρμογής της υπήρχαν μη αποδοτικά αποτελέσματα και λανθασμένες προτάσεις δρομολόγησης.

Διάφορες μέθοδοι προσομοίωσης εφαρμόζονται και από το πακέτο DynaMIT. Στο πακέτο αυτό χρησιμοποιείται μία μέθοδος προσομοίωσης σχετικά με τη ζήτηση που λαμβάνεται μέσω ιστορικών δεδομένων και δεδομένων σε τρέχον χρόνο. Ακόμα χρησιμοποιείται μία μέθοδος προσομοίωσης της προσφοράς, της δημιουργίας ουρών και της ταχύτητας διάνυσης.

Ολοκληρώνοντας την αναφορά στις αποκεντρωτικές προβλεπτικές μεθόδους θα αναφερθεί και μία βασική τους αδυναμία. Όλες οι προβλέψεις γίνονται με βάση ιστορικά στοιχεία, τρέχοντα στοιχεία και επεξεργασία τους μέσω μεθόδων προσομοίωσης. Ως αποτέλεσμα βγαίνουν κάποια συμπεράσματα σχετικά με την

αναμενόμενη κυκλοφοριακή ροή στο δίκτυο και τους χρόνους διάνυσης των συνδέσμων για το άμεσο μέλλον. Η αδυναμία έγκειται στο ότι τα συμπεράσματα μεταφέρονται στα συστήματα πλοήγησης, τα οποία επιλέγουν διαδρομές με βάση τα δεδομένα που τους παρέχονται. Ωστόσο αυτή η εκδοχή δεν έχει προβλεφθεί και δε μπορεί να αντιμετωπιστεί. Όσο δηλαδή το ποσοστό των χρηστών που διαθέτουν σύστημα πλοήγησης είναι μικρό μία πρόβλεψη για το μελλοντικό χρόνο διάνυσης των συνδέσμων είναι αποδοτική για τους χρήστη που διαθέτει σύστημα πλοήγησης. Εάν όμως το ποσοστό αυξηθεί και υπάρχουν μαζικές αντιδράσεις στις προτροπές που δίνονται από τον κεντρικό διαχειριστή το μοντέλο πρόβλεψης χάνει την αξία του.

7.4.3.ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΤΟΥ ΚΑΤΑΜΕΡΙΣΜΟΥ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ ΜΕΣΩ ΤΗΣ ΧΡΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ ΑΠΟ ΟΛΟΥΣ ΤΟΥΣ ΧΡΗΣΤΕΣ

7.4.3.1.ΕΙΣΑΓΩΓΙΚΑ ΣΤΟΙΧΕΙΑ ΚΑΙ ΔΙΑΤΥΠΩΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

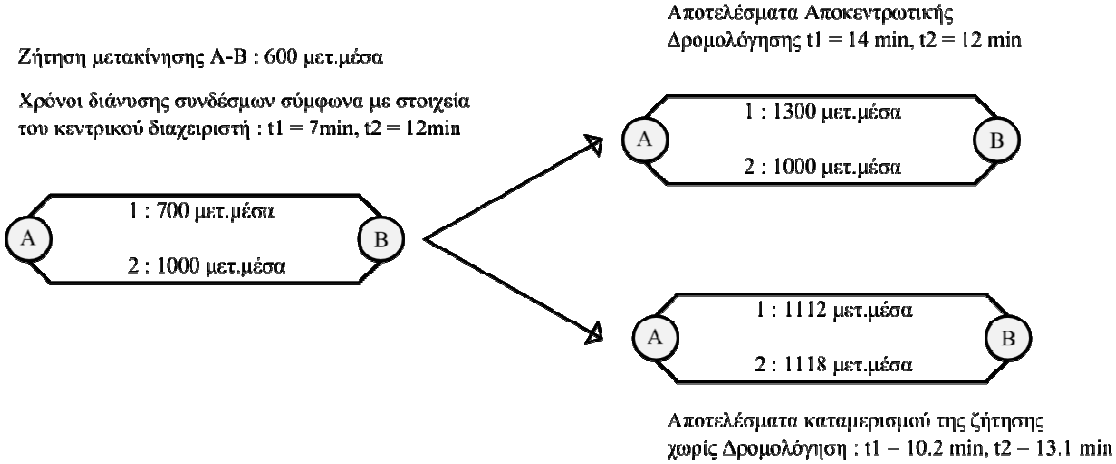
Μέχρι τώρα παρουσιάστηκαν προτάσεις για την επιλογή της βέλτιστης διαδρομής σε δυναμικά δίκτυα όπου ένα μικρό ποσοστό των χρηστών, το οποίο δεν επηρεάζει τον καταμερισμό της ζήτησης, διαθέτει συστήματα πλοήγησης. Οι προτάσεις αυτές οδηγούν στη μετάβαση από το αυτόνομο μοντέλο δρομολόγησης στο αποκεντρωτικό. Αυτές οι προτάσεις δεν οδηγούν σε βελτιστοποίηση του καταμερισμού της ζήτησης καθώς θεωρείται ότι τα συστήματα πλοήγησης δεν έχουν ευρεία χρήση. Το επόμενο ερώτημα είναι εάν η αποκεντρωτική δρομολόγηση μπορεί να αντιμετωπίσει την περίπτωση της χρήσης συστημάτων πλοήγησης από το σύνολο των χρηστών και εάν στο τέλος οδηγεί στο βέλτιστο καταμερισμό της ζήτησης.

Η απάντηση σε αυτό το ερώτημα είναι αρνητική καθώς η αποκεντρωτική δρομολόγηση αντιμετωπίζει πολλά προβλήματα στην περίπτωση που όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης. Ακόμα και σε περιπτώσεις όπου η παροχή πληροφοριών σχετικά με την κατάσταση του δικτύου από τον κεντρικό διαχειριστή είναι συνεχής προκύπτει μία αναπόφευκτη συνθήκη.

Κατά τη συνθήκη αυτή εάν πολλοί χρήστες επιθυμούν ταυτόχρονα την ίδια μετακίνηση θα προταθεί σε όλους η ίδια βέλτιστη διαδρομή με βάση τα τρέχουσες συνθήκες που επικρατούν στο δίκτυο. Ως συμπέρασμα προκύπτει ότι η αποκεντρωτική δρομολόγηση οδηγεί κάθε στιγμή σε καταμερισμό της ζήτησης στο δίκτυο με τη μέθοδο του όλα ή τίποτα. Αυτή η κατάσταση δεν επηρεάζει τον καταμερισμό της ζήτησης όταν τα συστήματα πλοήγησης δε χρησιμοποιούνται σε μεγάλο βαθμό και η αποκεντρωτική δρομολόγηση ήταν αποδοτική υπό αυτές τις συνθήκες. Όταν όμως όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης θα ακολουθούν προσωρινά την ίδια διαδρομή κατά την υλοποίηση μίας κοινής μετακίνησης. Το φαινόμενο αυτό επισημάνθηκε από τους Ben-Akiva and de Palma (1989). Ως αποτέλεσμα, αυτός ο τύπος δρομολόγησης δεν οδηγεί στο βέλτιστο καταμερισμό της ζήτησης, το αντίθετο μάλιστα, καθώς σε ορισμένες περιπτώσεις η

κατάσταση του δικτύου επιδεινώνεται. Έτσι προκύπτουν καθυστερήσεις και μειώνεται η αποδοτικότητα του συστήματος.

Μάλιστα εάν οι χρήστες γνωρίζουν ότι σε όλους προτείνεται ταυτόχρονα η ίδια διαδρομή είναι πιθανό να παρακούσουν αυτή την εντολή θεωρώντας ότι αφού θα χρησιμοποιηθεί από όλους είναι καλύτερο να ακολουθήσουν μία άλλη εναλλακτική (chicken dilemma). Η παραπάνω συνθήκη παρουσιάζεται σε μία μικρή εφαρμογή με δύο εναλλακτικές διαδρομές.



Σχήμα 7.4.Αποκεντρωτική Δρομολόγηση - Καταμερισμός της ζήτησης με τη μέθοδο όλα ή τίποτα μέχρι να ανανεωθούν τα βάρη των συνδέσμων από τον κεντρικό διαχειριστή

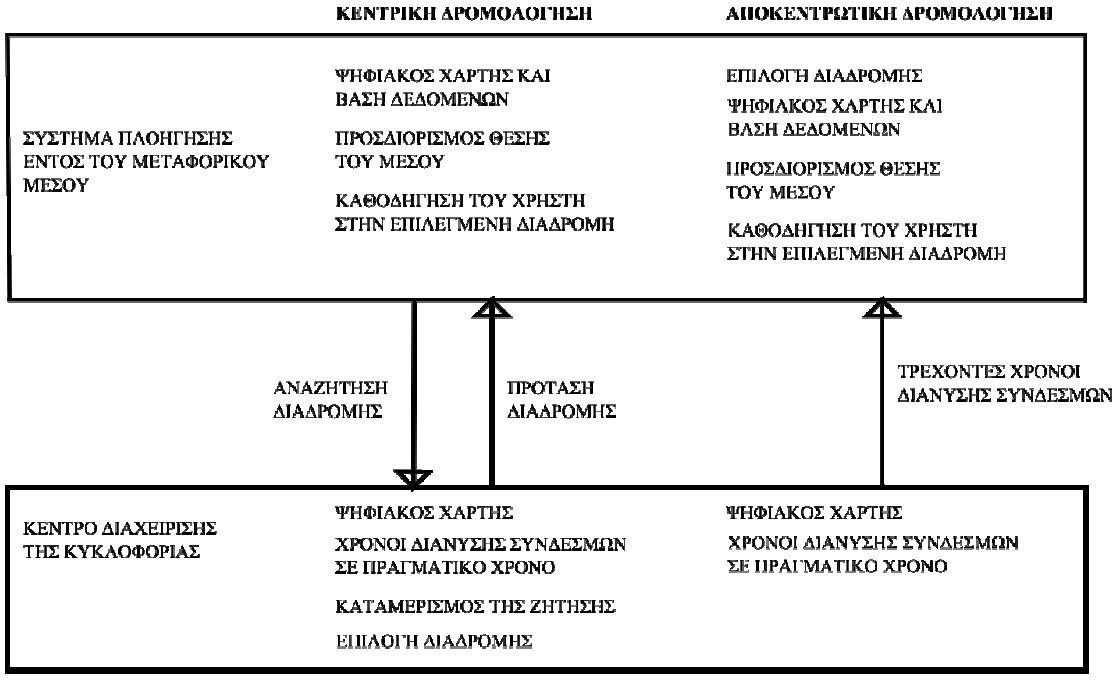
Το παραπάνω πρόβλημα δε μπορεί να αντιμετωπιστεί ούτε μέσω της προβλεπτικής αποκεντρωτικής δρομολόγησης. Εάν και η προβλεπτική αποκεντρωτική δρομολόγηση έχει καλύτερα αποτελέσματα από την αντιδραστική έχει αποδειχθεί μέσω ερευνών Kaufmann et al. (1991), Wunderlich and Smith (1992), ότι η κατάσταση στο δίκτυο επιδεινώνεται εάν το ποσοστό των χρηστών που χρησιμοποιεί σύστημα πλοήγησης ξεπεράσει το 30%. Προφανώς δεν επαρκεί ούτε η συνεχής ροή δεδομένων ώστε να βελτιστοποιηθεί ο καταμερισμός της ζήτησης μέσω των συστημάτων πλοήγησης σε δυναμικά δίκτυα. Για το λόγο αυτό, στη συνέχεια γίνονται κάποιες προτάσεις που προσανατολίζονται σε νέες βάσεις.

7.4.3.2.ΚΕΝΤΡΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Μέχρι στιγμής έχει εξεταστεί η αποκεντρωτική δρομολόγηση για την καθοδήγηση των μεταφορικών μέσων στα δυναμικά δίκτυα. Κατά την αποκεντρωτική δρομολόγηση ο κεντρικός διαχειριστής συλλέγει στοιχεία σχετικά με την κατάσταση στο δίκτυο και τα μεταφέρει στα συστήματα πλοήγησης. Λειτουργεί δηλαδή ως μία συνεχώς αναβαθμιζόμενη βάση δεδομένων. Στο νέο τύπο δρομολόγησης που προτείνεται μεταβάλλεται ο ρόλος του κεντρικού διαχειριστή. Η επιλογή της διαδρομής θα γίνεται πλέον μέσω του διαχειριστή και το σύστημα πλοήγησης θα ενημερώνεται απλά και θα καθοδηγεί το χρήστη. Αυτός ο τύπος δρομολόγησης έχει περισσότερα πλεονεκτήματα ώστε να επιτευχθεί η βελτιστοποίηση του καταμερισμού

της ζήτησης. Η επιλογή διαδρομών μέσα από έναν κεντρικό διαχειριστή είναι πιο αποδοτική, καθώς ο διαχειριστής μπορεί να ελέγξει ποιά μεταφορικά μέσα πρέπει να ακολουθήσουν μία διαδρομή και ποιά κάποιες άλλες εάν επιθυμούν όλα την ίδια μετακίνηση. Με τον τρόπο αυτό αποφεύγεται ο καταμερισμός της ζήτησης με τη μέθοδο του όλα ή τίποτα που εφαρμοζόταν από την αποκεντρωτική δρομολόγηση.

Μία αρχική έρευνα γι'αυτό τον τύπο δρομολόγησης πραγματοποιήθηκε από τον Lee (1994) με θετικά αποτελέσματα ακόμα και εάν μεγάλο ποσοστό των χρηστών διέθετε συστήματα πλοήγησης. Σύμφωνα με τη ροή των πραγμάτων φαίνεται ότι η αυτόνομη δρομολόγηση άρχισε να αφήνει τη θέση της στην αποκεντρωτική δρομολόγηση μόλις αναπτύχθηκε η ανάγκη για έγκυρη καθοδήγηση σε δυναμικά δίκτυα. Όσο το ποσοστό των χρηστών που διέθετε συστήματα πλοήγησης παρέμενε μικρό αυτή η λύση ήταν αποδεκτή. Όσο όμως το ποσοστό αυτό αυξάνεται και επηρεάζει τον καταμερισμό της ζήτησης η αποκεντρωτική δρομολόγηση πρέπει να αφήσει τη θέση της στην κεντρική δρομολόγηση. Μία παρόμοια παρατήρηση έχει διατυπωθεί και από το U.S. Dep. of Transportation (1998). Στη συνέχεια παρουσιάζονται σχηματικά οι διαφορές της κεντρικής και της αποκεντρωτικής δρομολόγησης.



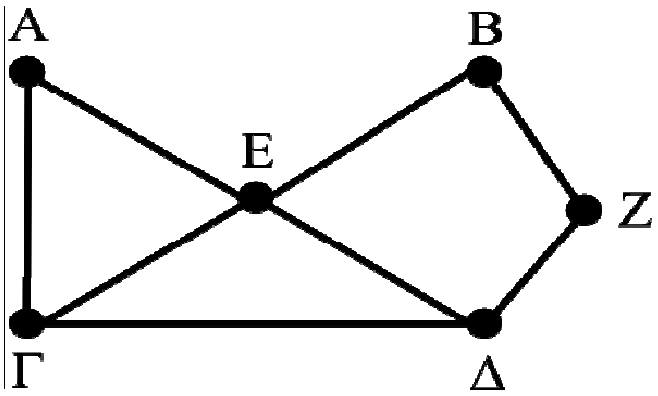
Σχήμα 7.5.Παρουσίαση της κεντρικής αρχιτεκτονικής

7.4.3.3.ΜΙΑ ΝΕΑ ΠΡΟΤΑΣΗ ΚΑΤΑΜΕΡΙΣΜΟΥ ΤΗΣ ΖΗΤΗΣΗΣ ΜΕΣΩ ΤΗΣ ΚΕΝΤΡΙΚΗΣ ΠΡΟΒΛΕΠΤΙΚΗΣ ΔΡΟΜΟΛΟΓΗΣΗΣ

Στην ενότητα αυτή ακολουθεί μία νέα πρόταση που παρουσιάζεται και σε μία εφαρμογή μικρού μεγέθους και στηρίζεται στη συνεργασία συστημάτων πλοήγησης-κεντρικού διαχειριστή. Αυτή η διαδικασία αφορά την περίπτωση που όλοι οι χρήστες

διαθέτουν συστήματα πλοήγησης και μπορεί να υποκαταστήσει ακόμα και την ανάγκη συνεχούς συλλογής δεδομένων σχετικά με τις συνθήκες που επικρατούν στο δίκτυο.

Η θέση του μεταφορικού μέσου καταγράφεται κάθε στιγμή από το σύστημα πλοήγησης, το οποίο μεταφέρει στοιχεία σχετικά με την τρέχουσα θέση του μεταφορικού μέσου και τη μετακίνηση που επιθυμεί να πραγματοποιήσει ο χρήστης. Ο κεντρικός διαχειριστής αποδέχεται τα στοιχεία αυτά, διαμορφώνει τη νέα φόρτιση του δικτύου και προτείνει μία διαδρομή στο σύστημα πλοήγησης. Η πρώτη σημαντική παρατήρηση είναι ότι εάν όλα τα συστήματα πλοήγησης επικοινωνούν με τον κεντρικό διαχειριστή, αυτός θα μπορεί να έχει μία αθροιστική εικόνα σχετικά με τη φόρτιση του δικτύου χωρίς να απαιτείται εξαντλητική συλλογή δεδομένων και αναβαθμίσεων. Αυτό το σκεπτικό αναλύεται ακολούθως.



Σχήμα 7.6.Τυχαία μορφή συγκοινωνιακού δικτύου

Θεωρείται ότι κατά την αρχικοποίηση επικρατούν συνθήκες ελεύθερης ροής : $q_e(t_0)$ = 0, ∀e ϵ E, όπου t_0 χρόνος αρχικοποίησης. Οι χρόνοι διάνυσης όλων των συνδέσμων υπό συνθήκες ελεύθερης ροής : $\tau_e(t_0)$, ∀e ϵ E είναι αποθηκευμένη αντίστοιχα στον κεντρικό διαχειριστή. Όταν το πρώτο μεταφορικό μέσο επιθυμεί τη μετακίνηση E, Z ακολουθούν οι διαδικασίες :

α) Το σύστημα πλοήγησης ενημερώνει τον κεντρικό διαχειριστή σχετικά με την τρέχουσα θέση του (E) και τη μετακίνηση E, Z που πρέπει να υλοποιηθεί.

β) Ο κεντρικός διαχειριστής προτείνει τη συντομότερη διαδρομή στο σύστημα πλοήγησης, για παράδειγμα E→Δ→Z βασιζόμενος στους χρόνους διάνυσης των συνδέσμων υπό συνθήκες ελεύθερης ροής.

γ) Ο κεντρικός διαχειριστής αναβαθμίζει αυτόματα τα δεδομένα του. Πλέον η φόρτιση του συνδέσμου E, Δ είναι $q_{E\Delta}(t)$ = 1, για t ϵ$\{t_0\ ,\ t_0+\tau_{E\Delta}(t_0)\ \}$ ενώ για οποιοδήποτε άλλο χρόνο t εξακολουθεί να είναι $q_{E\Delta}(t)$ = 0. Αντίστοιχα η φόρτιση του συνδέσμου Δ, Z είναι $q_{\Delta Z}\ (t)$ = 1, για t ϵ$\{t_0 + \tau_{E\Delta}(t_0)\ ,\ t_0+\tau_{E\Delta}(t_0) + \tau_{\Delta Z}(t_0)\}$.

δ) Μαζί με τη φόρτιση αναβαθμίζονται και τα δεδομένα των χρόνων διάνυσης των συνδέσμων. Εάν χρησιμοποιηθεί η συνάρτηση BPR, τότε :

$\tau_{E\Delta}(\ t\ ) = \tau_{E\Delta}(t_0)\ [\ 1+ 0{,}15\ (q_{E\Delta}(t)\ /\ q_{E\Delta,cap})^4]$, για $t \in \{t_0\ ,\ t_0 + \tau_{E\Delta}(t_0)\ \}$ και
$\tau_{\Delta Z}(\ t\ ) = \tau_{\Delta Z}(t_0)\ [\ 1+ 0{,}15\ (q_{\Delta Z}(t)\ /\ q_{\Delta Z,cap})^4]$, για $t \in \{t_0 + \tau_{E\Delta}(t_0)\ ,\ t_0 + \tau_{E\Delta}(t_0) + \tau_{\Delta Z}(t_0)\}$.

Συνοψίζοντας, οι χρόνοι διάνυσης των συνδέσμων μεταβάλλονται συνεχώς με το χρόνο και η έκφραση $\tau_e(t)$ σημαίνει ότι εάν κάποιος φθάσει στον κόμβο προέλευσης του συνδέσμου e σε χρόνο t απαιτείται χρόνος $\tau_e(t)$ ώστε να τον διανύσει. Η χρησιμότητα του κεντρικού διαχειριστή εμφανίζεται στο επόμενο βήμα της φόρτισης.

Έστω νέο μεταφορικό μέσο που εκκινά σε χρόνο επίσης t_0 για την πραγματοποίηση της μετακίνησης A, Z. Εάν γινόταν χρήση της αποκεντρωτικής δρομολόγησης η προτεινόμενη διαδρομή θα υπολογιζόταν με βάση τα αρχικά δεδομένα και αυτό θα συνεχιζόταν και για τα άλλα μεταφορικά μέσα, μέχρι τη στιγμή που θα αναβαθμίζονταν τα στοιχεία του κέντρου διαχείρισης. Μέχρι τότε βέβαια ο καταμερισμός της ζήτησης γίνεται με τη μέθοδο του όλα ή τίποτα. Επανερχόμενοι στην περίπτωσή μας η χρήση του κεντρικού διαχειριστή δείχνει να αντιμετωπίζει μέχρι ενός σημείου την παραπάνω παθογένεια. Για τη μετακίνηση A, Z θα επιλεγεί πλέον η συντομότερη διαδρομή ως εξής :

Αρχικά ο σύνδεσμος A, E είναι αφόρτιστος και έχει κόστος διάνυσης $\tau_{AE}(t_0)$. Στη συνέχεια συκρίνονται τα κόστη διάνυσης των διαδρομών E→B→Z και E→Δ→Z, όπου :
E→B→Z : $\tau_{EB}[\ \tau_{AE}(t_0)] + \tau_{BZ}[\ \tau_{AE}(t_0) + \tau_{EB}(\ \tau_{AE}(t_0)\ )]$, όπου $\tau_{AE}(t_0)$ ο απαιτούμενος χρόνος ώστε να φθάσει το μεταφορικό μέσο στον κόμβο E και $\tau_{AE}(t_0) + \tau_{EB}(\ \tau_{AE}(t_0)\ )$ ο απαιτούμενος χρόνος ώστε να φθάσει το μεταφορικό μέσο στον κόμβο B.
E→Δ→Z : $\tau_{E\Delta}[\ \tau_{AE}(t_0)] + \tau_{BZ}[\ \tau_{AE}(t_0) + \tau_{E\Delta}(\ \tau_{AE}(t_0)\ )]$. Το σημαντικό πλέον είναι ότι οι χρόνοι $\tau_{E\Delta}[\ \tau_{AE}(t_0)]$, $\tau_{BZ}[\ \tau_{AE}(t_0) + \tau_{E\Delta}(\ \tau_{AE}(t_0)\ )]$ δεν είναι πλέον οι χρόνοι ελεύθερης ροής εάν $\tau_{AE}(t_0) \in \{t_0\ ,\ t_0 + \tau_{E\Delta}(t_0)\ \}$ και $\tau_{AE}(t_0) + \tau_{E\Delta}(\ \tau_{AE}(t_0)) \in \{t_0 + \tau_{E\Delta}(t_0)\ ,\ t_0 + \tau_{E\Delta}(t_0) + \tau_{\Delta Z}(t_0)\}$ αντίστοιχα. Δηλαδή ενώ σε χρόνο t_0 προτάθηκε σε ένα μεταφορικό μέσο η διαδρομή E→Δ→Z, στον ίδιο χρόνο t_0 μπορεί να προταθεί σε ένα άλλο μεταφορικό μέσο η διαδρομή A→E→B→Z. Το συμπέρασμα αυτό είναι πολύ σημαντικό καθώς αντιμετωπίζεται το πρόβλημα που προέκυπτε όταν πολλοί χρήστες επιθυμούσαν την ίδια μετακίνηση και προτεινόταν σε όλους η ίδια διαδρομή μέχρι την ανανέωση των στοιχείων της κυκλοφοριακής κατάστασης. Ο καταμερισμός της ζήτησης βελτιώνεται και οι ροές φαίνεται ότι μπορούν να προσεγγίσουν τις ροές χρήστη, ωστόσο οφείλεται να διατυπωθούν εδώ τρία προβλήματα αυτής της μεθόδου :

α) Ο κεντρικός διαχειριστής πρέπει να εκτελέσει πολλούς υπολογισμούς ταυτόχρονα και για το λόγο αυτό πρέπει να χρησιμοποιεί έναν αλγόριθμο με μικρό χρόνο εκτέλεσης. Στη συνέχεια προτείνεται ο αλγόριθμος Dijkstra με διάφορες τροποποιήσεις για το σκοπό αυτό.

β) Εάν ένα μεταφορικό μέσο διασχίζει έναν σύνδεσμο σε χρόνο T και ένα άλλο μεταφορικό μέσο φθάσει στον κόμβο προέλευσης αυτού του συνδέσμου σε χρόνο ~ T δε μπορεί να θεωρείται ότι ο σύνδεσμος είναι φορτισμένος με το πρώτο μεταφορικό μέσο καθώς αυτό είναι προς αποχώρηση και δεν αναμένεται να επηρεάσει τη ροή του δεύτερου μεταφορικού μέσου που μόλις εισέρχεται. Για το λόγο αυτό μπορεί να θεωρηθεί απλοποιητικά ότι εάν ένα μεταφορικό μέσο εισέλθει σε ένα σύνδεσμο σε χρόνο T_0 και απαιτείται χρόνος T για να τον διασχίσει, τότε για τα επόμενα μεταφορικά μέσα που εισέρχονται στον σύνδεσμο σε χρόνο T_0 + T/2 η προηγούμενη φόρτιση δε λαμβάνεται υπόψη.

γ) Μία διαδρομή που αποτελείται από κάποιους συνδέσμους έχει προταθεί σε ένα μεταφορικό μέσο. Η διαδρομή αυτή προτάθηκε με βάση τις επικρατούσες συνθήκες του δικτύου. Στη συνέχεια προτείνονται και σε άλλα μεταφορικά μέσα διαδρομές και στο καθένα από αυτά η συντομότερη με βάση τις παρούσες συνθήκες που επικρατούν πριν προταθεί κάθε διαδρομή ξεχωριστά. Ωστόσο το πρώτο μεταφορικό μέσο ακολουθεί μία διαδρομή, στους συνδέσμους της οποίας έχουν εισέλθει και μεταφορικά μέσα τα οποία εκκίνησαν αργότερα από αυτό. Έτσι η διαδρομή του πρώτου μεταφορικού μέσου μπορεί να μην είναι πλέον βέλτιστη. Αυτή είναι μία πρόταση που ενδέχεται να επηρεάζει σημαντικά το βέλτιστο καταμερισμό της ζήτησης, κάτι τέτοιο όμως δε μπορεί να εξακριβωθεί παρά μόνο μέσω εφαρμογών.

ΠΡΟΤΕΙΝΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΩΝ ΣΥΝΤΟΜΟΤΕΡΩΝ ΔΙΑΔΡΟΜΩΝ ΑΠΟ ΤΟΝ ΚΕΝΤΡΙΚΟ ΔΙΑΧΕΙΡΙΣΤΗ

Βήμα 1(Αρχικοποιήσεις)

Μεταφορικό δίκτυο G={N,E} και S(e), E(e) οι κόμβοι προέλευσης και προορισμού του συνδέσμου e ϵ E

Flow(e,t) : Η φόρτιση του συνδέσμου e σε χρόνο t

τ(e,t) : Ο χρόνος διάνυσης του συνδέσμου e σε χρόνο t

Κατά την αρχικοποίηση το δίκτυο θεωρείται αφόρτιστο, οπότε

Flow(e,t) ← 0 ∀e, t

τ(e,t) ← τ(e) , όπου τ(e) ο χρόνος διάνυσης του συνδέσμου eϵE υπό συνθήκες ελεύθερης ροής

Βήμα 2(Δρομολόγηση ενός μεταφορικού μέσου)

Έστω ενημέρωση από κάποιο σύστημα πλοήγησης για ανάγκη μετακίνησης από τον κόμβο r στον κόμβο s με χρόνο εκκίνησης T.

TS ← T

u ← r

Για Κάθε e = 1,E

TP(e) ← 0 // χρόνος στον οποίο το μεταφορικό μέσο εισέρχεται στο σύνδεσμο e.

SPLink(e) ← 0 // Οι σύνδεσμοι από τους οποίους αποτελείται η συντομότερη

// διαδρομή r→s που θα υπολογιστεί στη συνέχεια.

```
Τέλος Για Κάθε
Για Κάθε i = 1,N
  Time( i ) ← 0 // χρόνος στον οποίο το μεταφορικό μέσο εισέρχεται στον κόμβο i.
  Next( i ) ← 0 // ο επόμενος κόμβος για τη δημιουργία του δένδρου συντ.διαδρομής
Τέλος Για Κάθε
Για Κάθε i = 1,N
    D( i ) ← +∞ // απόσταση κάθε κόμβου από τον κομβο προέλευσης r
    Q( i ) ← +∞
Τέλος Για Κάθε
D( r ) ← 0, Q( r ) ← 0
Όσο u ≠ s
      Για Κάθε e = 1,E
      Εάν S(e) = u & D( E(e) ) > D(S (e) ) + τ(e,TS)
          D ( E(e) ) ← D ( S(e) ) + τ(e,TS)
          Q( E(e) ) ← D ( E (e) )
          Time( E(e) ) ← TS + τ(e,TS) // χρόνος στον οποίο το μεταφορικό μέσο
                                      // εισέρχεται στον κόμβο E(e).
          TP (e) ← TS
          Next( E(e) ) ← e
      Τέλος Εάν
      Τέλος Για Κάθε
      ShortestDistance ← +∞
      Για Κάθε i=1,N
      Εάν Q(i)≠0 & D (i) < ShortestDistance
        ShortestDistance←Distan(i)
        uz←i
        Times2 ← Time(uz) // χρόνος στον οποίο το μεταφορικό μέσο εισέρχεται στον
                          // επόμενο πλησιέστερο κόμβο σε σχέση με τον αρχικό.
      Τέλος Εάν
      Τέλος Για Κάθε
      TS ← Timesss
      u ← uz
      Q(u) ← 0
Τέλος Όσο
```

Βήμα 3(Εύρεση των συνδέσμων από τους οποίους αποτελείται η διαδρομή)

```
LN ← 1 // Αριθμός Συνδέσμων από τους οποίους αποτελείται η διαδρομή
Ir ← s
Αρχή
    SPLink (LN) ← Next(Ir)
    Ir ← S(SPLink (LN))
    LN ← LN +1
    Εάν Ir ≠ r
```

Επέστρεψε στην Αρχή
Τέλος Εάν
LN ← LN - 1
Για Κάθε i=1,LN/2
IH ← SPLink (i)
SPLink (i) ← SPLink (LN+1-i)
SPLink (LN+1-i) ← IH
Τέλος Για Κάθε
Βήμα 4(Αναβάθμιση της φόρτισης των συνδέσμων και του χρόνου διάνυσής τους λόγω της τελευταίας μετακίνησης)
Για Κάθε i=1,LN
ID ← D (E(SPLink(i))) + T // Ο χρόνος στον οποίο το μεταφορικό μέσο έφθασε
// στον κόμβο E(SPLink(i))
Για Κάθε j=TP(SPLink(i)), IDistan-1 // j : χρονικό διάστημα κατά το οποίο ο
// σύνδεσμος SPLink(i) φορτίζεται λόγω αυτής της μετακίνησης.
IVar ← SPLink(i)
Flow(IVar,j) ← Flow(IVar,j) + 1 // αυξάνεται η φόρτιση του συνδέσμου
// SPLink(i) κατά ένα μεταφορικό μέσο στο χρονικό διάστημα j.
τ(IVar,j) ← τ(IVar) × (1+0.15 × ((Flow(IVar,j) / $q_{cap}(IVar)$)^4))
// μεταβάλλεται ο χρόνος διάνυσης του συνδέσμου SPLink(i) κατά το χρονικό
// διάστημα j λόγω της επιπλέον φόρτισης. Για το σκοπό αυτό χρησιμοποιήθηκε η
// συνάρτηση BPR με α=0.15 και β=4.
Τέλος Για Κάθε
Τέλος Για Κάθε

Επέστρεψε στο **Βήμα 2** και επανέλαβε τη διαδικασία για το επόμενο μεταφορικό μέσο.

Η χρησιμοποίηση του παραπάνω τύπου δρομολόγησης με χρήση του κεντρικού διαχειριστή προσφέρει και άλλα πλεονεκτήματα σε σύγκριση με την αποκεντρωτική δρομολόγηση. Ίσως το σημαντικότερο από αυτά είναι η δυνατότητα καταμερισμού της ζήτησης χωρίς ο στόχος να είναι η ισορροπία του χρήστη. Επειδή το κέντρο διαχείρισης έχει αθροιστική εικόνα της κατάστασης που επικρατεί στο δίκτυο και επιλέγει αυτό τις διαδρομές που θα ακολουθήσουν οι χρήστες μπορεί να οδηγήσει στον καταμερισμό της ζήτησης με στόχο την ισορροπία του συστήματος ώστε να χρησιμοποιηθεί πλήρως η χωρητικότητά του. Μπορεί επίσης να οδηγήσει σε καταμερισμό της ζήτησης ώστε να μειωθεί η κατανάλωση ενέργειας και οι εκπομπές CO_2 και ταυτόχρονα οι διαδρομές που προτείνονται να είναι αποδοτικές. Καμία άλλη μέθοδος δρομολόγησης δε προσφέρει αυτή τη δυνατότητα καθώς εάν παρέχονται τα ίδια δεδομένα σε όλα τα συστήματα πλοήγησης δε μπορεί να υπάρξει συνεργασία μεταξύ τους ώστε να καταμεριστεί αποδοτικά η ζήτηση.

7.5.ΚΑΤΑΜΕΡΙΣΜΟΣ ΤΗΣ ΖΗΤΗΣΗΣ ΜΕΣΩ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ ΩΣΤΕ ΝΑ ΙΚΑΝΟΠΟΙΕΙΤΑΙ ΕΝΑΣ ΠΟΛΥΚΡΙΤΗΡΙΑΚΟΣ ΣΤΟΧΟΣ

7.5.1.ΕΙΣΑΓΩΓΗ

Στις προηγούμενες ενότητες του κεφαλαίου προτάθηκαν διάφορες μέθοδοι και τύποι δρομολόγησης που μπορούν να χρησιμοποιηθούν από τα συστήματα πλοήγησης σε δυναμικά δίκτυα. Μέσω της χρήσης αυτών των μεθόδων, τα συστήματα πλοήγησης μπορούν να χρησιμοποιηθούν μαζικά και να συμβάλουν στην επίλυση του προβλήματος του καταμερισμού της ζήτησης. Από τη φύση τους τα συστήματα πλοήγησης καταμερίζουν τη ζήτηση με στόχο την ισορροπία ως προς το χρήστη καθώς προτείνουν για κάθε χρήστη τη βέλτιστη διαδρομή που πρέπει να ακολουθήσει. Η ισορροπία προκύπτει δηλαδή υπό το πρίσμα της βελτιστοποίησης κάθε επιμέρους μετακίνησης. Αποτελεί ερώτημα ωστόσο εάν και υπό ποιές συνθήκες μπορεί ο καταμερισμός της ζήτησης να βελτιστοποιηθεί περαιτέρω ώστε το σύστημα να αξιοποιείται υπό το βέλτιστο τρόπο και το συνολικό κόστος μετακινήσεων να είναι το ελάχιστο δυνατόν. Τα ζητήματα αυτά εξετάζονται σε αυτό το κεφάλαιο και αναλύονται επιμέρους ως :

Η περίπτωση στοχαστικής ισορροπίας του δικτύου, θεωρώντας ότι μόνο μικρό ποσοστό των χρηστών διαθέτει συστήματα πλοήγησης.

Ο καταμερισμός της ζήτησης σε δυναμικά δίκτυα μέσω της μαζικής χρήσης συστημάτων πλοήγησης και χρήσης της κεντρικής δρομολόγησης. Ο καταμερισμός της ζήτησης που προκύπτει μέσω αυτής της διαδικασίας προσεγγίζει την ισορροπία ως προς το χρήστη.

Ο καταμερισμός της ζήτησης σε δυναμικά δίκτυα μέσω της μαζικής χρήσης συστημάτων πλοήγησης και χρήσης της κεντρικής δρομολόγησης με στόχο την ισορροπία ως προς το σύστημα.

Ο καταμερισμός της ζήτησης ως προς το σύστημα υπό περιορισμούς. Αυτή η ισορροπία αποτελεί μία ιδεατή ισορροπία με στόχο τη βελτιστοποίηση της χρήσης του δικτύου χωρίς να προτείνονται μη αποδοτικές διαδρομές στους χρήστες, καθώς αυτές θα απορριφθούν.

7.5.2.ΣΤΟΧΑΣΤΙΚΗ ΙΣΟΡΡΟΠΙΑ ΤΟΥ ΔΙΚΤΥΟΥ ΘΕΩΡΩΝΤΑΣ ΟΤΙ ΕΝΑ ΜΙΚΡΟ ΠΟΣΟΣΤΟ ΤΩΝ ΧΡΗΣΤΩΝ ΔΙΑΘΕΤΕΙ ΣΥΣΤΗΜΑΤΑ ΠΛΟΗΓΗΣΗΣ

Όπως έχει αναφερθεί ήδη όταν ένα μικρό ποσοστό των χρηστών διαθέτει συστήματα πλοήγησης, τότε θεωρείται ότι δεν επηρεάζει τον καταμερισμό της ζήτησης. Πιο συγκεκριμένα ο καταμερισμός της ζήτησης έχει προκύψει από τις επιμέρους αποφάσεις των χρηστών κάτω από συνθήκες ελλιπούς πληροφόρησης. Η στοχαστική ισορροπία ως προς το χρήστη αποτελεί μία προσέγγιση της τελικής μορφής του καταμερισμού της ζήτησης σε ένα δυναμικό δίκτυο υπό αυτές τις συνθήκες.

Η στοχαστική ισορροπία παρουσιάστηκε από τους Daganzo and Sheffi (1977) και είναι μία από τις σημαντικότερες ισορροπίες. Βασίζεται στην πρώτη αρχή του Wardrop με τη διαφορά ότι η επιλογή διαδρομής από κάθε χρήστη βασίζεται σε τυχαία μοντέλα επιλογής. Σε πολλά μοντέλα της θεωρίας παιγνίων θεωρείται ότι υπάρχει αβεβαιότητα στη διαδικασία λήψης αποφάσεων. Ομοίως και κατά την επιλογή μετακίνησης δε μπορεί να γίνει η υπόθεση ότι οι χρήστες του δικτύου γνωρίζουν το χρόνο διαδρομής κάθε εναλλακτικής επιλογής τους. Ο τρόπος για να εκφραστεί αυτό μαθηματικά είναι να γίνει η υπόθεση ότι οι χρήστες θεωρούν ότι ο χρόνος διάνυσης κάθε διαδρομής (r) διαφέρει από τον πραγματικό χρόνο κατά μία τιμή, που μπορεί να καλείται και τυχαίο σφάλμα (random error). Έτσι ο χρόνος διάνυσης μίας διαδρομής r για έναν χρήστη του δικτύου μπορεί να θεωρηθεί ότι ισούται με τον όρο : $t_r^{αναμενόμενο} = t_r^{πραγματικό} - \varepsilon_r$. Οι χρήστες συγκρίνουν τους αναμενόμενους χρόνους διάνυσης κάθε εναλλακτικής διαδρομής και επιλέγουν τη συντομότερη από αυτές. Για να περιγραφεί καλύτερα το πρόβλημα, θεωρείται ένα ζεύγος κόμβων προέλευσης προορισμού : p, q.

Σε δυναμικά δίκτυα θεωρείται ότι ο χρόνος διάνυσης ενός συνδέσμου α επηρεάζεται από τη φόρτιση του συνδέσμου q_α, άρα : $t_a = t_a(q_\alpha)$.

Ο χρήστης δεν είναι δυνατό να γνωρίζει τον ακριβή χρόνο διάνυσης ενός συνδέσμου. Για το λόγο αυτό μπορεί να θεωρηθεί ότι ο χρόνος αυτός ισούται με τον πραγματικό χρόνο συν μία τιμή σφάλματος : $t_a(q_\alpha) = t_a(q_\alpha) - \varepsilon_\alpha$, $\forall$α.

Τα σφάλματα αυτά ακολουθούν την ίδια πιθανολογική κατανομή. Έχουν προταθεί διάφορες πιθανολογικές κατανομές, με επικρατέστερη την κατανομή Gumbel. Έτσι εάν θεωρηθεί ότι οι τυχαίες μεταβλητές των σφαλμάτων εκτίμησης έχουν ταυτόσημες κατανομές Gumbel και είναι ανεξάρτητα κατανεμημένες, τότε ο αντιληπτός χρόνος διάνυσης είτε στο επίπεδο του συνδέσμου, είτε στο επίπεδο της διαδρομής προέρχεται από ισοδύναμες εκφράσεις :
$t_r^{αναμενόμενο} = t_r^{πραγματικό} - \sum_{\alpha \in r} \varepsilon_a$, όπου r η διαδρομή και $\alpha \in r$ οι σύνδεσμοι από τους οποίους αποτελείται. Με βάση αυτήν τη θεώρηση οι τυχαίες μεταβλητές σφαλμάτων μιας διαδρομής r είναι : $\varepsilon_r = \sum_{\alpha \in r} \varepsilon_a$,$\forall$r

Στοιχεία Γενικευμένης Κατανομής Gumbel

Η γενικευμένη κατανομή Gumbel είναι μία πιθανολογική κατανομή που ορίζεται από δύο παραμέτρους β, μ>0.

Η συνάρτηση κατανομής πιθανότητας είναι : F (x) = $P_r[X \leq x] = e^{-e^{\frac{\beta - x}{\mu}}}$

Η συνάρτηση πυκνότητας πιθανότητας είναι : f (x) = $\frac{ze^{-z}}{\mu}$, όπου z = $e^{-\frac{x-\beta}{\mu}}$

Η μέση τιμή της τυχαίας μεταβλητής x είναι $\int_{-\infty}^{+\infty} xf(x)\,dx$ = β + γ μ

Η διακύμανσή της είναι $\mu^2\pi^2$ / 6

γ : σταθερά ίση με $\log_{n\to\infty}[(\sum_{k=1}^{n}\frac{1}{k}) - \ln(n)] = 0.57721566$

Η κατανομή Gumbel όμως που θα χρησιμοποιηθεί θα πρέπει να έχει μέση τιμή τυχαίας μεταβλητής ίση με μηδέν. Έτσι εάν η μεταβλητή x είναι το σφάλμα, ο μέσος όρος αυτής της τυχαίας μεταβλητής θα είναι ίσος με μηδέν, κάτι που είναι επιθυμητό να ισχύει πάντα. Έτσι το σφάλμα ε_r κατά την εκτίμηση του χρόνου διάνυσης μίας διαδρομής r από το χρήστη μπορεί να θεωρηθεί ότι ακολουθεί την μονοπαραμετρική γενικευμένη κατανομή Gumbel με παραμέτρους (β, μ) = (-γ μ, μ) ώστε να ισχύει πάντα : E[ε_r] = $\int_{-\infty}^{+\infty} \varepsilon_r f(\varepsilon_r)\,dx$ = 0, οπότε προκύπτουν :

F (ε_r) = $e^{-e^{(\frac{\varepsilon_r}{\mu}+\gamma)}}$, Διακύμανση = $\mu^2\pi^2$ / 6.

Κάθε χρήστης μπορεί να επιλέξει μία διαδρομή r κατά τη μετακίνησή του. Εάν οι εναλλακτικές διαδρομές είναι k, τότε η πιθανότητα να επιλεγεί η διαδρομή r από το χρήστη είναι :

P($t_r^{αναμ.} = \min t_i^{αναμ.}$, $\forall i \in k$) = P($t_r^{πραγ.} - \varepsilon_r = \min[\, t_i^{πραγ.} - \varepsilon_i]$, $\forall i \in k$) =

P($t_r^{πραγ.} - \varepsilon_r \leq t_i^{πραγ.} - \varepsilon_i$, $\forall i \in k$) =

P($\varepsilon_1 \leq t_1^{πραγ.}$-$t_r^{πραγ.}$+$\varepsilon_r$, $\varepsilon_2 \leq t_2^{πραγ.}$-$t_r^{πραγ.}$ + ε_r, ...) =

P($\bigcap_{i\in k, i\neq r} \varepsilon_i \leq t_i^{πραγ.}$ - $t_r^{πραγ.}$ + ε_r) =

$\int_{-\infty}^{+\infty} f_{\varepsilon_r}(x) \times \prod_{i\in k, i\neq r} F_{\varepsilon_i}(t_i^{πραγ.} - t_r^{πραγ.} + x)\,dx$ =

$\int_{-\infty}^{+\infty} \frac{1}{\mu} e^{-\frac{x}{\mu}-\gamma} \times e^{-e^{-\frac{x}{\mu}-\gamma}} \times \prod_{i\in k, i\neq r} e^{-e^{-\frac{t_i^{πραγ.} - t_r^{πραγ.} + x}{\mu} - \gamma}}\,dx$

(Έστω δ = $e^{-\frac{x}{\mu}-\gamma}$, $\frac{d\delta}{dx}$ = - $\frac{1}{\mu}$ δ $\Rightarrow dx$ = - $\frac{\mu}{\delta}\,d\delta$, $y_i = e^{-\frac{t_i^{πραγ.}}{\mu}}$), τότε :

$\int_{+\infty}^{0} \frac{1}{\mu}\,\delta e^{-\delta} \times \prod_{i\in k, i\neq r} e^{-\delta\frac{y_i}{y_r}} \times \frac{-\mu}{\delta}\,d\delta$ =

$\int_{0}^{+\infty} e^{-\delta} \times \prod_{i\in k, i\neq r} e^{-\delta\frac{y_i}{y_r}}\,d\delta = \int_{0}^{+\infty} \prod_{i\in k} e^{-\delta\frac{y_i}{y_r}}\,d\delta = \int_{0}^{+\infty} e^{-\delta\,\Sigma_{i\in k}\frac{y_i}{y_r}}\,d\delta$ =

$$\frac{-1}{\sum_{i\epsilon k}\frac{y_i}{y_r}}\ e^{-\delta\ \sum_{i\epsilon k}\frac{y_i}{y_r}}\Big|_{\delta=0}^{\delta=\infty} = \frac{1}{\sum_{i\epsilon k}\frac{y_i}{y_r}} = \frac{y_r}{\sum_{i\epsilon k} y_i} \Rightarrow$$

$\mathrm{P}(t_r^{\pi\rho\alpha\gamma\cdot} - \varepsilon_r = \min[\, t_i^{\pi\rho\alpha\gamma\cdot} - \varepsilon_i],\ \forall i \epsilon k) = \frac{e^{\frac{t_r^{\pi\rho\alpha\gamma\mu\cdot}}{\mu}}}{\sum_{i\epsilon k} e^{\frac{t_i^{\pi\rho\alpha\gamma\mu\cdot}}{\mu}}}$, δηλαδή για την επιλογή μιας διαδρομής r ανάμεσα στις εναλλακτικές της iϵk-{r} μπορεί να προταθεί το λογιστικό μοντέλο επιλογής (logit model).

Επίσης ισχύουν οι προϋποθέσεις :
$\sum_k f_k^{rs} = q^{rs}$, δηλαδή η συνολική φόρτιση μεταξύ δύο κόμβων προέλευσης προορισμού r→s που συμβολίζεται ως q^{rs} πρέπει να καταμερίζεται σε όλες τις διαδρομές k που έχουν κόμβο προέλευσης τον r και κόμβο προορισμού τον s : f_k^{rs}

$f_k^{rs} \geq 0$, δηλαδή κάθε διαδρομή k μεταξύ δύο κόμβων r, s μπορεί να παραλαμβάνει κυκλοφοριακό φόρτο ή να μη φορτίζεται.

$q_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_k^{rs}$, ∀α. Αυτός ο περιορισμός υπονοεί ότι η φόρτιση κάθε συνδέσμου α ισούται με το άθροισμα της φόρτισης που παραλαμβάνουν όλες οι διαδρομές k με κόμβο προέλευσης τον τυχαίο κόμβο r και κόμβο προορισμού τον τυχαίο κόμβο s για τις οποίες ισχύει ότι ο σύνδεσμος α είναι μέλος της διαδρομής, κάτι που συνεπάγεται ότι $\delta_k^{rs} = 1$, αλλιώς $\delta_k^{rs} = 0$.

Σύμφωνα με τις προϋποθέσεις αυτές η συνολική ζήτηση μπορεί να καταμεριστεί στο δίκτυο σταδιακά. Σε κάθε στάδιο φόρτισης η μετακίνηση r→s έχει ζήτηση $\frac{q^{rs}}{N}$, όπου N ο αριθμός των σταδιακών φορτίσεων. Μία τυχαία διαδρομή z θα παραλάβει φόρτιση ίση με :

$$f_z^{rs} = \frac{e^{\frac{t_z^{\pi\rho\alpha\gamma\mu\cdot}}{\mu}}}{\sum_{i\in k} e^{\frac{t_i^{\pi\rho\alpha\gamma\mu\cdot}}{\mu}}} \times \frac{q^{rs}}{N}$$

Κάτι τέτοιο απαιτεί τη γνώση των χρόνων διάνυσης όλων των εναλλακτικών διαδρομών για την πραγματοποίηση της μετακίνησης r → s : $t_i^{\pi\rho\alpha\gamma\mu\cdot}$, ∀iϵk. Θεωρείται όμως ότι οι χρήστες δε μπορούν να απαριθμήσουν όλες τις πιθανές εναλλακτικές διαδρομές, οι οποίες στην ακραία περίπτωση μπορεί να είναι και άπειρες. Για το λόγο αυτό λαμβάνονται υπόψη ορισμένες μόνο εναλλακτικές διαδρομές, οι οποίες έχουν και το μικρότερο κόστος. Ωστόσο για την απαρίθμηση αυτών των εναλλακτικών διαδρομών απαιτείται μεγάλος υπολογιστικός φόρτος και πολλές θέσεις μνήμης. Λόγω αυτής της αδυναμίας στη συνέχεια παρουσιάζεται η μέθοδος του Dial (1971) μέσω της οποίας μειώνεται ο αριθμός των εναλλακτικών διαδρομών χρησιμοποιώντας μόνο αποδοτικούς συνδέσμους, δηλαδή συνδέσμους που οδηγούν σε κόμβους οι οποίοι είναι πιο απομακρυσμένοι από τον κόμβο

προέλευσης. Σύμφωνα με το μοντέλο logit η επιλογή μίας διαδρομής έχει πιθανότητα

$P(t_r^{πραγ.} - \varepsilon_r = \min[\, t_i^{πραγ.} - \varepsilon_i],\ \forall i \in k) = \frac{e^{\Theta t_r^{πραγμ.}}}{\sum_{i \epsilon k} e^{\Theta t_i^{πραγμ.}}}$, όπου $\Theta = \frac{1}{\mu}$

Η χρησιμότητα επιλογής ενός συνδέσμου α : S(α)→E(α) σε σχέση με μία μετακίνηση από τον κόμβο i στον κόμβο j μπορεί να οριστεί ως :

$h_a =$ { D(E(α)) - D(S(α)) - t_a +ε_α , εάν D(E(α)) > D(S(α))
{ - ∞ , όπου D(E(α)), D(S(α)) το ελάχιστο κόστος για τη μετακίνηση i→E(α) και i→S(α) αντίστοιχα.

Ο παραπάνω ορισμός της χρησιμότητας διασφαλίζει ότι χρησιμοποιούνται μόνο αποδοτικοί σύνδεσμοι. Όπως έχει αναφερθεί ήδη σε δυναμικά δίκτυα ο χρόνος διάνυσης ενός συνδέσμου μεταβάλλεται όταν αυξηθεί η φόρτισή του. Έτσι όσο μεταβάλλεται η φόρτιση στο δίκτυο μεταβάλλεται και η χρησιμότητα επιλογής των συνδέσμων για την υλοποίηση μίας μετακίνησης. Ο καταμερισμός της ζήτησης στο δίκτυο μπορεί να υλοποιηθεί ακολουθώντας το στοχαστικό αλγόριθμο του Dial με μετατροπές όμως ώστε να υπακούει σε συνθήκες δυναμικών δικτύων. Τα βήματα του αλγορίθμου είναι :

α) Φόρτιση του δικτύου σταδιακά ώστε να λαμβάνονται υπόψη οι μεταβολές στους χρόνους διάνυσης των συνδέσμων λόγω κυκλοφοριακής συμφόρησης.

β) Υπολογισμός της πιθανότητας επιλογής κάθε εναλλακτικής διαδρομής από τον κόμβο προέλευσης προς όλους τους υπόλοιπους κόμβους προορισμού.

γ) Καταμερισμός της ζήτησης των μετακινήσεων στις διαδρομές που προέκυψαν και επιστροφή στο πρώτο βήμα.

ΑΛΓΟΡΙΘΜΟΣ DIAL ΜΕ ΤΡΟΠΟΠΟΙΗΣΕΙΣ ΓΙΑ ΕΦΑΡΜΟΓΗ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ

Μεταφορικό δίκτυο G={N,E}
Κάθε σύνδεσμος e ϵ E έχει κόμβο προέλευσης S(e) και κόμβο προορισμού E(e)
Διάβασε από ένα αρχείο τη ζήτηση για μετακινήσεις στο δίκτυο μεταξύ κάθε ζεύγους κόμβων προέλευσης-προορισμού i, j : Demand(i,j) και διαίρεσε τη ζήτηση αυτή ***Z*** φορές ώστε η ζήτηση για κάθε μετακίνηση να είναι μικρή (φόρτοι σε δεκάδες ή εκατοντάδες). Αυτό συμβαίνει ώστε να καταμεριστεί η ζήτηση σταδιακά.
Διάβασε από ένα αρχείο τους χρόνους διάνυσης των συνδέσμων σε ελεύθερη ροή ή υπό τις υπάρχουσες συνθήκες, t(e), ∀ e∈{1,E} και την κυκλοφοριακή ικανότητα των συνδέσμων q_{cap}(e) , ∀ e∈{1,E}. Οι τρέχοντες χρόνοι διάνυσης των συνδέσμων είναι t'(e) ← t(e), ∀ e∈{1,E}
Για Κάθε z = 1, ***Z***
 Για Κάθε r = 1,N // r : Κόμβος Προέλευσης
 Χρησιμοποίησε έναν αλγόριθμο εύρεσης της συντομότερης διαδρομής από τον

κόμβο r προς όλους τους υπόλοιπους κόμβους του δικτύου με χρόνο διάνυσης συνδέσμων : t'(e). Έπειτα από την εφαρμογή αυτού του αλγορίθμου προκύπτουν οι συντομότερες αποστάσεις D(i),∀iϵN κατά τη μετακίνηση r→i, ∀iϵN

Για Κάθε e=1,E
Εάν D(E(e)) > D(S(e))
h(e) ← $2.71828^{\Theta\,(D(E(e))-D(S(e))-t'(e))}$
Αλλιώς
h(e) ← 0
Τέλος Εάν
$\sum(E(e))$ ← $\sum(E(e))$ + h(e)
Τέλος Για Κάθε
Για Κάθε l = 1,N
Εάν l ≠ r & Demand(r,l) ≠ 0
T(r,l) ← Demand(r,l) / ***Z*** // T(r,l) : Ζήτηση μετακίνησης r,l σε αυτό το στάδιο
// φόρτισης
Για Κάθε i = N,1
Για Κάθε e = 1,E
Εάν E(e) = i & T(r,E(e)) ≠ 0
Flow(e) ← Flow(e) + T(r,E(e)) × [h (e) / $\sum(E(e))$] // Ο σύνδεσμος e
// παραλαμβάνει τη φόρτιση T(r,E(e)) × [h (e) / $\sum(E(e))$] σύνφωνα με
// τη χρησιμότητα επιλογής της αντίστοιχης διαδρομής
T(r,S(e)) ← T(r,E(e))*(h (e) / $\sum(E(e))$)
Τέλος Εάν
Τέλος Για Κάθε
Τέλος Για Κάθε
Τέλος Για Κάθε
Τέλος Για Κάθε
Για Κάθε e = 1,E
t'(e) ← t(e) × [1+ α (Flow(e) / $(q_{cap}(e))^{\beta}$] // αλλαγή στους χρόνους διάνυσης
// συνδέσμων λόγω φόρτισης με χρήση της συνάρτησης BPR
Τέλος Για Κάθε
Τέλος Για Κάθε

7.5.3.ΙΣΟΡΡΟΠΙΑ ΩΣ ΠΡΟΣ ΤΟ ΧΡΗΣΤΗ ΥΠΟ ΤΗΝ ΙΔΕΑΤΗ ΘΕΩΡΗΣΗ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΧΡΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

Έχουν παρουσιαστεί ήδη διάφοροι τύποι δρομολόγησης για δυναμικά δίκτυα που οδηγούν σε μία προσέγγιση της ισορροπία ως προς το χρήστη. Για την αποτίμηση του καταμερισμού της ζήτησης που προκύπτει μέσω των συστημάτων πλοήγησης αναλύεται στη συνέχεια η ισορροπία ως προς το χρήστη και οι τρόποι υπολογισμού της. Στη βιβλιογραφία υπάρχουν πολλές μαθηματικές εκφράσεις σχετικά με την πρώτη αρχή του Wardrop αλλά η πρώτη που αναπτύχθηκε από τους Beckmann et al. (1956) έχει ευρύτερη χρήση. Σε αυτήν τη μαθηματική έκφραση θεωρείται ότι τα κόστη των συνδέσμων c_α είναι συνεχείς αύξουσες συναρτήσεις ως προς τους κυκλοφοριακούς φόρτους : c_α(q). Αν θεωρηθεί ότι τα κόστη των συνδέσμων εξαρτώνται μόνο από το χρόνο διάνυσής τους, τότε c_α(q) = t_a(q). Έτσι εάν :

f(q) = $\sum_a \int_0^{q_a} t_a(q)dq$, όπου

q_a : ο κυκλοφοριακός φόρτος του συνδέσμου α

$t_a(q)$: ο χρόνος διάνυσης του συνδέσμου α ως συνεχής αύξουσα συνάρτηση του κυκλοφοριακού φόρτου στο σύνδεσμο

Και αν υπάρχουν οι περιορισμοί :

$\sum_k f_k^{rs} = q^{rs}$, δηλαδή ο συνολική φόρτιση μεταξύ δύο κόμβων προέλευσης προορισμού r→s που συμβολίζεται ως q^{rs} πρέπει να καταμερίζεται σε όλες τις διαδρομές k που έχουν κόμβο προέλευσης τον r και κόμβο προορισμού τον s : f_k^{rs}

$f_k^{rs} \geq 0$, δηλαδή κάθε διαδρομή k μεταξύ δύο κόμβων r, s μπορεί να παραλαμβάνει κυκλοφοριακό φόρτο ή να μη φορτίζεται.

$q_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_k^{rs}$, ∀α. Αυτός ο περιορισμός υπονοεί ότι η φόρτιση κάθε συνδέσμου α ισούται με το άθροισμα της φόρτισης που παραλαμβάνουν όλες οι διαδρομές k με κόμβο προέλευσης τον τυχαίο κόμβο r και κόμβο προορισμού τον τυχαίο κόμβο s για τις οποίες ισχύει ότι ο σύνδεσμος α είναι μέλος της διαδρομής, κάτι που συνεπάγεται ότι δ_k^{rs} = 1, αλλιώς δ_k^{rs} = 0.

Αναζητείται η φόρτιση κάθε συνδέσμου ώστε :

f(q) = Minimum

Η επίλυση του παραπάνω μαθηματικού μοντέλου δίνει τους φόρτους ισορροπίας ως προς το χρήστη (ροές χρήστη) :

$f_k^{rs}(t_k^{rs} - t^{rs*}) = 0$, $t_k^{rs} - t^{rs*} \geq 0$,∀k, r, s όπου t^{rs*} το κόστος ισορροπίας του χρήστη και t_k^{rs} το κόστος κάποιας διαδρομής k για τη μετακίνηση r→s.

Εάν αναλυθεί η παραπάνω σχέση προκύπτουν τα ακόλουθα συμπεράσματα :

- $t_k^{rs} - t^{rs*} \geq 0$: Μπορούν να υπάρχουν διαδρομές k στο δίκτυο που συνδέουν δύο κόμβους r, s μόνο αν τα κόστη τους είναι μεγαλύτερα ή ίσα με το κόστος ισορροπίας του χρήστη. Μικρότερο δε μπορεί να είναι γιατί κανένας χρήστης δε μπορεί να επιλέξει πιο σύντομή διαδρομή από το r στο s κατά την ισορροπία.
- $f_k^{rs}(t_k^{rs} - t^{rs*}) = 0$: Μία διαδρομή k μεταξύ δύο κόμβων προέλευσης-προορισμού r, s ή θα έχει κόστος ίσο με το κόστος ισορροπίας ως προς το χρήστη ή θα έχει μεγαλύτερο κόστος και δε θα παραλαμβάνει καθόλου φόρτιση ($f_k^{rs} = 0$). Από αυτό προκύπτει και η μέθοδος φόρτισης του όλα ή τίποτα (all or nothing) καθώς αν μία διαδρομή z είναι η συντομότερη για το ζεύγος r, s τότε ο χρόνος ισορροπίας αρχικά ισούται με το χρόνο της διαδρομής αυτής $t^{rs*} = t_z^{rs}$. Εάν όμως ο χρόνος της διαδρομής z είναι συνεχώς μικρότερος από το χρόνο οποιασδήποτε άλλης εναλλακτικής k-z, τότε $\sum_{k-z} f_{k-z}^{rs} = 0$ και $f_z^{rs} = q^{rs}$.

Ο υπολογισμός της ισορροπίας του χρήστη ακολουθεί τα εξής βήματα : Η κυκλοφορία καταμερίζεται στο δίκτυο με βάση τις διαδρομές ελαχίστου κόστους, οι χρόνοι διάνυσης των συνδέσμων υπολογίζονται εκ νέου λαμβάνοντας υπόψη τις μειώσεις της ταχύτητας λόγω της αύξησης των φόρτων και εν συνεχεία επανακαταμερίζονται με βάση τον ίδιο κανόνα. Η παραπάνω διαδικασία βέβαια ανακυκλώνεται και συνήθως δε συγκλίνει. Για το λόγο αυτό χρησιμοποιείται συνήθως η σταδιακή φόρτιση του δικτύου (incremental loading) με τη μέθοδο όλα ή τίποτα που και αυτή όμως είναι δυνατό να δώσει αρχικές τιμές φόρτου που υπερβαίνουν τις βέλτιστες. Έτσι, στο τέλος κάθε σταδίου γίνεται διόρθωση της φόρτισης ως προς την ικανότητα των συνδέσμων, μία διαδικασία που είναι γνωστή ως καταμερισμός με έλεγχο ικανότητας των συνδέσμων (capacity restraint).

Σύμφωνα με τα παραπάνω προτείνεται ένας αλγόριθμος για την εξισορρόπηση των δικτύων ως προς το χρήστη εάν η φόρτισή τους είναι γνωστή :

ΑΛΓΟΡΙΘΜΟΣ : User's Equilibrium – incremental loading

Μεταφορικό δίκτυο G={N,E}
Διάβασε από ένα αρχείο τη ζήτηση για μετακινήσεις στο δίκτυο μεταξύ κάθε ζεύγους κόμβων προέλευσης-προορισμού i, j : Demand(i,j) και διαίρεσε τη ζήτηση αυτή $\boldsymbol{Z}$ φορές ώστε η ζήτηση για κάθε μετακίνηση να είναι μικρή (φόρτοι σε δεκάδες ή εκατοντάδες). Αυτό συμβαίνει ώστε να καταμεριστεί η ζήτηση σταδιακά.
Διάβασε από ένα αρχείο τους χρόνους διάνυσης των συνδέσμων σε ελεύθερη ροή ή υπό τις υπάρχουσες συνθήκες, t(e), ∀ e∈{1,E} και την κυκλοφοριακή ικανότητα των συνδέσμων q_{cap}(e) , ∀ e∈{1,E}. Οι τρέχοντες χρόνοι διάνυσης των συνδέσμων είναι t'(e) ← t(e), ∀ e∈{1,E}
Για Κάθε z = 1, $\boldsymbol{Z}$
Για Κάθε r = 1,N // r : Κόμβος Προέλευσης
Χρησιμοποίησε έναν αλγόριθμο εύρεσης της συντομότερης διαδρομής από τον

κόμβο r προς όλους τους υπόλοιπους κόμβους του δικτύου με χρόνο διάνυσης συνδέσμων : t'(e).
Για Κάθε j=1,N
Εάν j ≠ r
Χρησιμοποίησε έναν αλγόριθμο εύρεσης των συνδέσμων e ϵ p, από τους οποίους αποτελείται η συντομότερη διαδρομή r, j
Για Κάθε e ϵ p
Flow(e) ← Flow(e) + Demand(r, j) / $\boldsymbol{Z}$ // όπου Flow(e) η φόρτιση του // συνδέσμου e
Τέλος Για Κάθε
Τέλος Εάν
Τέλος Για Κάθε
Τέλος Για Κάθε
Για Κάθε e = 1,E
t'(e) ← t(e)× [1+ α (Flow(e) / $(q_{cap}(e))^{\beta}$] // χρήση της συνάρτησης BPR
Τέλος Για Κάθε
Τέλος Για Κάθε

Υπάρχουν και άλλοι αλγόριθμοι που έχουν χρησιμοποιηθεί για τον καταμερισμό της ζήτησης με βάση την ισορροπία του χρήστη καθώς το πρόβλημα αυτό αποτελεί ένα από τα βασικότερα προβλήματα για την ισορροπία των δικτύων. Στην εργασία των LeBlanc et al. (1975) προτάθηκε μία μέθοδος που βασίζεται στον αλγόριθμο Frank and Wolfe (1956) αλλά οδηγεί στην ισορροπία του χρήστη σε στατικά δίκτυα και όχι σε δυναμικά. Επίσης, πάλι σε εργασία των LeBlanc et al. (1985) επεκτάθηκε η προηγούμενη μέθοδος ώστε να συγκλίνει ταχύτερα. Σε δυναμικά δίκτυα εφαρμόστηκε από τους Ran and Boyce (1994).

7.5.4.ΚΑΤΑΜΕΡΙΣΜΟΣ ΤΗΣ ΖΗΤΗΣΗΣ ΜΕ ΣΤΟΧΟ ΤΗΝ ΙΣΟΡΡΟΠΙΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Η κατάσταση ισορροπίας του συστήματος απαιτεί τη συνεργασία των χρηστών με τον κεντρικό διαχειριστή ακόμα και αν τους προτείνονται μη αποδοτικές διαδρομές, ώστε να μειωθεί στο ελάχιστο το συνολικό κόστος μετακίνησης. Σύμφωνα με τη δεύτερη αρχή του Wardrop αυτή η κατάσταση ισορροπίας εκφράζεται μαθηματικά ως :

Υπό τις προϋποθέσεις :

$\sum_k f_k^{rs} = q^{rs}$, δηλαδή ο συνολική φόρτιση μεταξύ δύο κόμβων προέλευσης προορισμού r→s που συμβολίζεται ως q^{rs} πρέπει να καταμερίζεται σε όλες τις διαδρομές k που έχουν κόμβο προέλευσης τον r και κόμβο προορισμού τον s : f_k^{rs}

$f_k^{rs} \geq 0$, δηλαδή κάθε διαδρομή k μεταξύ δύο κόμβων r, s μπορεί να παραλαμβάνει κυκλοφοριακό φόρτο ή να μη φορτίζεται.

$q_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_k^{rs}$, ∀α. Αυτός ο περιορισμός υπονοεί ότι η φόρτιση κάθε συνδέσμου α ισούται με το άθροισμα της φόρτισης που παραλαμβάνουν όλες οι διαδρομές k με κόμβο προέλευσης τον τυχαίο κόμβο r και κόμβο προορισμού τον τυχαίο κόμβο s για τις οποίες ισχύει ότι ο σύνδεσμος α είναι μέλος της διαδρομής, κάτι που συνεπάγεται ότι $\delta_k^{rs} = 1$, αλλιώς $\delta_k^{rs} = 0$.

Απαιτείται : Αν $t_a(q_a)$ το κόστος του συνδέσμου α συναρτήσει του φόρτου του συνδέσμου, $q_a \in [0,+\infty)$, τότε : Minimize z(x) = $\sum_a q_a t_a(q_a)$

Η βέλτιστη λύση επιτυγχάνεται όταν τα οριακά κόστη για όλες τις διαδρομές που δέχονται φόρτιση είναι ίσα :

$f_k^{rs}(t_k^{rs} - t^{rs*}) = 0$, $t_k^{rs} - t^{rs*} \geq 0$, όπου t_k^{rs} το οριακό κόστος της διαδρομής k και t^{rs*} το βέλτιστο οριακό κόστος για τη μετακίνηση r→s.

Αναλύοντας την παραπάνω μαθηματική έκφραση εάν ένα μεταφορικό μέσο πρέπει να επιλέξει μία διαδρομή k για τη μετακίνηση r→s, τότε θα επιλέξει τη διαδρομή αυτή που έχει το μικρότερο οριακό κόστος. Αυτό σημαίνει ότι το κόστος που αντιστοιχεί στη διέλευση ενός οχήματος από αυτήν τη διαδρομή είναι το μικρότερο δυνατό. Κάτι τέτοιο όμως εξαρτάται από τα οριακά κόστη των συνδέσμων που συγκροτούν αυτή τη διαδρομή. Κάθε σύνδεσμος α έχει οριακό κόστος :

$MC_a = q_a t_a(q_a) \frac{dt_a}{dq_a} = t_a(q_a) + q_a\, t'_\alpha(q_a)$

Εάν θεωρηθεί ότι ο χρόνος διάνυσης του συνδέσμου α συνδέεται με το φόρτο μέσω της σχέσης BPR, τότε $t_a(q_a) = t_a^0$ [1+ 0,15 $(q_a / q_{a,cap})^4$], έτσι

$$q_a t_a(q_a) \frac{dt_a}{dq_a} = t_a^0 \left[1+ 0{,}15 \left(q_a / q_{a,cap}\right)^4\right] + q_a \left[(1+0{,}15 \left(q_a / q_{a,cap}\right)^4) + t_a^0 \frac{0.6 q_\alpha^3}{q_{a,cap}^4}\right]$$

, όπου t_a^0 ο χρόνος διάνυσης του συνδέσμου α σε συνθήκες ελεύθερης ροής.

Η διαδρομή k θα είναι η διαδρομή για την οποία

$\sum_\alpha q_a t_a(q_a) \frac{dt_a}{dq_a}$ = Minimum, α∈k

Σημειώνεται ότι στο πρόβλημα της ισορροπίας ως προς το χρήστη η διαδρομή k ήταν η : $\sum_\alpha t_a(q_a)$ = Minimum, α∈k

Έτσι το παραπάνω πρόβλημα επιλύεται μέσω ενός αλγορίθμου, παρόμοιου με αυτού που χρησιμοποιήθηκε για την εύρεση της ισορροπίας του χρήστη. Η διαφορά έγκειται στο ότι ο αλγόριθμος αυτός θεωρεί κόστος συνδέσμου α = $q_a t_a(q_a) \frac{dt_a}{dq_a}$

Επίσης είναι χρήσιμη η παρατήρηση ότι για μηδενικό φόρτο συνδέσμου, όταν δηλαδή ξεκινά ο καταμερισμός της ζήτησης προκύπτει $q_a t_a(q_a) \frac{dt_a}{dq_a} = t_\alpha^0$, ∀α

ΑΛΓΟΡΙΘΜΟΣ : System Equilibrium – incremental loading

Μεταφορικό δίκτυο G={N,E}

Διάβασε από ένα αρχείο τη ζήτηση για μετακινήσεις στο δίκτυο μεταξύ κάθε ζεύγους κόμβων προέλευσης-προορισμού i, j : Demand(i,j) και διαίρεσε τη ζήτηση αυτή $\boldsymbol{Z}$ φορές ώστε η ζήτηση για κάθε μετακίνηση να είναι μικρή (φόρτοι σε δεκάδες ή εκατοντάδες). Αυτό συμβαίνει ώστε να καταμεριστεί η ζήτηση σταδιακά.

Διάβασε από ένα αρχείο τους χρόνους διάνυσης των συνδέσμων σε ελεύθερη ροή ή υπό τις υπάρχουσες συνθήκες, t(e), ∀ e∈{1,E} και την κυκλοφοριακή ικανότητα των συνδέσμων q_{cap}(e) , ∀ e∈{1,E}. Οι τρέχοντες χρόνοι διάνυσης των συνδέσμων είναι t'(e) ← t(e), ∀ e∈{1,E}

Για Κάθε z = 1, $\boldsymbol{Z}$

Για Κάθε r = 1,N // r : Κόμβος Προέλευσης

Χρησιμοποίησε έναν αλγόριθμο εύρεσης της συντομότερης διαδρομής από τον κόμβο r προς όλους τους υπόλοιπους κόμβους του δικτύου με χρόνο διάνυσης συνδέσμων : t'(e).

Για Κάθε j=1,N

Εάν j ≠ r

Χρησιμοποίησε έναν αλγόριθμο εύρεσης των συνδέσμων e ϵ p, από τους οποίους αποτελείται η συντομότερη διαδρομή r, j

Για Κάθε e ϵ p

Flow(e) ← Flow(e) + Demand(r, j) / $\boldsymbol{Z}$ // όπου Flow(e) η φόρτιση του // συνδέσμου e

Τέλος Για Κάθε

Τέλος Εάν

Τέλος Για Κάθε

Τέλος Για Κάθε

Για Κάθε e = 1,E

Ο νέος χρόνος διάνυσης t'(e) θεωρείται ότι ισούται με : $q_a t_a(q_a) \frac{dt_a}{dq_a}$, δηλαδή

t'(e) ← t(e) × [1+ 0.15 (Flow(e) / $(q_{cap}(e))^4$] +

Flow(e) × [1+ 0.15 (Flow(e) / $(q_{cap}(e))^4$ + t(e) $\frac{0.6 \times \text{Flow(e)}^3}{(q_{cap}(e))^4}$]

Τέλος Για Κάθε

Τέλος Για Κάθε

7.5.5.ΣΥΓΚΡΙΣΗ ΤΗΣ ΙΣΟΡΡΟΠΙΑΣ ΩΣ ΠΡΟΣ ΤΟ ΧΡΗΣΤΗ ΚΑΙ ΤΗΣ ΙΣΟΡΡΟΠΙΑΣ ΩΣ ΠΡΟΣ ΤΟ ΣΥΣΤΗΜΑ - ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΜΙΑ ΝΕΑ ΠΟΛΥΚΡΙΤΗΡΙΑΚΗ ΜΟΡΦΗ ΔΡΟΜΟΛΟΓΗΣΗΣ

Σύμφωνα με τις μεθόδους δρομολόγησης που έχουν προταθεί ο καταμερισμός της ζήτησης σε δυναμικά δίκτυα μέσω των συστημάτων πλοήγησης μπορεί να προσεγγίσει την ισορροπία ως προς το χρήστη. Η ισορροπία του χρήστη, ωστόσο, ικανοποιεί τους χρήστες αλλά δεν ελαχιστοποιεί το συνολικό χρόνο διάνυσης, που ορίζεται ως το άθροισμα των επιμέρους χρόνων διάνυσης. Αυτό είναι εμφανές από τον ορισμό αυτής της ισορροπίας. Οι Roughgarden and Tardos (2001) έκαναν πολλές εφαρμογές σε αυτό το θέμα και κατέληξαν στο ότι η ισορροπία ως προς το χρήστη που επιχειρείται από τα συστήματα πλοήγησης μπορεί να δώσει συνολικό χρόνο διάνυσης πολύ μεγαλύτερο από την περίπτωση που η δρομολόγηση για κάθε χρήστη βασιζόταν στην ισορροπία του συστήματος.

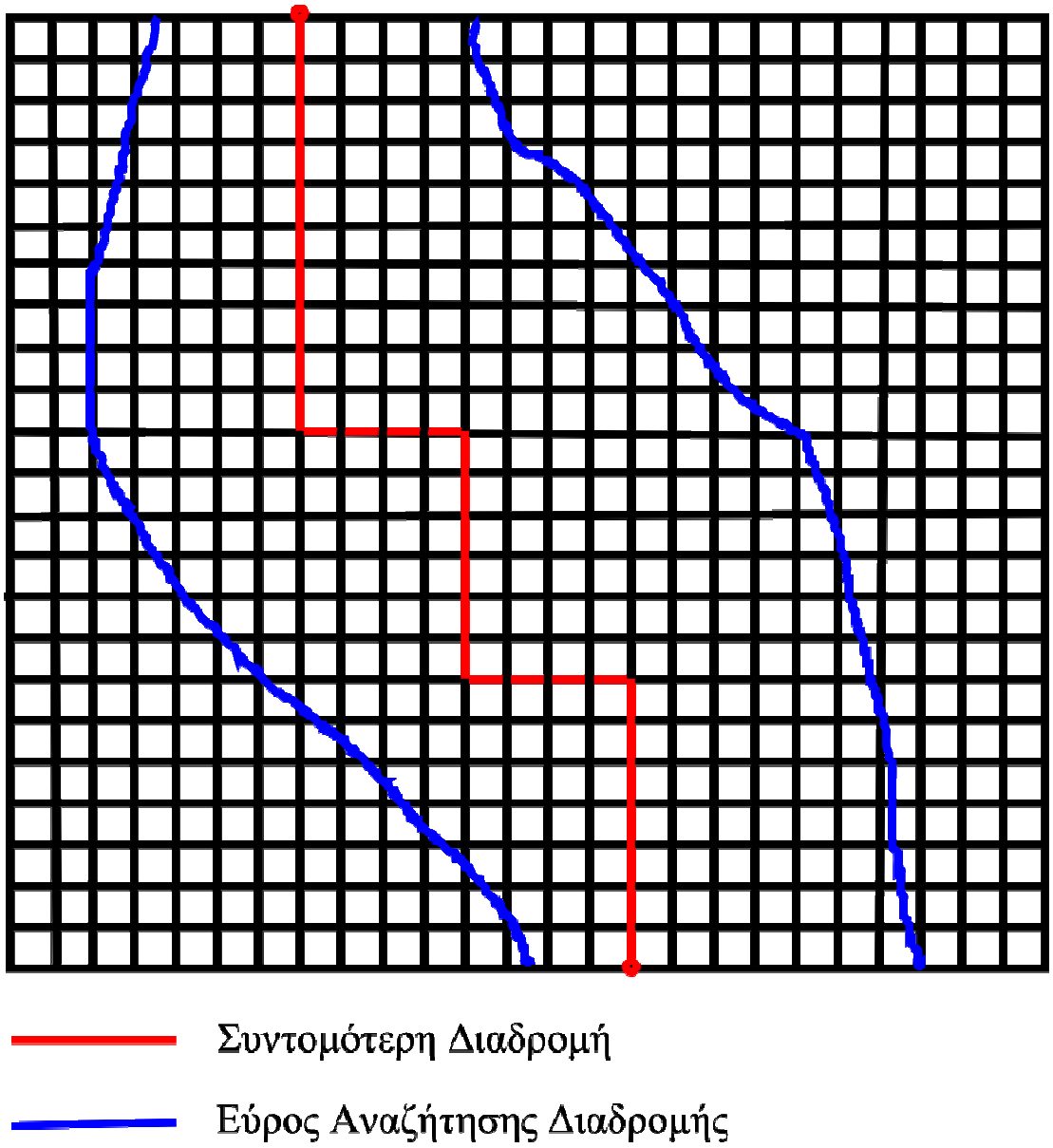
Άλλο ένα μειονέκτημα της ισορροπία ως προς το χρήστη είναι η έλλειψη μονοτονικότητας σε σχέση με την κυκλοφοριακή συμφόρηση, όπως παρατηρήθηκε πρώτη φορά από τον Braess (1968). Αυτό σημαίνει ότι ακόμα κι αν προστεθεί ένας σύνδεσμος στο συγκοινωνιακό δίκτυο, γεγονός που οδηγεί στην αύξηση της χωρητικότητάς του, μπορεί να μην υπάρχει καμία μεταβολή στο συνολικό χρόνο διάνυσης κατά τη νέα ισορροπία ως προς το χρήστη που θα προκύψει λόγω αυτής της μεταβολής.

Αρχικά λοιπόν φαίνεται ότι η ισορροπία του συστήματος είναι η βέλτιστη λύση και πρέπει να επιδιώκεται η δρομολόγηση μέσω των συστημάτων πλοήγησης να οδηγεί σε αυτήν. Ωστόσο τα συστήματα πλοήγησης σχεδιάζουν διαδρομές για τους χρήστες και αν ακολουθηθεί αυτή η πρακτική μπορεί να δημιουργηθούν διάφορα προβλήματα όπως :

Μία διαδρομή που προτείνεται στο χρήστη και έχει στόχο τον καταμερισμό της ζήτησης ως προς την ισορροπία του συστήματος μπορεί να έχει μεγαλύτερο κόστος σε σχέση με το αν υπολογιζόταν με βάση την ισορροπία του χρήστη : Roughgarden (2004). Αυτό το σημείο είναι κρίσιμο καθώς πρέπει να ληφθεί υπόψη ότι οι διαδρομές σχεδιάζονται για το χρήστη και εάν δεν είναι αποδοτικές μπορεί να μην προτιμηθούν. Δεν είναι υπερβολικό να θεωρηθεί ότι θα είναι πολλοί λίγοι οι χρήστες που θα θυσιάσουν τη βέλτιστη διαδρομή τους για το κοινό συμφέρον.

Για το λόγο αυτό πρέπει να αναπτυχθεί μία στρατηγική δρομολόγησης που θα ικανοποιεί και τα δύο παραπάνω κριτήρια. Η βέλτιστη λύση είναι η δρομολόγηση αυτή να οδηγεί στην εξισορρόπηση του συστήματος υπό περιορισμούς και όχι στην ισορροπία ως προς το χρήστη. Μία έρευνα προς αυτή την κατεύθυνση πραγματοποιήθηκε από τους Schulz and Stier-Moses (2003) οι οποίοι πρότειναν ένα θεωρητικό τύπο δρομολόγησης που οδηγεί στην ισορροπία του συστήματος προτείνοντας εφικτές διαδρομές προς τους χρήστες. Αυτό επιτυγχάνεται μέσω κάποιων περιορισμών. Για παράδειγμα η προτεινόμενη διαδρομή δε μπορεί να απέχει

περισσότερο από έναν αριθμό χιλιομέτρων σε σχέση με τη συντομότερη διαδρομή. Για το λόγο αυτό η νέα ισορροπία που προκύπτει καλείται και ισορροπία του συστήματος υπό περιορισμούς. Ένας από τους πρώτους αλγόριθμους σε αυτή την κατεύθυνση παρουσιάστηκε από τους Jahn et al. (1999).



Σχήμα 7.7.Πεδίο αναζήτησης διαδρομής με στόχο την ισορροπία του συστήματος υπό περιορισμούς

ΚΕΦΑΛΑΙΟ 8^{ο}

ΕΦΑΡΜΟΓΕΣ ΣΕ ΣΥΓΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

8.1.ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό πραγματοποιούνται εφαρμογές αλγορίθμων που έχουν προταθεί ήδη σε προηγούμενες ενότητες της εργασίας. Σκοπός του κεφαλαίου αυτού είναι η συγκριτική αξιολόγηση των διαθέσιμων μεθόδων επίλυσης προβλημάτων σε δίκτυα. Όπως έχει αναφερθεί ήδη τα συστήματα πλοήγησης έχουν περιορισμένες υπολογιστικές δυνατότητες και κατά τον υπολογισμό μίας διαδρομής χρησιμοποιούν τον αλγόριθμο Dijkstra με ορισμένες διαφοροποιήσεις. Εξαιτίας αυτού του περιορισμού μέχρι σήμερα δεν έχουν επεκταθεί στην επίλυση πιο σύνθετων προβλημάτων. Οι εφαρμογές που ακολουθούν στη συνέχεια αποσκοπούν στην αποσαφήνιση ζητημάτων όπως :

α) Αξιολόγηση μεθόδων υπολογισμού της βέλτιστης διαδρομής από έναν κόμβο του συγκοινωνιακού δικτύου προς όλους τους υπόλοιπους κόμβους.

β) Αξιολόγηση μεθόδων υπολογισμού της βέλτιστης διαδρομής μεταξύ δύο κόμβων προέλευσης-προορισμού του δικτύου.

γ) Αξιολόγηση μεθόδων υπολογισμού της βέλτιστης διαδρομής από όλους τους κόμβους προς όλους τους κόμβους του δικτύου.

δ) Αξιολόγηση μεθόδων υπολογισμού εναλλακτικών διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού και δυνατότητες εφαρμογής σε συστήματα πλοήγησης.

ε) Αξιολόγηση μεθόδων υπολογισμού της βέλτιστης διαδρομής σε δίκτυα με περιορισμούς διαθέσιμων πόρων. Διερεύνηση της δυνατότητας εφαρμογής αυτών των μεθόδων σε συστήματα πλοήγησης ώστε να υπολογίζονται διαδρομές σύμφωνα με τις εξατομικευμένες ανάγκες κάθε χρήστη και τον περιορισμό των εκπομπών CO_2.

στ) Διερεύνηση της απόδοσης των συγκοινωνιακών δικτύων στην ιδεατή κατάσταση στην οποία όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης. Η διερεύνηση εξετάζει τις περιπτώσεις :

i) Καταμερισμός της ζήτησης σε στατικά δίκτυα.
ii) Καταμερισμός της ζήτησης σε δυναμικά δίκτυα όταν το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης δεν επηρεάζει τη διαμορφωμένη κατάσταση.
iii) Καταμερισμός της ζήτησης σε δυναμικά δίκτυα με ελλιπή πληροφόρηση των συστημάτων πλοήγησης σχετικά με τις κυκλοφοριακές συνθήκες. Αυτή η περίπτωση αποτελεί τη ρεαλιστική κατάσταση που επικρατεί σήμερα στον Ελλαδικό χώρο.

iv) Καταμερισμός της ζήτησης σε δυναμικά δίκτυα με πλήρη πληροφόρηση των συστημάτων πλοήγησης σχετικά με τις συνθήκες που επικρατούν στο δίκτυο και χρήση της κεντρικής δρομολόγησης με στόχο την ισορροπία του χρήστη.

v) Καταμερισμός της ζήτησης σε δυναμικά δίκτυα με πλήρη πληροφόρηση των συστημάτων πλοήγησης σχετικά με τις συνθήκες που επικρατούν στο δίκτυο και χρήση της κεντρικής δρομολόγησης με στόχο την ισορροπία του συστήματος.

Για να υπάρξει όσο το δυνατόν μεγαλύτερη αντικειμενικότητα στη συγκριτική αξιολόγηση των παραπάνω μεθόδων όλοι οι αλγόριθμοι προγραμματίστηκαν σε κοινή γλώσσα προγραμματισμού, Fortran 95. Η εφαρμογή τους υλοποιήθηκε σε ηλεκτρονικό υπολογιστή με επεξεργαστή Intel Core 2 Duo ταχύτητας 2.66 GHz.

Κάθε επιμέρους αξιολόγηση αποτελείται από τέσσερα διακριτά σκέλη :

i) Παρουσίαση αποτελέσματος εφαρμογής των μεθόδων σε συγκοινωνιακό δίκτυο υπό τη μορφή εικόνας.

ii) Πίνακας με τον χρόνο εκτέλεσης και τις απαιτούμενες θέσεις μνήμης των μεθόδων που συγκρίνονται κατά την εφαρμογή τους σε διάφορα συγκοινωνιακά δίκτυα.

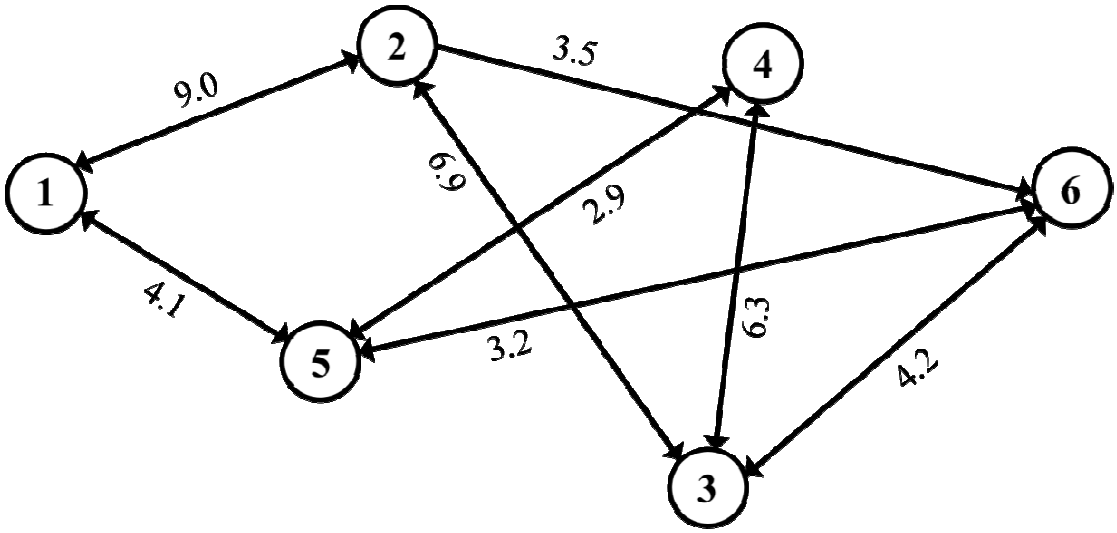
iii) Διαγράμματα χρόνων εκτέλεσης των μεθόδων που συγκρίνονται.

iv) Συμπεράσματα και σχολιασμός.

Όλοι οι αλγόριθμοι που χρησιμοποιήθηκαν είναι προγραμματισμένοι και βρίσκονται στην Ενότητα του Παραρτήματος σύμφωνα με τη σειρά που παρουσιάζονται στις εφαρμογές. Τα δίκτυα που χρησιμοποιούνται αποτελούνται από πλήθος κόμβων και συνδέσμων και είναι αποθηκευμένα με τη μορφή λίστας γειτνίασης σε αρχεία από τα οποία ενημερώνονται τα προγράμματα. Για παράδειγμα, ενδεικτικά ένα δικτύου 6 κόμβων και 16 συνδέσμων βρίσκεται αποθηκευμένο σε ένα αρχείο ως :

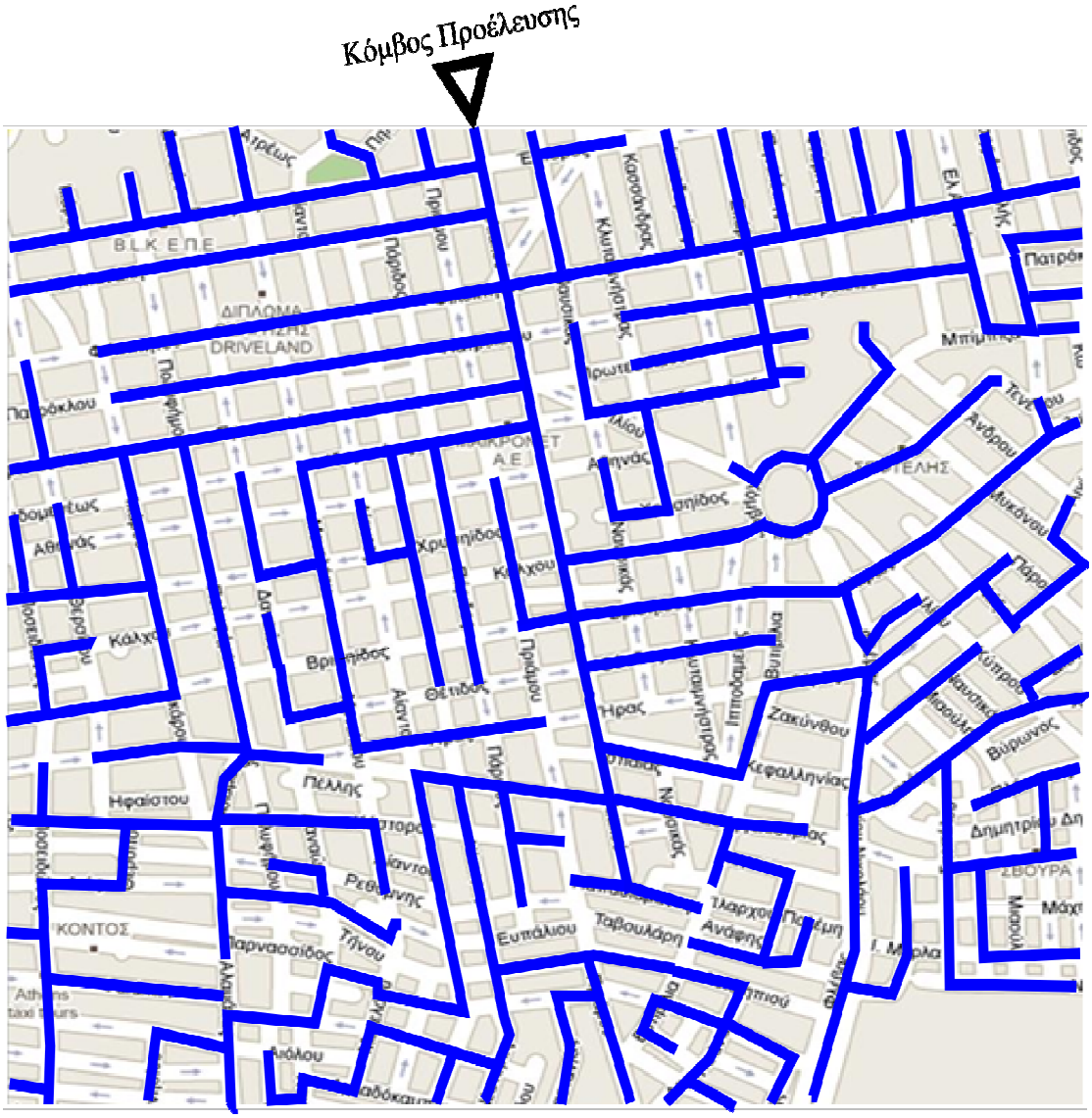
Αριθμός Κόμβων = 6
Αριθμός Συνδέσμων = 16
Κόμβος Προέλευσης = 1, Κόμβος Προορισμού = 6

Κόμβων Προέλευσης	Κόμβος Προορισμού	Αριθμός Συνδέσμου	Βάρος Συνδέσμου
1	2	1	9.
1	5	2	4.1
2	1	3	9.
2	3	4	6.9
2	6	5	3.5
3	4	6	6.3
3	6	7	4.2
3	2	8	6.9
4	5	9	2.9
4	3	10	6.3
5	1	11	4.1
5	4	12	2.9
5	6	13	3.2
6	2	14	3.5
6	3	15	4.2
6	5	16	3.2



Σχήμα 8.1.Πραγματική μορφή συγκοινωνιακού δικτύου

8.2.ΑΞΙΟΛΟΓΗΣΗ ΜΕΘΟΔΩΝ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΑΠΟ ΕΝΑΝ ΚΟΜΒΟ ΠΡΟΣ ΟΛΟΥΣ ΤΟΥΣ ΥΠΟΛΟΙΠΟΥΣ ΚΟΜΒΟΥΣ ΤΟΥ ΔΙΚΤΥΟΥ

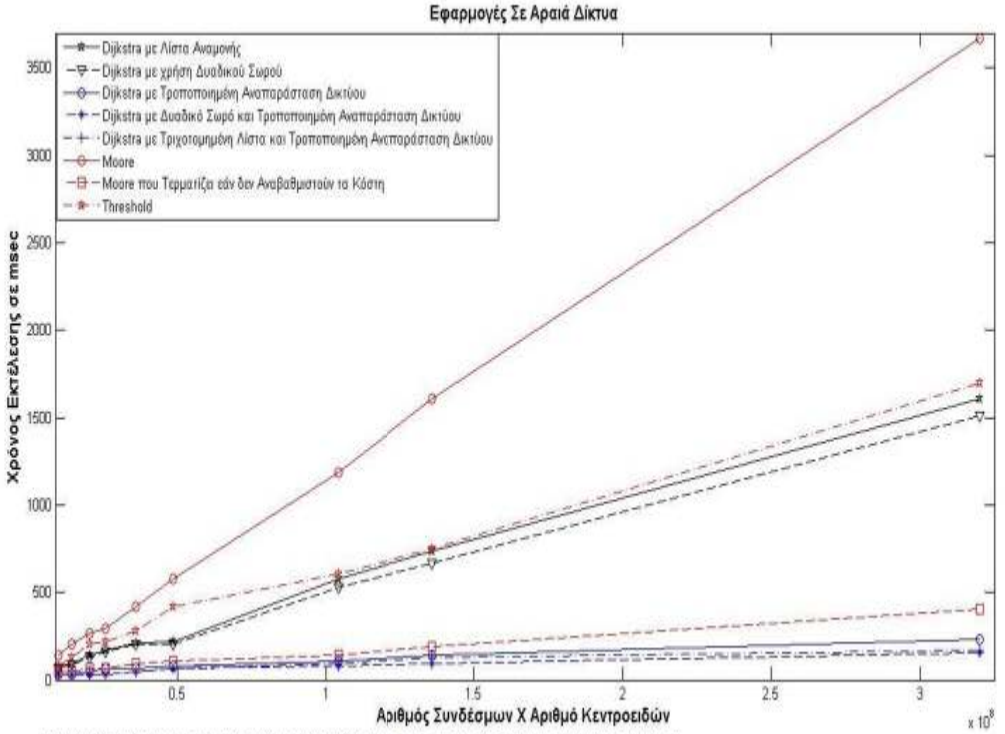


Εικόνα 8.1.Δένδρα Ελάχιστων διαδρομών από έναν κόμβο προέλευσης προς όλους τους υπόλοιπους κόμβους του δικτύου

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης
Δίκτυα με συνηθισμένη αναλογία κόμβων / συνδέσμων
Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10
Χρόνος Εκτέλεσης αλγορίθμου σε **msec** :

*Αριθμός Επαναλήψεων Αλγορίθμων	ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ					ΜΕΘΟΔΟΙ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ		
	DIJKSTRA ΜΕ ΛΙΣΤΑ ΑΝΑΜΟΝΗΣ	DIJKSTRA ΜΕ ΧΡΗΣΗ ΔΥΑΔΙΚΟΥ ΣΩΡΟΥ	DIJKSTRA ΜΕ ΤΡΟΠΟΠΟΙΗΜΕΝΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΙΚΤΥΟΥ, ΕΝΟΤΗΤΑ 5.8.	DIJKSTRA ΜΕ ΧΡΗΣΗ ΔΥΑΔΙΚΟΥ ΣΩΡΟΥ ΣΕ ΤΡΟΠΟΠΟΙΗΜΕΝΟ ΔΙΚΤΥΟ	DIJKSTRA ΜΕ ΤΡΙΧΟΤΟΜΗΜΕΝΗ ΛΙΣΤΑ ΑΝΑΜΟΝΗΣ ΣΕ ΤΡΟΠΟΠΟΙΗΜΕΝΟ ΔΙΚΤΥΟ	MOORE - BELLMAN, FORD	MOORE ΠΟΥ ΤΕΡΜΑΤΙΖΕΙ ΕΑΝ ΔΕΝ ΑΝΑΒΑΘΜΙΣΤΟΥΝ ΤΑ ΚΟΣΤΗ ΤΩΝ ΚΟΜΒΩΝ	THRESHOLD
ΚΕΝΤΡΟΕΙΔΗ = 1000	78,0005	63,4	31,4	32,1	32,1	140,401	46,8	78,02
ΣΥΝΔΕΣΜΟΙ = 10010	32030 / 998*	34030	43040	45040	48040	31030	31030 / 70*	35030
ΚΕΝΤΡΟΕΙΔΗ = 1200	93,6	89,7	31,7	32,1	32,1	202,801	47,4	124,8
ΣΥΝΔΕΣΜΟΙ = 12010	38430 / 1196*	40830	51640	54040	57640	37230	37230 / 87*	39630
ΚΕΝΤΡΟΕΙΔΗ = 1400	140,409	124,801	46,8	32,1	32,1	265,202	62,4	202,801
ΣΥΝΔΕΣΜΟΙ = 14610	46630 / 1399*	49430	62640	65440	69640	45230	45230 / 100*	48030
ΚΕΝΤΡΟΕΙΔΗ = 1600	171,6	161,6	62,4	32,1	32,1	296,402	64,1	218,401
ΣΥΝΔΕΣΜΟΙ = 16210	51830 / 1595*	55030	69640	72840	77640	50230	50230 / 113*	53430
ΚΕΝΤΡΟΕΙΔΗ = 1800	209,0015	202,801	62,4	46,8	46,8	421,203	93,6	280,802
ΣΥΝΔΕΣΜΟΙ = 20000	63600 / 1799*	67200	85400	89000	94400	61800	61800 / 131*	65400
ΚΕΝΤΡΟΕΙΔΗ = 2000	218,402	204,201	78,1	62,4	62,4	577,204	109,2	416,803
ΣΥΝΔΕΣΜΟΙ = 24290	76870 / 1877*	80870	103160	107160	113160	74870	74870 / 130*	78870
ΚΕΝΤΡΟΕΙΔΗ = 2600	577,203	530,404	109,2	78,1	93,6	1185,61	140,401	608,403
ΣΥΝΔΕΣΜΟΙ = 40000	125200 / 2600*	130400	167800	173000	180800	122600	122600 / 188*	127800
ΚΕΝΤΡΟΕΙΔΗ = 2800	733,204	696,804	140,4	93,6	124,8	1606,81	187,201	748,804
ΣΥΝΔΕΣΜΟΙ = 48580	151340 / 2800*	156940	202720	208320	216720	148540	148540 / 213*	154140
ΚΕΝΤΡΟΕΙΔΗ = 4000	1606,81	1513,209	234	156,1	171,601	3666,2	405,6	1700,1
ΣΥΝΔΕΣΜΟΙ = 80000	24800 / 4000*	32800	332000	340000	352000	244000	244000 / 320*	252000

Πίνακας 8.1.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των βέλτιστων διαδρομών από ένα σημείο προέλευσης προς όλα τα σημεία προορισμού με χρήση αλγορίθμων σε αραιά δίκτυα

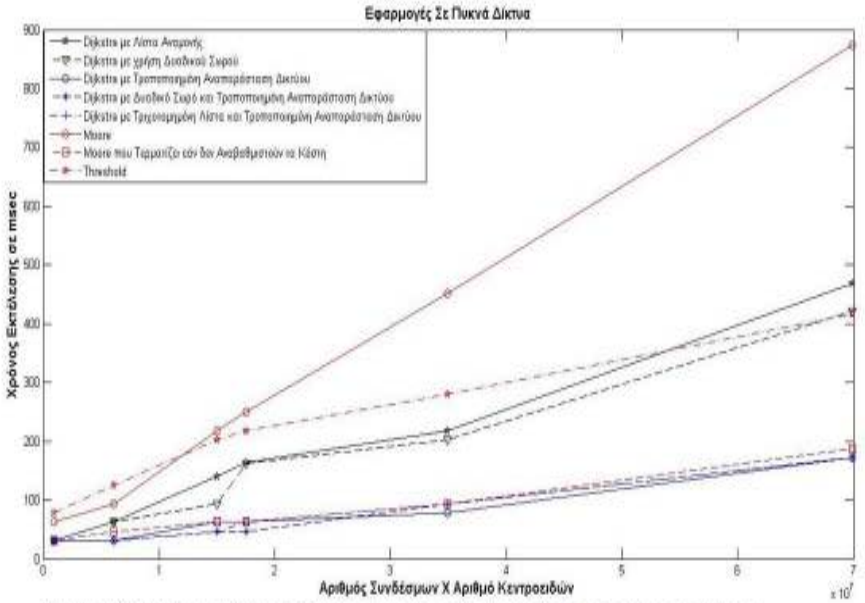


Διάγραμμα 8.1.Χρόνοι εύρεσης των βέλτιστων διαδρομών από ένα σημείο προέλευσης προς όλα τα σημεία προορισμού σε αραιά δίκτυα

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης
Πυκνά δίκτυα με ιδεατή αναλογία κόμβων / συνδέσμων
Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.010
Χρόνος Εκτέλεσης αλγορίθμου σε **msec** :

*Αριθμός Επαναλήψεων Αλγορίθμων	ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ					ΜΕΘΟΔΟΙ ΔΥΝΑΜΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ		
	DIJKSTRA ΜΕ ΛΙΣΤΑ ΑΝΑΜΟΝΗΣ	DIJKSTRA ΜΕ ΧΡΗΣΗ ΔΥΑΔΙΚΟΥ ΣΩΡΟΥ	DIJKSTRA ΜΕ ΤΡΟΠΟΠΟΙΗΜΕΝΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΙΚΤΥΟΥ, ΕΝΟΤΗΤΑ 5.8.	DIJKSTRA ΜΕ ΧΡΗΣΗ ΔΥΑΔΙΚΟΥ ΣΩΡΟΥ ΣΕ ΤΡΟΠΟΠΟΙΗΜΕΝΟ ΔΙΚΤΥΟ	DIJKSTRA ΜΕ ΤΡΙΧΟΤΟΜΗΜΕΝΗ ΛΙΣΤΑ ΑΝΑΜΟΝΗΣ ΣΕ ΤΡΟΠΟΠΟΙΗΜΕΝΟ ΔΙΚΤΥΟ	MOORE - BELLMAN, FORD	MOORE ΠΟΥ ΤΕΡΜΑΤΙΖΕΙ ΕΑΝ ΔΕΝ ΑΝΑΒΑΘΜΙΣΟΥΝΤΑ ΚΟΣΤΗ ΤΩΝ ΚΟΜΒΩΝ	THRESHOLD
ΚΕΝΤΡΟΕΙΔΗ = 100	31,2	31,2	31,2	31,2	31,2	62,4	31,2	78,02
ΣΥΝΔΕΣΜΟΙ = 9600	29000 / 98*	29200	38700	38900	39200	28900	28900 / 6*	29100
ΚΕΝΤΡΟΕΙΔΗ = 300	62,4	62,4	31,2	31,2	31,2	93,600	46,8	124,8
ΣΥΝΔΕΣΜΟΙ = 20400	61800 / 299*	62400	82500	83100	84000	61500	61500 / 5*	62100
ΚΕΝΤΡΟΕΙΔΗ = 500	140,409	93,6	62,4	46,8	46,8	218,401	62,4	202,801
ΣΥΝΔΕΣΜΟΙ = 30035	91105 / 500*	92105	121640	123140	124640	90605	90605 / 7*	91605
ΚΕΝΤΡΟΕΙΔΗ = 500	164,2	161,6	62,4	46,8	62,4	249,600	63.6	218,401
ΣΥΝΔΕΣΜΟΙ = 35066	106198 / 1595*	107198	141764	143264	144764	105698	105698 / 7*	106698
ΚΕΝΤΡΟΕΙΔΗ = 700	218,4	202,801	78,005	93,6	93,6	452,403	93,6	280,802
ΣΥΝΔΕΣΜΟΙ = 49932	151196 / 700*	152596	201828	203928	206028	150496	105496 / 10*	106896
ΚΕΝΤΡΟΕΙΔΗ = 700	468,003	421,203	171,6	171,6	171,6	873,605	187,201	416,803
ΣΥΝΔΕΣΜΟΙ = 99864	300992 / 700*	302392	401556	402956	405056	300292	300292 / 6*	301692

Πίνακας 8.2.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των βέλτιστων διαδρομών από ένα σημείο προέλευσης προς όλα τα σημεία προορισμού με χρήση αλγορίθμων σε πυκνά δίκτυα



Διάγραμμα 8.2.Χρόνοι εύρεσης των βέλτιστων διαδρομών από ένα σημείο προέλευσης προς όλα τα σημεία προορισμού σε πυκνά δίκτυα

Συμπεράσματα

Όλες οι μέθοδοι που χρησιμοποιήθηκαν έχουν παρουσιαστεί στην Ενότητα 5.2., εκτός από τη μέθοδο της τριχοτομημένης λίστας που παρουσιάστηκε στην Ενότητα 6.1. και έχουν θεωρητικούς χρόνους εκτέλεσης :

Dijkstra με Λίστα Αναμονής : $O(N^2) \rightarrow O(\mathrm{N} \times \mathrm{E})$ λόγω αναπαράστασης του δικτύου με λίστα γειτνίασης και όχι με πίνακα γειτνίασης.
Dijkstra με Χρήση Δυαδικού Σωρού : $O((\mathrm{N}+\mathrm{E})log_2 N) \rightarrow O(\mathrm{N} \times \mathrm{E})$ λόγω αναπαράστασης του δικτύου με λίστα γειτνίασης.
Dijkstra με Λίστα Αναμονής για τροποποιημένη αναπαράσταση δικτύου η οποία παρουσιάστηκε στην Ενότητα 5.8. : $O(N^2)$.
Dijkstra με Χρήση Τριχοτομημένης Λίστας και τροποποιημένη αναπαράσταση δικτύου : $O(\mathrm{N} \times \frac{N}{3})$.
Dijkstra με Χρήση Δυαδικού Σωρού και τροποποιημένη αναπαράσταση δικτύου : $O((\mathrm{N}+\mathrm{E})log_2 N)$.
Moore, Bellman-Ford : $O(\mathrm{N} \times \mathrm{E})$.
Moore που Τερματίζει εάν δεν αναβαθμιστούν τα κόστη των Κόμβων : $O(\delta \times \mathrm{E})$.
Threshold : $O(N^2 \times E)$.

Παρατηρήσεις μέσω των εφαρμογών :

1) Ο χρόνος εκτέλεσης του αλγόριθμου Bellman-Ford είναι αρκετά μειωμένος σε σύγκριση με το θεωρητικό χρόνο. Ο θεωρητικός χρόνος εκτέλεσής του στη χειρότερη περίπτωση είναι $O(\mathrm{N} \times \mathrm{E})$, αλλά είναι πολύ μικρότερος σε πρακτικά παραδείγματα. Αυτό οφείλεται στο ότι ο αλγόριθμος εκτελεί μόνο δ επαναλήψεις, όπου δ είναι η επανάληψη κατά την οποία δεν αναβαθμίζεται καμία τιμή στο δίκτυο. Σε πυκνά δίκτυα η τιμή του δ είναι πολύ μικρή, αλλά ακόμη και σε αραιά δίκτυα η τιμή του παραμένει σε ικανοποιητικά επίπεδα. Ακόμα και σε δυσμενείς περιπτώσεις, ο αριθμός των επαναλήψεων ‘δ’ είναι απρόσμενα μικρός σε σύγκριση με τον αριθμό των κόμβων του δικτύου N. Για παράδειγμα σε μία εφαρμογή εξετάστηκε ένα δίκτυο 2800 κόμβων και 48580 συνδέσμων, όπου ο αναμενόμενος χρόνος εκτέλεσης του αλγορίθμου σε αυτή την περίπτωση είναι $O(2800 \times 48580)$. Στην πραγματικότητα όμως καμία μεταβολή δε λαμβάνει χώρα στο δίκτυο έπειτα από δ = 213 επαναλήψεις. Έτσι ο πραγματικός χρόνος εκτέλεσης του αλγορίθμου είναι μόλις $O(213 \times 48580)$. Είναι λογικό λοιπόν σε αυτή την εφαρμογή να προκύπτουν χρόνοι εκτέλεσης 1606,81 msec και 187,201 msec αντίστοιχα. Το φαινόμενο αυτό παρατηρείται σε όλες τις εφαρμογές και καθιστά τον αλγόριθμο Bellman-Ford ιδιαίτερα ελκυστικό για την επίλυση αυτού του προβλήματος. Επίσης παρατηρείται ότι σε συνήθη δίκτυα η τιμή των επαναλήψεων είναι : $\delta = \{ \frac{N}{10}, \frac{N}{20} \}$.

2) Ο αλγόριθμος Dijkstra δεν είναι αποδοτικός όταν διαχειρίζεται δίκτυα τα οποία είναι αποθηκευμένα με τη μορφή Λίστας Γειτνίασης. Αυτό είναι εμφανές εάν γίνει

μία υπενθύμιση των βασικών πράξεων που απαιτούνται σε κάθε διαδικασία. Για το λόγο αυτό αναλύονται στη συνέχεια.

Αρχικά πρέπει να επιλέγουν σχεδόν όλοι οι κόμβοι του δικτύου με εκκίνηση τον κόμβο προέλευσης και προτεραιότητα ανάλογα με το κόστος τους. Για το λόγο αυτό η μέθοδος επαναλαμβάνει N φορές τις δύο ακόλουθες διαδικασίες :
α) Για Κάθε κόμβο x ϵ N γίνεται αναβάθμιση του κόστους όλων των κόμβων y ϵ N εάν : D(y) > D(x) + w(x,y). Αυτή η διαδικασία απαιτεί N επαναλήψεις εάν το δίκτυο είναι αποθηκευμένο σε μορφή πίνακα γειτνίασης, E επαναλήψεις εάν είναι αποθηκευμένο σε μορφή λίστας γειτνίασης και **φ** επαναλήψεις εάν τροποποιηθεί ώστε να υπάρχουν δείκτες μεταξύ κάθε κόμβου και των συνδέσμων που εξέρχονται από αυτόν.
β) Γίνεται επιλογή του νέου κόμβου με το μικρότερο κόστος σε σχέση με τον αρχικό κόμβο. Αυτό απαιτεί το πολύ N βασικές πράξεις όταν χρησιμοποιείται η απλή λίστα ως δομή δεδομένων ενώ σε κάθε άλλη περίπτωση απαιτούνται k < N βασικές πράξεις.

Συγκεντρωτικά οι βασικές πράξεις που απαιτούνται είναι : Βασικές Πράξεις

Για Κάθε Κόμβο u = 1,N
  Για Κάθε Κόμβο yϵN | D(y) > D(u) +w(u,y)
  D(y) ← D(u) +w(u,y) } $N \times E$ ή N^2 ή $\boldsymbol{\varphi} \times N$
  Τέλος Για Κάθε
  Εύρεση του επόμενου κόμβου vϵN | D(v) = Ελάχιστο $N \times k \leq N^2$
Τέλος Για Κάθε

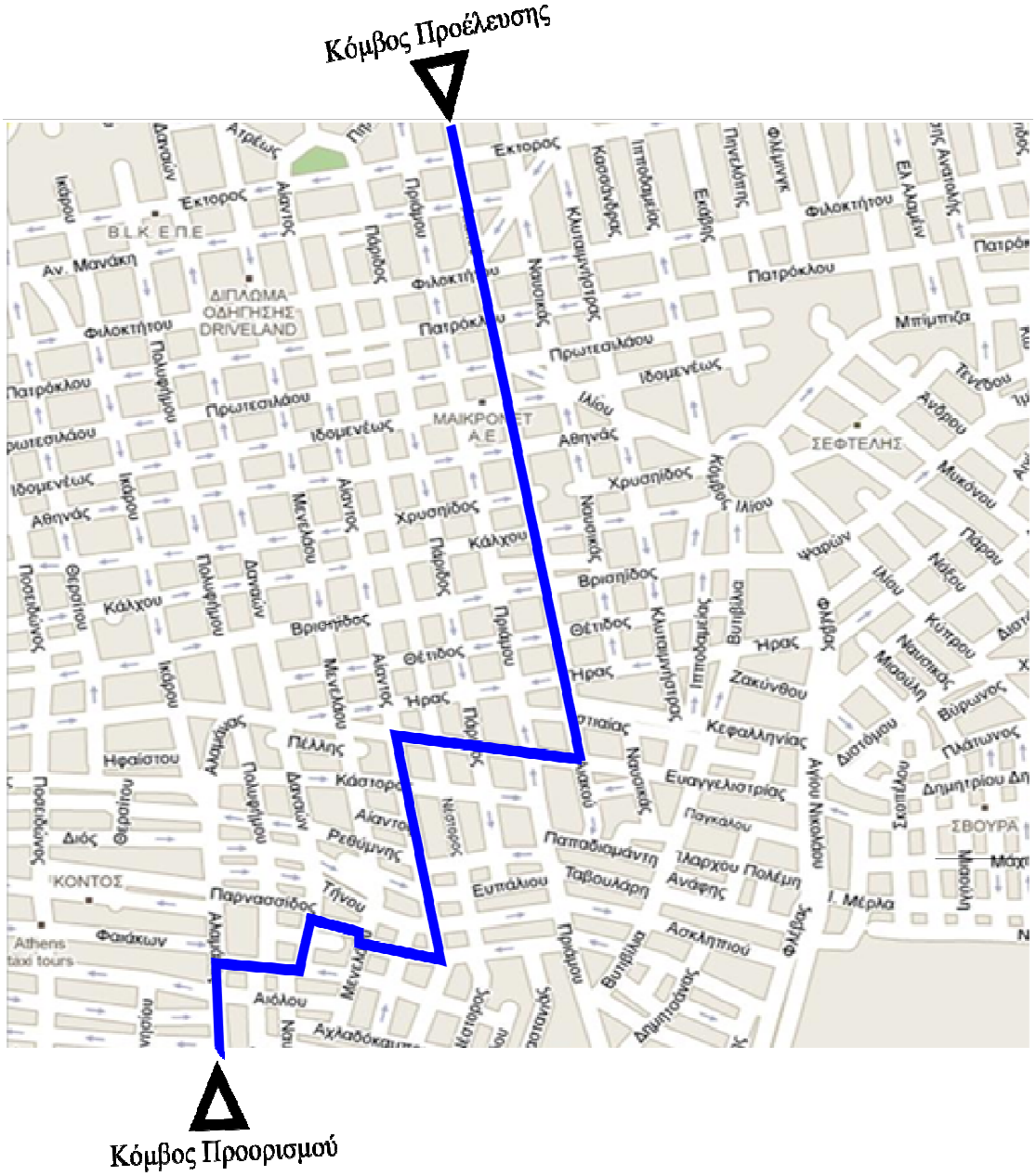
Αποδεικνύεται έτσι θεωρητικά ότι εάν το δίκτυο είναι αποθηκευμένο υπό τη μορφή Λίστας Γειτνίασης ή Πίνακα Γειτνίασης η κρίσιμη διαδικασία είναι η διαδικασία αναβάθμισης του κόστους των κόμβων, διότι $N \times E$ ή $N^2 \geqslant N \times k$. Σε αυτές τις περιπτώσεις δεν έχει σημασία ποιά δομή δεδομένων χρησιμοποιείται. Αυτό επαληθεύεται και μέσα από τις εφαρμογές, καθώς για δύο διαφορετικές δομές δεδομένων (Απλή Λίστα και Δυαδικός Σωρός) δεν υπήρχε ουσιαστική διαφορά στο χρόνο εκτέλεσής του αλγορίθμου σε δίκτυα αποθηκευμένα υπό τη μορφή λίστας γειτνίασης. Πιο συγκεκριμένα η διαφορά των χρόνων εκτέλεσης ήταν στη χειρότερη περίπτωση 47msec, κάτι που αποδεικνύει την ορθότητα της υπόθεσης. Ακόμα βέβαια και εάν το δίκτυο είναι αποθηκευμένο υπό τη μορφή πίνακα γειτνίασης, ο χρόνος εκτέλεσης του αλγορίθμου θα είναι $O(N^2)$ ανεξαρτήτως της δομής δεδομένων που χρησιμοποιεί ο αλγόριθμος. Από τα παραπάνω προκύπτει το συμπέρασμα ότι μόνο στην περίπτωση όπου υπάρχουν δείκτες μεταξύ κάθε κόμβου και των συνδέσμων που προέρχονται από αυτόν μπορεί να γίνει κρίσιμη η διαδικασία εύρεσης του επόμενου κόμβου με το μικρότερο κόστος, διότι μόνο τότε μπορεί να ισχύει $\boldsymbol{\varphi} \times N < N \times k$. Έτσι μόνο σε αυτή την περίπτωση η χρήση διαφορετικών δομών δεδομένων είναι αποδοτική. Μάλιστα ο χρόνος εκτέλεσης του αλγορίθμου είναι πλέον $O(N \times k)$ ή $O(\boldsymbol{\varphi} \times N)$, ο οποίος είναι σε κάθε περίπτωση ο μικρότερος δυνατός. Αυτό είναι εμφανές και μέσω τον εφαρμογών όπου ο χρόνος εκτέλεσης : $O(N \times k)$ είναι έως και

190% μικρότερος σε σχέση με το χρόνο εκτέλεσης σε δίκτυα που αναπαρίστανται μέσω πίνακα γειτνίασης : $O(\mathrm{N} \times \mathrm{E})$.

3) Οι διαφορετικές δομές δεδομένων για τον αλγόριθμο Dijkstra δε δίνουν ιδιαίτερα έντονες μεταβολές στους χρόνους εκτέλεσής του. Ο αριθμός των εφαρμογών βέβαια είναι μικρός για να γενικευθεί αυτό το συμπέρασμα, ωστόσο είναι εμφανές ότι είναι πολύ σημαντικότερο να γίνεται σωστή αναπαράσταση στο δίκτυο από το να βελτιώνονται οι δομές δεδομένων. Για παράδειγμα η χρήση δυαδικού σωρού σε δίκτυο αποθηκευμένο μέσω Λίστας Γειτνίασης δίνει χρόνο εκτέλεσης 1513,209 msec, ενώ η απλή λίστα σε τροποποιημένο δίκτυο δίνει χρόνο εκτέλεσης 231 msec για την ιδια εφαρμογή. Παρατηρείται δηλαδή βελτίωση της τάξης του 555%. Επίσης η χρήση δυαδικού σωρού αυτή τη φορά στο τροποποιημένο δίκτυο για την ίδια εφαρμογή δίνει χρόνο εκτέλεσης 156,1 msec που αποτελεί βελτίωση της τάξης του 50% σε σύγκριση με τη χρήση απλής λίστας. Είναι εμφανές λοιπόν ότι ο τρόπος αναπαράστασης του δικτύου είναι πολύ σημαντικότερος από τη δομή δεδομένων που θα χρησιμοποιηθεί.

4) Σε πρακτικά παραδείγματα ο αλγόριθμος Bellman-Ford είναι ιδιαίτερα ανταγωνιστικός σε σύγκριση με τον αλγόριθμο Dijkstra, ο οποίος είναι ταχύτερος μόνο κατά την περίπτωση που τροποποιείται η αναπαράσταση του δικτύου. Θεωρητικά οι χρόνοι εκτέλεσης των δύο αλγορίθμων είναι $O(\delta \times \mathrm{E})$ και $O(\mathrm{N} \times \mathrm{k})$ ή $O(\boldsymbol{\varphi} \times \mathrm{N})$ αντίστοιχα. Από τις παραπάνω εφαρμογές αποδεικνύεται ότι δεν υπάρχει πολύ μεγάλη διαφορά σε αυτούς τους χρόνους εκτέλεσης. Αυτό συμβαίνει διότι η τιμή του δ είναι μικρή και ο αλγόριθμος ολοκληρώνεται μέσα σε λίγες επαναλήψεις, όπως παρουσιάστηκε προηγουμένως. Επίσης ο αλγόριθμος Bellman-Ford απαιτεί αρκετά λιγότερες θέσεις μνήμης = $3 \times \mathrm{E} + \mathrm{N}$ σε σύγκριση με τον αλγόριθμο Dijkstra, ο οποίος απαιτεί στην καλύτερη περίπτωση, με χρήση της απλής λίστας, $4 \times \mathrm{E} + 3 \times \mathrm{N}$ θέσεις. Για το λόγο αυτό η επιλογή μεθόδου πρέπει να εξετάζεται πιο προσεκτικά ανάλογα με το δίκτυο και τις διαθέσιμες θέσεις μνήμης.

8.3.ΑΞΙΟΛΟΓΗΣΗ ΜΕΘΟΔΩΝ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΜΕΤΑΞΥ ΔΥΟ ΚΟΜΒΩΝ ΠΡΟΕΛΕΥΣΗΣ – ΠΡΟΟΡΙΣΜΟΥ



Εικόνα 8.2.Βέλτιστη διαδρομή μεταξύ δύο κόμβων προέλευσης-προορισμού του δικτύου με κόστος : $w_p = \sum_{e \epsilon p_*} w(e) = \min \{ \sum_{e \epsilon p} w(e) \}$, ∀ p

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης

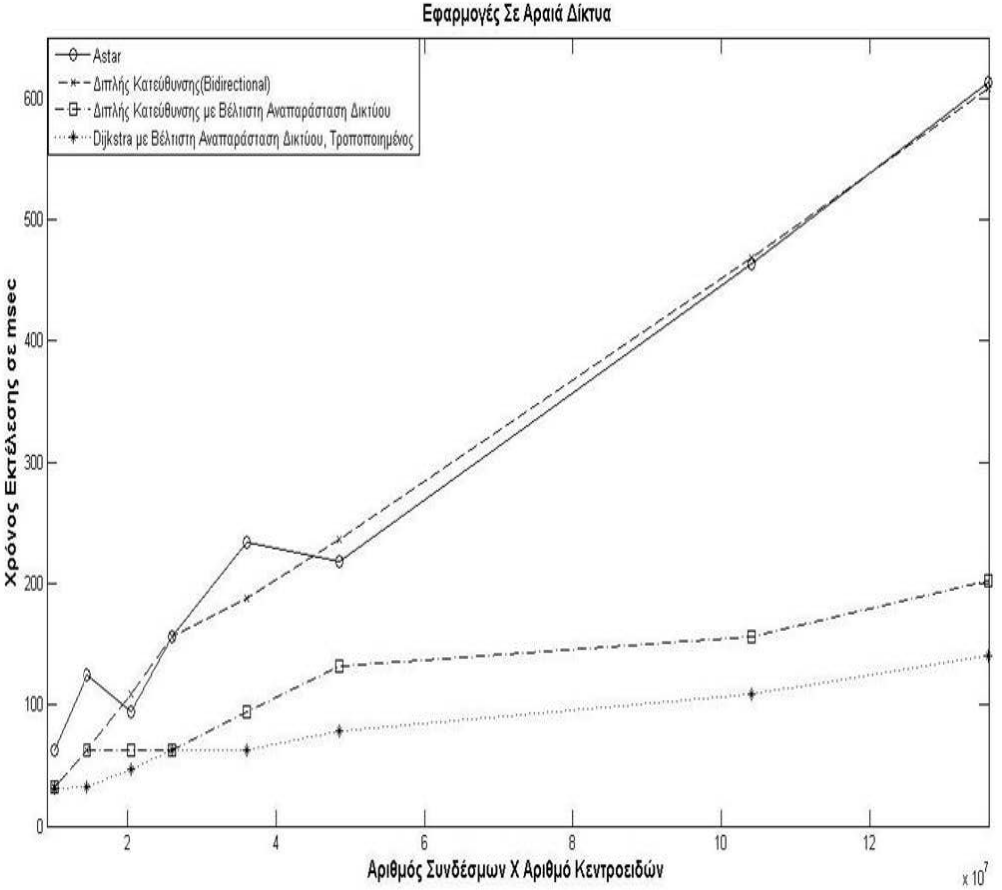
Δίκτυα με συνηθισμένη αναλογία κόμβων / συνδέσμων

Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10

Χρόνος Εκτέλεσης αλγορίθμου σε **msec** : Απαιτούμενες Θέσεις Μνήμης :

	A STAR*	ΔΙΠΛΗΣ ΚΑΤΕΥΘΥΝΣΗΣ (BIDIRECTIONAL)	ΔΙΠΛΗΣ ΚΑΤΕΥΘΥΝΣΗΣ ΜΕ ΒΕΛΤΙΣΤΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΙΚΤΥΟΥ	ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ DIJKSTRA
ΚΕΝΤΡΟΕΙΔΗ = 1000 ΣΥΝΔΕΣΜΟΙ = 10010	62,4 33030	31,9 34030	32,1 45040	31,4 43040
ΚΕΝΤΡΟΕΙΔΗ = 1200 ΣΥΝΔΕΣΜΟΙ = 12010	124,8 39630	62,4 40830	62,1 54040	31,7 51640
ΚΕΝΤΡΟΕΙΔΗ = 1400 ΣΥΝΔΕΣΜΟΙ = 14610	93,6 48030	109,207 49430	62,4 65440	46,8 62640
ΚΕΝΤΡΟΕΙΔΗ = 1600 ΣΥΝΔΕΣΜΟΙ = 16210	156,001 53430	156,1 55030	62,4 72840	62,4 69640
ΚΕΝΤΡΟΕΙΔΗ = 1800 ΣΥΝΔΕΣΜΟΙ = 20000	234,001 65400	187,201 67200	93,6 89000	62,4 85400
ΚΕΝΤΡΟΕΙΔΗ = 2000 ΣΥΝΔΕΣΜΟΙ = 24290	218,401 78870	236,8 80870	131,9 107160	78,1 103160
ΚΕΝΤΡΟΕΙΔΗ = 2600 ΣΥΝΔΕΣΜΟΙ = 40000	463,72 127800	468,003 130400	156,01 173000	109,2 167800
ΚΕΝΤΡΟΕΙΔΗ = 2800 ΣΥΝΔΕΣΜΟΙ = 48580	612,506 154140	608,404 156940	202,801 208320	140,4 202720

Πίνακας 8.3.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των βέλτιστων διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού σε αραιά δίκτυα

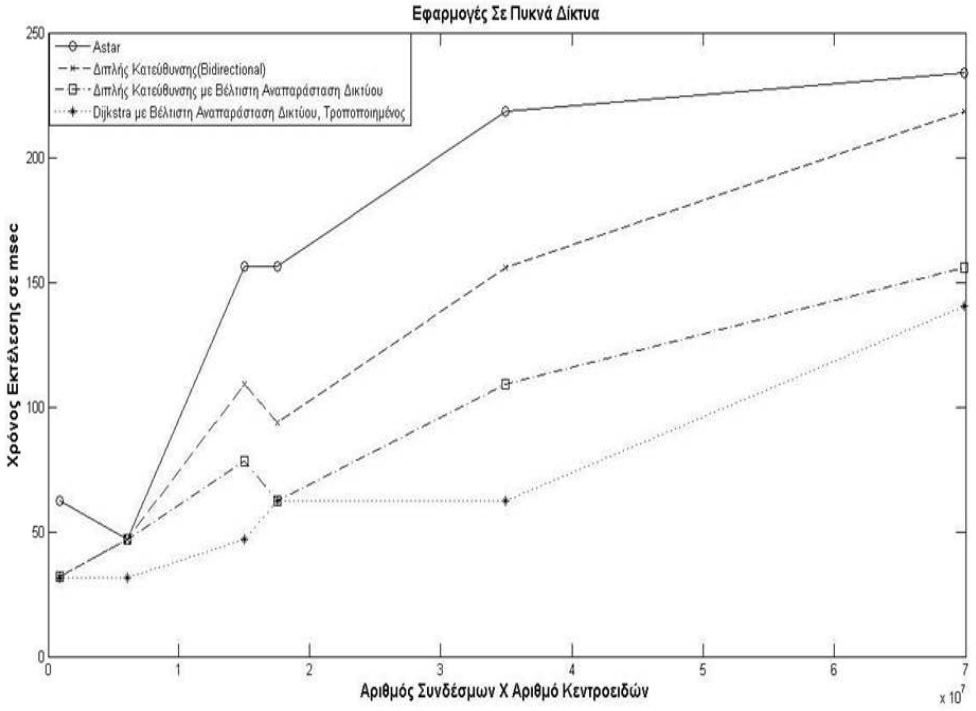


Διάγραμμα 8.3Χρόνοι εύρεσης των βέλτιστων διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού σε αραιά δίκτυα

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης
Πυκνά δίκτυα με ιδεατή αναλογία κόμβων / συνδέσμων
Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.010
Χρόνος Εκτέλεσης αλγορίθμου σε **msec** : Απαιτούμενες Θέσεις Μνήμης :

*Το προϋπολογισμένο δίκτυο που χρησιμοποιείται από τον αλγόριθμο A star δεν είναι βέλτιστο	A STAR*	ΔΙΠΛΗΣ ΚΑΤΕΥΘΥΝΣΗΣ (BIDIRECTIONAL)	ΔΙΠΛΗΣ ΚΑΤΕΥΘΥΝΣΗΣ ΜΕ ΒΕΛΤΙΣΤΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΙΚΤΥΟΥ	ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ DIJKSTRA
ΚΕΝΤΡΟΕΙΔΗ = 100 ΣΥΝΔΕΣΜΟΙ = 9600	62.4 31800	32,1 31900	32,1 38900	31,4 38700
ΚΕΝΤΡΟΕΙΔΗ = 300 ΣΥΝΔΕΣΜΟΙ = 20400	46,8 62100	46,8 62500	46,8 83100	31,7 82500
ΚΕΝΤΡΟΕΙΔΗ = 500 ΣΥΝΔΕΣΜΟΙ = 30035	156,1 91605	109,207 92105	78,1 122640	46,8 121640
ΚΕΝΤΡΟΕΙΔΗ = 500 ΣΥΝΔΕΣΜΟΙ = 35066	156,1 106698	93,6 107198	62,4 142764	62,4 141764
ΚΕΝΤΡΟΕΙΔΗ = 700 ΣΥΝΔΕΣΜΟΙ = 49932	218,401 151896	156,001 152596	109,2 203228	62,4 201828
ΚΕΝΤΡΟΕΙΔΗ = 700 ΣΥΝΔΕΣΜΟΙ = 99864	234,001 301692	218,401 302392	156,001 402956	140,4 401556

Πίνακας 8.4.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των βέλτιστων διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού σε πυκνά δίκτυα



Διάγραμμα 8.4.Χρόνοι εύρεσης των βέλτιστων διαδρομών μεταξύ δύο κόμβων προέλευσης-προορισμού σε πυκνά δίκτυα

Συμπεράσματα

Όλες οι μέθοδοι που χρησιμοποιήθηκαν έχουν παρουσιαστεί στις Ενότητες 5.2. και 5.4 και έχουν θεωρητικούς χρόνους εκτέλεσης :

Astar και αναπαράσταση δικτύου με λίστα γειτνίασης : *Εξαρτάται εύρος του πεδίου αναζήτησης*.
Διπλής Κατεύθυνσης και αναπαράσταση δικτύου με λίστα γειτνίασης : *Εξαρτάται από το εύρος του πεδίου αναζήτησης*.
Διπλής Κατεύθυνσης με Βέλτιστη Αναπαράσταση Δικτύου : *Εξαρτάται από το εύρος του πεδίου αναζήτησης*.
Dijkstra με Βέλτιστη Αναπαράσταση Δικτύου : $O((N+E)log_2 N)$.

Παρατηρήσεις μέσω των εφαρμογών :

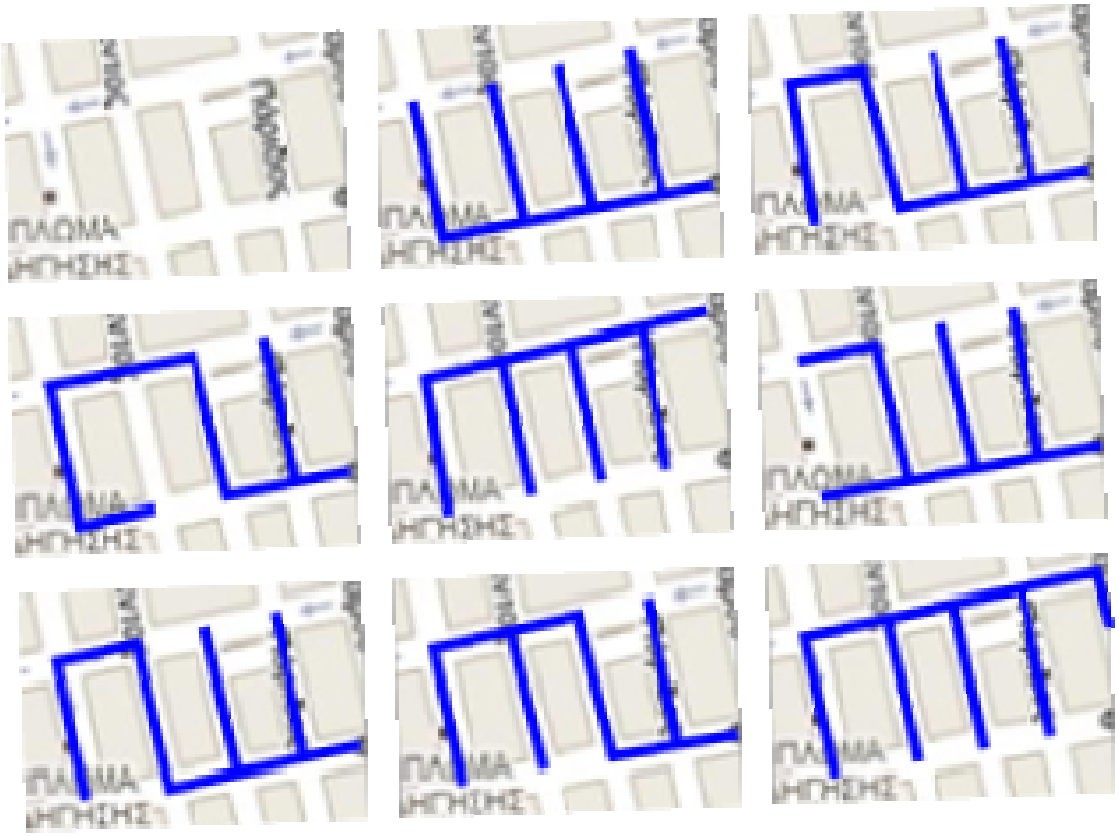
1) Οι αλγόριθμοι A* και Διπλής Κατεύθυνσης (Bidirectional) δεν έχουν συγκεκριμένο χρόνο εκτέλεσης επειδή είναι ευρετικοί αλγόριθμοι και ο χρόνος αυτός εξαρτάται από το πεδίο αναζήτησής τους. Τα πεδία αναζήτησης αυτών των μεθόδων έχουν παρουσιαστεί στα Σχήματα 5.8., 5.9. Τα πεδία αναζήτησης αυτών των μεθόδων αναμένεται να είναι μικρότερα από το πεδία αναζήτησης του αλγορίθμου Dijkstra, δηλαδή αναμένεται να εξερευνηθούν λιγότεροι κόμβοι που ανήκουν στο δίκτυο. Αυτό όμως δεν παρατηρείται τόσο έντονα μέσω των εφαρμογών. Βέβαια, το πεδίο αναζήτησης εξαρτάται και από τη μορφή του δικτύου και γι'αυτό το λόγο δε μπορούν να γίνουν εύκολα γενικεύσεις. Ως παράδειγμα σε ένα δίκτυο με 2800 κόμβους και 48896 συνδέσμους το πεδίο αναζήτησης ήταν ίδιο για τους αλγόριθμου Dijkstra και Διπλής Κατεύθυνσης και μάλιστα απαιτήθηκαν 2799 επαναλήψεις και από τους δύο αλγόριθμους. Μικρότερο ήταν το πεδίο αναζήτησης του αλγορίθμου A*, ο οποίος ολοκληρώθηκε μετά από 2269 επαναλήψεις, απέφυγε δηλαδή την εξέταση 530 κόμβων.

2) Ο αλγόριθμος Bellman-Ford δε μπορεί να χρησιμοποιηθεί αποδοτικά για την επίλυση αυτού του προβλήματος. Αυτό οφείλεται στο ότι ακόμα και εάν επιλεγεί ο κόμβος προορισμού s η διαδικασία δε μπορεί να τερματίσει διότι εάν δεν επιλυθεί όλο το δίκτυο δεν υπάρχει βεβαιότητα για την τιμή του κόστους D(s). Αυτό έχει σαν αποτέλεσμα ακόμα και εάν ένας κόμβος συνδέεται άμεσα με τον κόμβο προέλευσης να πρέπει να επιλυθεί όλο το δίκτυο για να προκύψει το κόστος του και απαιτείται αυξημένος χρόνος εκτέλεσης.

3) Ο αλγόριθμος Dijkstra έχει μικρότερους χρόνους εκτέλεσης από τις άλλες δύο μεθόδους. Βέβαια οι χρόνοι αυτοί είναι σχεδόν ίδιοι με τους χρόνους του αλγορίθμου Διπλής Κατεύθυνσης, ο οποίος χρησιμοποιεί άλλωστε τον αλγόριθμο Dijkstra ως υπορουτίνα. Το παραπάνω συμπέρασμα είναι συνέπεια των μικρών διαφορών στο πεδίο αναζήτησης που έχουν οι τρείς αλγόριθμοι, κάτι που δεν αναμενόταν θεωρητικά. Έτσι ο αλγόριθμος Dijkstra που απαιτεί λιγότερες βασικές πράξεις κατά την εκτέλεσή του σε σύγκριση με τους άλλους δύο ευρετικούς αλγορίθμους παρουσιάζει και μικρότερους χρόνους εκτέλεσης.

4) Ο αλγόριθμος A* χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Dijkstra και χρησιμοποιήθηκε σε δίκτυα με αναπαράσταση μέσω λίστας γειτνίασης. Αυτός είναι και ο κύριος λόγος που παρουσιάζει αυξημένους χρόνους εκτέλεσης. Ωστόσο ακόμα και κατά την εφαρμογή του σε τροποποιημένο δίκτυο προσεγγίζει το χρόνο εκτέλεσης του αλγορίθμου Dijkstra. Παρόλο που εξετάζει μικρότερο πεδίο αναζήτησης, όπως στο παράδειγμα που αναφέρθηκε προηγουμένως ο χρόνος εκτέλεσής του δεν είναι μικρότερος. Πιο συγκεκριμένα ο αλγόριθμος Dijkstra με 2799 επαναλήψεις ολοκληρώθηκε σε χρόνο 140,4 msec και ο αλγόριθμος A* σε τροποποιημένο δίκτυο αναπαράστασης με 2269 επαναλήψεις ολοκληρώθηκε στον ίδιο χρόνο. Δε μπορεί να αποκλειστεί βέβαια ότι ο χρόνος εκτέλεσης του αλγορίθμου A∗ μπορεί να γίνει μικρότερος από τον αντίστοιχο του Dijkstra σε ακόμα μεγαλύτερα δίκτυα, ωστόσο οι επιπλέον θέσεις μνήμης που απαιτεί και κυρίως η ανάγκη για έναν υπολογισμό του δικτύου πριν την εφαρμογή του δεν τον καθιστούν περισσότερο ελκυστικό.

8.4.ΑΞΙΟΛΟΓΗΣΗ ΜΕΘΟΔΩΝ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΔΡΟΜΗΣ ΑΠΟ ΟΛΟΥΣ ΤΟΥ ΚΟΜΒΟΥΣ ΠΡΟΣ ΟΛΟΥΣ ΤΟΥΣ ΚΟΜΒΟΥΣ ΤΟΥ ΔΙΚΤΥΟΥ



Εικόνα 8.3.Βέλτιστες διαδρομές από κάθε κόμβο προς κάθε κόμβο του δικτύου

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης

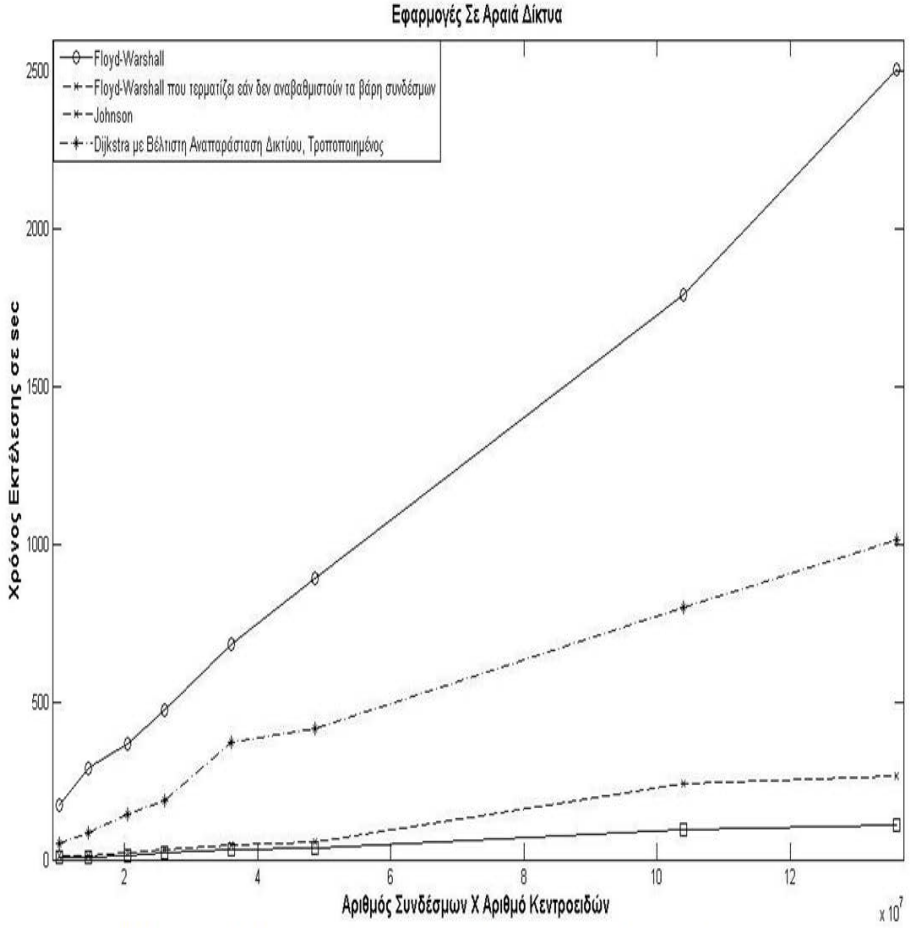
Δίκτυα με συνηθισμένη αναλογία κόμβων / συνδέσμων

Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10

Χρόνος Εκτέλεσης αλγορίθμου σε **sec** : [ ] Απαιτούμενες Θέσεις Μνήμης : [ ]

*Θεωρείται ότι η απόσταση κάθε κόμβου από το σταθερό κόμβο εκτυπόνεται σε κάθε βήμα και δεν απαιτείται η αποθήκευσή της	FLOYD-WARSHALL	FLOYD-WARSHALL ΜΕ ΥΠΟΡΟΥΤΙΝΑ ΤΗ ΜΕΘΟΔΟ BELLMAN-FORD ΠΟΥ ΤΕΡΜΑΤΙΖΕΙ ΕΑΝ ΔΕΝ ΑΝΑΒΑΘΜΙΣΤΟΥΝ ΤΑ ΒΑΡΗ ΣΥΝΔΕΣΜΩΝ	JOHNSON	ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ DIJKSTRA ΓΙΑ ΤΟ ASSPP ΜΕ ΒΕΛΤΙΣΤΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΙΚΤΥΟΥ
ΚΕΝΤΡΟΕΙΔΗ = 1000 ΣΥΝΔΕΣΜΟΙ = 10010	172,46 31030*	10,03 31030*	50,0405 51050*	5,71 43040*
ΚΕΝΤΡΟΕΙΔΗ = 1200 ΣΥΝΔΕΣΜΟΙ = 12010	286,823 37230	13,6033 37230	86,47135 66050	9,344 51640
ΚΕΝΤΡΟΕΙΔΗ = 1400 ΣΥΝΔΕΣΜΟΙ = 14610	368,172 45230	21,591 45230	141,3525 80050	14,508 62640
ΚΕΝΤΡΟΕΙΔΗ = 1600 ΣΥΝΔΕΣΜΟΙ = 16210	474,629 50230	31,5902 50230	189,176 89050	20,53 69640
ΚΕΝΤΡΟΕΙΔΗ = 1800 ΣΥΝΔΕΣΜΟΙ = 20000	684,004 61800	45,7551 61800	372,422 109000	29,923 85400
ΚΕΝΤΡΟΕΙΔΗ = 2000 ΣΥΝΔΕΣΜΟΙ = 24290	893,873 74870	57,315 74870	416,936 131450	36,192 103160
ΚΕΝΤΡΟΕΙΔΗ = 2600 ΣΥΝΔΕΣΜΟΙ = 40000	1792,63 122600	238,712 122600	798,65 213000	93,6 167800
ΚΕΝΤΡΟΕΙΔΗ = 2800 ΣΥΝΔΕΣΜΟΙ = 48580	2502,84 148540	264,484 148540	1013,45 256900	106,9231 202720

Πίνακας 8.5.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των βέλτιστων διαδρομών από όλους τους κόμβους προς όλους τους κόμβους σε αραιά δίκτυα



Διάγραμμα 8.5.Χρόνοι Εύρεσης βέλτιστων διαδρομών από όλους τους κόμβους προς όλους τους κόμβους σε αραιά δίκτυα

Συμπεράσματα

Όλες οι μέθοδοι που χρησιμοποιήθηκαν έχουν παρουσιαστεί στην Ενότητα 5.5. και έχουν θεωρητικούς χρόνους εκτέλεσης :

Floyd-Warshall με Υπορουτίνα τον αλγόριθμο Bellman-Kalaba : $O(N^2 \times E)$.
Floyd-Warshall που Τερματίζει εάν δεν αναβαθμιστούν τα κόστη των Κόμβων : $O(N \times \delta \times E)$.
Johnson με Υπορουτίνα τον αλγόριθμο Dijkstra και Δομή Δεδομένων απλή Λίστα : $O(N^3)$.
Τροποποιημένος Dijkstra με Βέλτιστη Αναπαράσταση Δικτύου : $O(N((N+E)log_2 N))$.

Παρατηρήσεις μέσω των εφαρμογών :

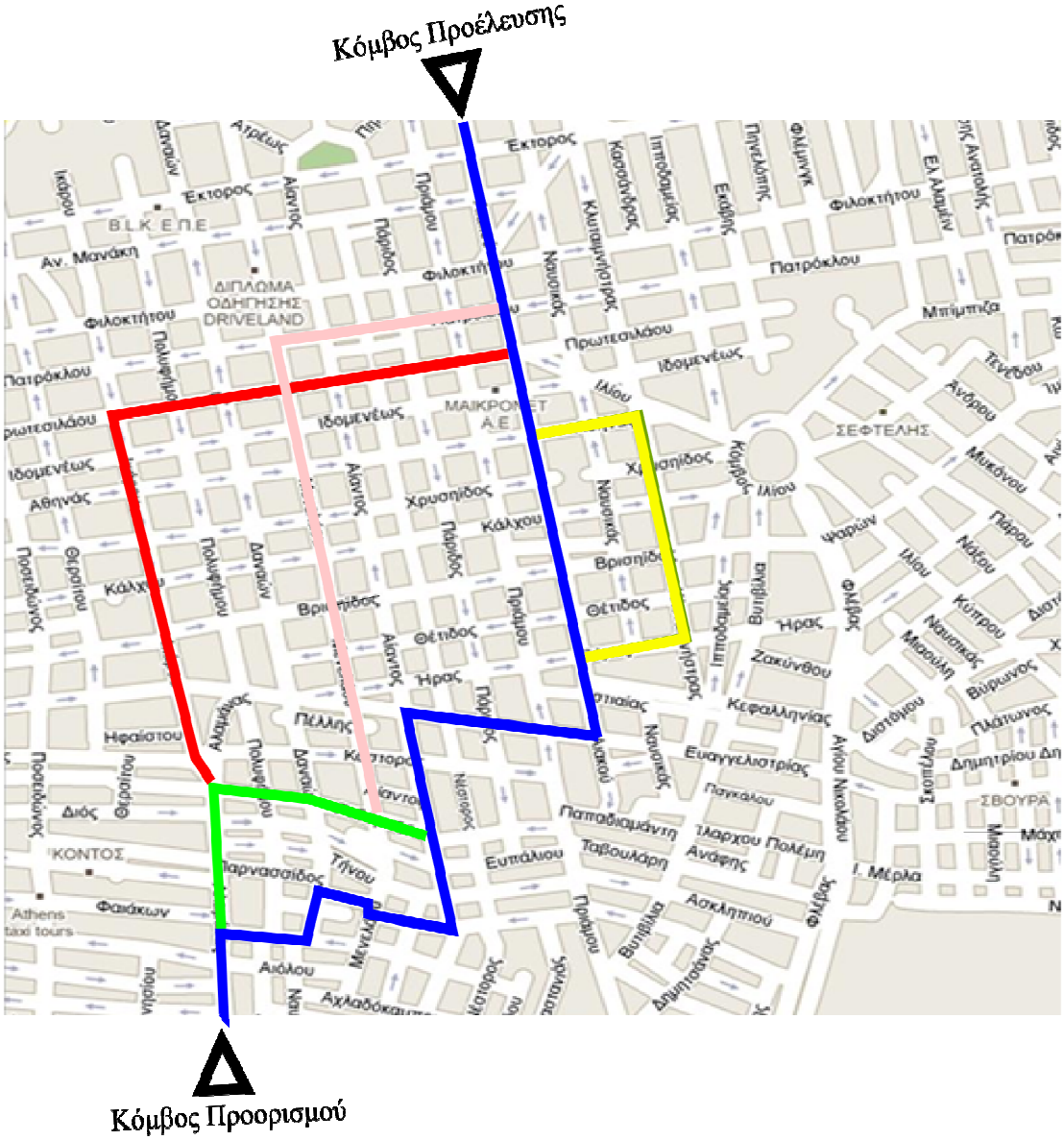
1) Ο υπολογισμός των βέλτιστων διαδρομών από όλους τους κόμβους προς όλους τους κόμβους απαιτεί πολυωνυμικό χρόνος εκτέλεσης, εντός των ορίων $O(k \times N^2)$, $O(N^2 \times E)$ όπου k > 1. Έτσι όσο αυξάνεται το μέγεθος του δικτύου αυξάνεται έντονα και ο χρόνος εκτέλεσης. Πιο συγκεκριμένα, εξετάζοντας τον αλγόριθμο Dijkstra μέσω των εφαρμογών ο χρόνος εκτέλεσής του αυξάνεται σύμφωνα με τη σχέση :

Χρόνος Εκτέλεσης = $\frac{N^3}{170000000 \text{ έως } 230000000}$ (sec), σε συνήθη συγκοινωνιακά δίκτυα.
Ο χρόνος αυτός ξεπερνά το 1 sec σε δίκτυα με περισσότερους από 600 κόμβους περίπου. Εάν βέβαια ο αριθμός των κόμβων αυξηθεί και άλλο ο χρόνος εκτέλεσης αυξάνεται ραγδαία (1000 κόμβοι 5,71 sec). Για το λόγο αυτό, το πρόβλημα δε μπορεί να αντιμετωπιστεί εντός ρεαλιστικού χρόνου σε μεγάλα δίκτυα ανεξαιρέτως του αλγορίθμου που χρησιμοποιείται.

β) Ο αλγόριθμος Floyd-Warshall τερματίζει μετά από λίγες επαναλήψεις σε χρόνο $O(N \times \delta \times E)$. Όπως έχει αναφερθεί ήδη ο αριθμός των επαναλήψεων 'δ' είναι πολύ μικρότερος του αριθμού των κόμβων N σε πρακτικές εφαρμογές. Σε συνδυασμό μάλιστα με τις μικρές απαιτήσεις σε θέσεις μνήμης ο αλγόριθμος αυτός καθίσταται ιδιαίτερα ελκυστικός, σε κάθε περίπτωση όμως έχει μεγαλύτερου χρόνους εκτέλεσης από τον τροποποιημένο αλγόριθμο Dijkstra.

γ) Ο αλγόριθμος Johnson αποτελείται από δύο ξεχωριστές υπορουτίνες, αρχικά χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Bellman-Ford και στη συνέχεια επαναλαμβάνει N φορές ως υπορουτίνα τον αλγόριθμο Dijkstra. Φυσικά η δεύτερη διαδικασία είναι η κρίσιμη και για το λόγο αυτό ο χρόνος εκτέλεσής του εξαρτάται από τη δομή δεδομένων που χρησιμοποιείται στον αλγόριθμο Dijkstra και τον τρόπο αναπαράστασης του δικτύου. Στις εφαρμογές το δίκτυο αναπαρίσταται υπό τη μορφή λίστας γειτνίασης και για το λόγο αυτό προκύπτουν αυξημένοι χρόνοι εκτέλεσης. Εάν όμως τροποποιηθεί η αναπαράσταση του δικτύου ο χρόνος εκτέλεσής του προσεγγίζει αυτόν του τροποποιημένου αλγόριθμου Dijkstra. Για να πραγματοποιηθεί αυτή η αναπαράσταση όμως απαιτούνται πολλές θέσεις μνήμης.

8.5.ΑΞΙΟΛΟΓΗΣΗ ΜΕΘΟΔΩΝ ΥΠΟΛΟΓΙΣΜΟΥ ΕΝΑΛΛΑΚΤΙΚΩΝ ΔΙΑΔΡΟΜΩΝ ΜΕΤΑΞΥ ΔΥΟ ΚΟΜΒΩΝ ΠΡΟΕΛΕΥΣΗΣ - ΠΡΟΟΡΙΣΜΟΥ

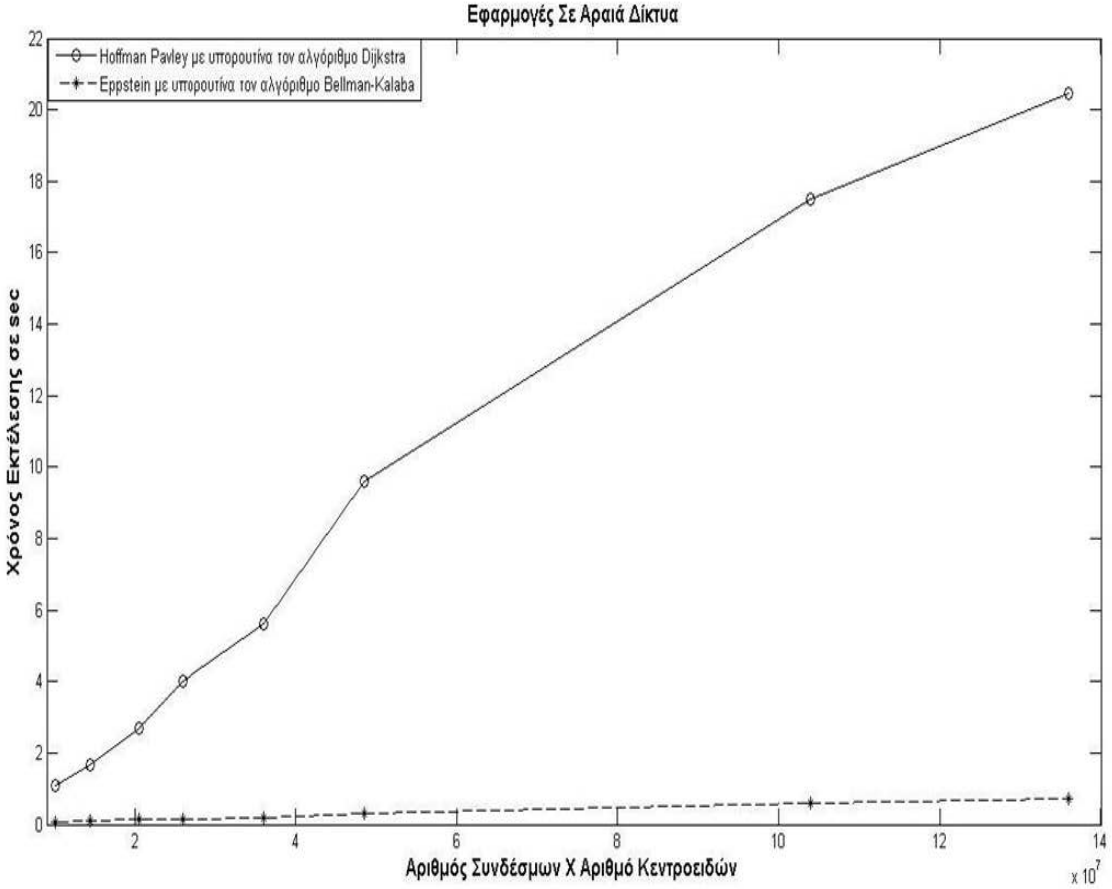


Εικόνα 8.4.Οι πρώτες k διαδρομές ελαχίστου κόστους μεταξύ δύο κόμβων προέλευσης-προορισμού

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης
Δίκτυα με συνηθισμένη αναλογία κόμβων / συνδέσμων
Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10
Χρόνος Εκτέλεσης αλγορίθμου σε **sec** :
Απαιτούμενες Θέσεις Μνήμης :

	HOFFMAN-PAVLEY ΜΕ ΥΠΟΡΟΥΤΙΝΑ ΤΟΝ ΑΛΓΟΡΙΘΜΟ DIJKSTRA	EPPSTEIN ΜΕ ΥΠΟΡΟΥΤΙΝΑ ΤΟΝ ΑΛΓΟΡΙΘΜΟ BELLMAN-KALABA
ΚΕΝΤΡΟΕΙΔΗ = 1000 ΣΥΝΔΕΣΜΟΙ = 10010	1,1076 56050	0,078 ~70570
ΚΕΝΤΡΟΕΙΔΗ = 1200 ΣΥΝΔΕΣΜΟΙ = 12010	1,685 67250	0,0936 ~90070
ΚΕΝΤΡΟΕΙΔΗ = 1400 ΣΥΝΔΕΣΜΟΙ = 14610	2,6832 81450	0,1404 ~109270
ΚΕΝΤΡΟΕΙΔΗ = 1600 ΣΥΝΔΕΣΜΟΙ = 16210	3,9936 90650	0,1404 ~121470
ΚΕΝΤΡΟΕΙΔΗ = 1800 ΣΥΝΔΕΣΜΟΙ = 20000	5,613 110800	0,1716 ~149000
ΚΕΝΤΡΟΕΙΔΗ = 2000 ΣΥΝΔΕΣΜΟΙ = 24290	9,61 133450	0,2964 ~180030
ΚΕΝΤΡΟΕΙΔΗ = 2600 ΣΥΝΔΕΣΜΟΙ = 40000	17,51 215600	0,6084 ~293000
ΚΕΝΤΡΟΕΙΔΗ = 2800 ΣΥΝΔΕΣΜΟΙ = 48580	20,4517 259700	0,702 ~354060

Πίνακας 8.6.Συγκριτικά Αποτελέσματα χρόνου εύρεσης των δύο πρώτων διαδρομών ελαχίστου κόστους σε αραιά δίκτυα



Διάγραμμα 8.6.Χρόνοι εύρεσης των δύο πρώτων διαδρομών ελαχίστου κόστους σε αραιά δίκτυα

Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης

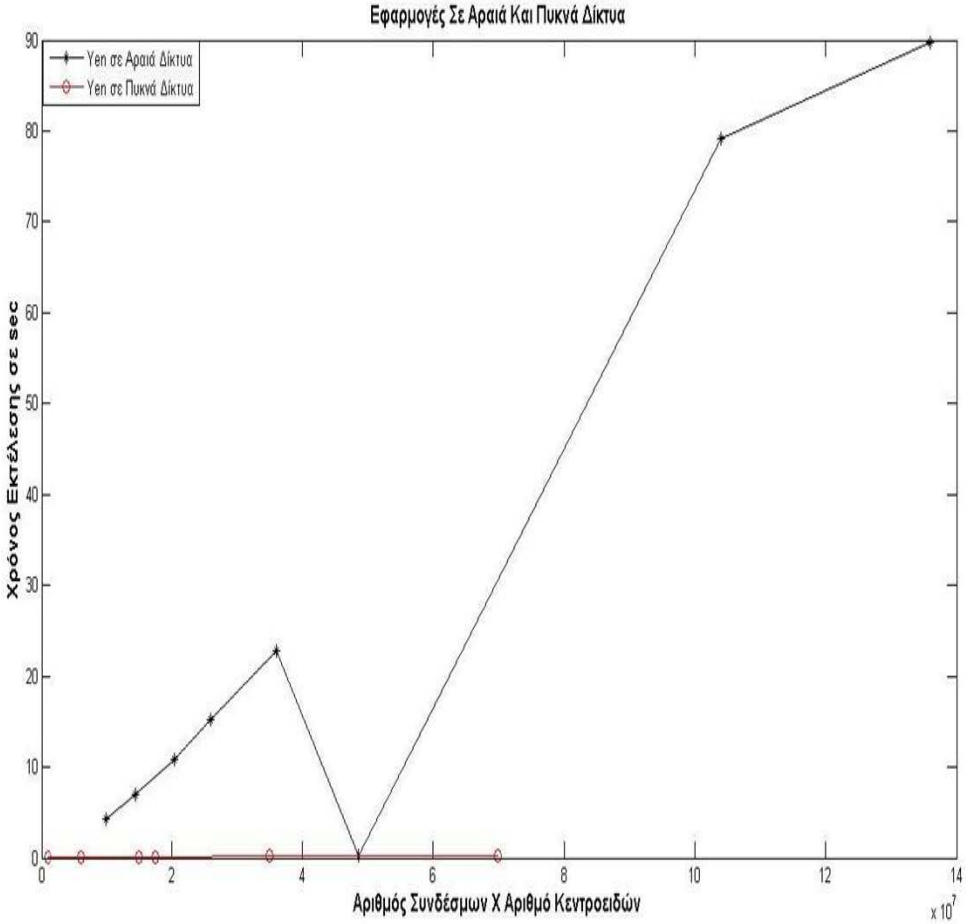
Αραιά και Πυκνά δίκτυα με ιδεατή αναλογία κόμβων / συνδέσμων

Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10 Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.010

Χρόνος Εκτέλεσης αλγορίθμου σε **sec** : Απαιτούμενες Θέσεις Μνήμης :

*Αριθμός Δένδρων Ελάχιστων Διαδρομών που προκύπτουν κατά την εκτέλεση	YEN	YEN	
	7435, 7780	4, 12 →	Κόστοι Βέλτιστης και κοστής διαδρομής
ΚΕΝΤΡΟΕΙΔΗ = 1000	4,212	0,062	ΚΕΝΤΡΟΕΙΔΗ = 100
ΣΥΝΔΕΣΜΟΙ = 10010	3009000 / 167*	30900 / 5*	ΣΥΝΔΕΣΜΟΙ = 9600
	10300, 10705	6, 13	
ΚΕΝΤΡΟΕΙΔΗ = 1200	6,9576	0,062	ΚΕΝΤΡΟΕΙΔΗ = 300
ΣΥΝΔΕΣΜΟΙ = 14610	4330800 / 194*	272700 / 5*	ΣΥΝΔΕΣΜΟΙ = 20400
	10546, 11079	18, 37	
ΚΕΝΤΡΟΕΙΔΗ = 1400	10,904	0,1092	ΚΕΝΤΡΟΕΙΔΗ = 500
ΣΥΝΔΕΣΜΟΙ = 14610	5892600 / 215*	754500 / 8*	ΣΥΝΔΕΣΜΟΙ = 30035
	12628, 13046	16, 27	
ΚΕΝΤΡΟΕΙΔΗ = 1600	15,1790	0,1092	ΚΕΝΤΡΟΕΙΔΗ = 500
ΣΥΝΔΕΣΜΟΙ = 16210	7694400 / 255*	754500 / 7*	ΣΥΝΔΕΣΜΟΙ = 35066
	13509, 13943	55, 82	
ΚΕΝΤΡΟΕΙΔΗ = 1800	22,7605	0,2652	ΚΕΝΤΡΟΕΙΔΗ = 700
ΣΥΝΔΕΣΜΟΙ = 20000	9736200 / 272*	1476300 / 14*	ΣΥΝΔΕΣΜΟΙ = 49932
	∞	11, 25	
ΚΕΝΤΡΟΕΙΔΗ = 2000	0,2496	0,327	ΚΕΝΤΡΟΕΙΔΗ = 700
ΣΥΝΔΕΣΜΟΙ = 24290	12018000 / 0*	1476300 / 10*	ΣΥΝΔΕΣΜΟΙ = 99864
	17460, 17777		
ΚΕΝΤΡΟΕΙΔΗ = 2600	79,071		
ΣΥΝΔΕΣΜΟΙ = 40000	20303400 / 401*		
	17312 / 18522		
ΚΕΝΤΡΟΕΙΔΗ = 2800	89,747		
ΣΥΝΔΕΣΜΟΙ = 48580	23545200 / 422*		

Πίνακας 8.7.Χρόνοι εύρεσης όλων των εναλλακτικών διαδρομών μεταξύ δύο κόμβων κατά σειρά προτεραιότητας ως προς το κόστος διάνυσης



Διάγραμμα 8.7.Χρόνοι Εύρεσης όλως των εναλλακτικών διαδρομών σε αραιά δίκτυα

Συμπεράσματα

Όλες οι μέθοδοι που χρησιμοποιήθηκαν έχουν παρουσιαστεί στην Ενότητα 5.6. και έχουν θεωρητικούς χρόνους εκτέλεσης :

Hoffman-Pavley : $O(\mathrm{k} \times (\mathrm{N+E}) log_2 N))$, όπου k ο αριθμός των κόμβων που ανήκουν στη διαδρομή ελαχίστου κόστους.
Eppstein με Υπορουτίνα τον αλγόριθμο Bellman-Kalaba για την εύρεση της $2^{ης}$ διαδρομής ελαχίστου κόστους: $O((\mathrm{N+E}) log_2 N))$.
Yen για την εύρεση όλων των εναλλακτικών διαδρομών, την ταξινόμησή τους ανάλογα με το κόστος τους και την εκτύπωσή τους : $O(\mathrm{k} \times \mathrm{N}(\mathrm{E+N}(\mathrm{N} \times \log \mathrm{N})))$.

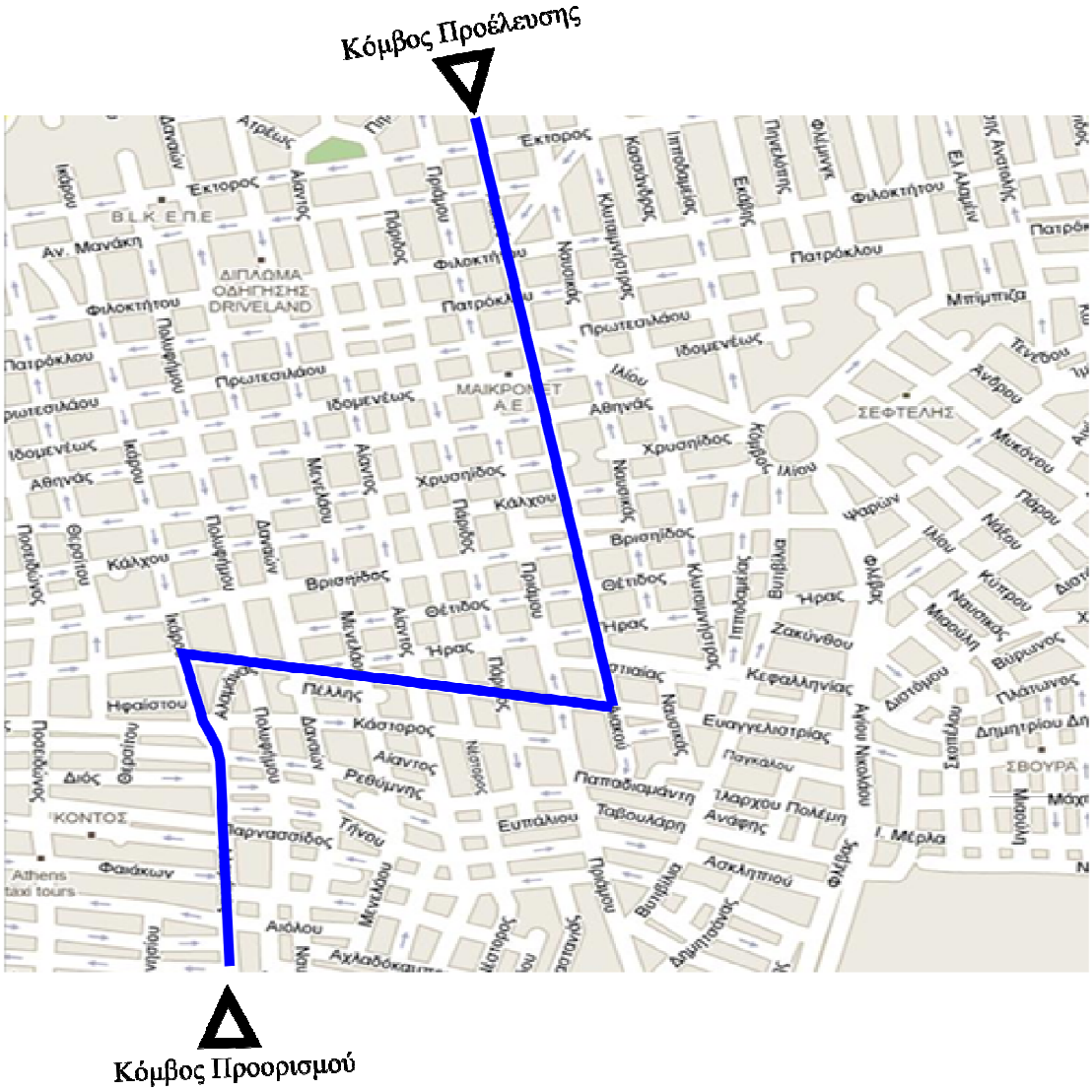
Παρατηρήσεις μέσω των εφαρμογών :

1) Όσον αφορά το πρόβλημα της $2^{ης}$ διαδρομής ελαχίστου κόστους, ο αλγόριθμος Hoffman-Pavley χρησιμοποιεί ως υπορουτίνα τον αλγόριθμο Dijkstra και απαιτεί k επαναλήψεις του, όσοι είναι δηλαδή και οι κόμβοι που ανήκουν στη διαδρομή ελαχίστου κόστους. Αντίθετα ο αλγόριθμος Eppstein επιλύει το πρόβλημα με μία επανάληψη του αλγορίθμου Dijkstra ή Bellman-Kalaba, χρησιμοποιώντας βέβαια περισσότερες θέσεις μνήμης. Για το λόγο αυτό οι χρόνοι εκτέλεσής τους διαφέρουν σημαντικά. Για παράδειγμα σε τυχαίο δίκτυο 2800 κόμβων και 48896 ο αλγόριθμος Hoffman-Pavley απαιτεί χρόνο εκτέλεσης 20,4517 sec και 259700 θέσεις μνήμης, ενώ ο Eppstein 0,702 sec και 354060 αντίστοιχα. Το γεγονός αυτό είναι ιδιαίτερα σημαντικό στο χώρο των συστημάτων πλοήγησης καθώς χρησιμοποιώντας τη μέθοδο του Eppstein μπορεί να προταθεί και η δεύτερη διαδρομή ελαχίστου κόστους στο χρήστη χωρίς κάποια ιδιαίτερη χρονική επιβάρυνση. Κάτι τέτοιο αποδεικνύεται και μέσω των εφαρμογών όπου οι χρόνοι εύρεσης της $2^{ης}$ διαδρομής ελαχίστου κόστους μέσω αυτής της μεθόδου σε δίκτυα που αναπαρίστανται μέσω λίστας γειτνίασης δε διαφέρει από το χρόνο εύρεσης της βέλτιστης διαδρομής με τη μέθοδο Dijkstra περισσότερο από 10%. Αυτό οφείλεται στο ότι όλες οι βασικές πράξεις του αλγορίθμου Eppstein απαιτούν γραμμικό χρόνο εκτέλεσης εκτός από τον αλγόριθμο Dijkstra που χρησιμοποιεί ως υπορουτίνα και αποτελεί την κρίσιμη διαδικασία για τον καθορισμό του χρόνου εκτέλεσής του.

2) Κατά την επίλυση του προβλήματος εύρεσης όλων των εναλλακτικών διαδρομών μεταξύ δύο κόμβων και την ταξινόμησή τους με βάση το κόστος του χρησιμοποιήθηκε ο αλγόριθμος Yen με τροποποιήσεις και αναπαράσταση του δικτύου μέσω πίνακα γειτνίασης. Ο αλγόριθμος Eppstein έχει αδυναμίες κατά την εφαρμογή του σε τυχαία δίκτυα καθώς πολλές φορές καταλήγει σε ατέρμονες βρόγχους. Από την εφαρμογή της μεθόδου του Yen σε συγκοινωνιακά δίκτυα προκύπτουν κρίσιμα συμπεράσματα. Αρχικά μέσα από την εφαρμογή του σε δίκτυο με 1000 κόμβους και 10010 συνδέσμους προκύπτουν 167 δένδρα εναλλακτικών διαδρομών μεταξύ των κόμβων r, s ταξινομημένα με βάση το κόστος κάθε διαδρομής σε χρόνο 4,212 sec. Βέβαια δεν έχει ιδιαίτερη σημασία η παρουσίαση και των 167 διαδρομών στο χρήστη και για το λόγο αυτό η μέθοδος μπορεί να τερματίζει έπειτα

από την εύρεση των k πρώτων εναλλακτικών διαδρομών. Το σημαντικό είναι ότι με αυτό τον τρόπο μπορεί να επιλεγεί και η πρώτη διαδρομή που έχει το ελάχιστο κόστος και ικανοποιεί και μία σειρά περιορισμών. Για παράδειγμα μπορεί να υπολογιστεί η πρώτη διαδρομή ελαχίστου κόστους με αποτύπωμα άνθρακα μικρότερο από κάποια συγκεκριμένη ποσότητα και αριθμό διανυόμενων χιλιομέτρων που δεν ξεπερνούν μία συγκεκριμένη τιμή. Βέβαια όταν το δίκτυο είναι αρκετά μεγάλο οι χρόνοι εκτέλεσης του αλγορίθμου δεν είναι αποδεκτοί σύμφωνα με τη σημερινή τεχνολογία των συστημάτων πλοήγησης. Για παράδειγμα εφαρμογή σε τυχαίο δίκτυο με 2800 κόμβους και 48580 συνδέσμους απαιτεί χρόνο εκτέλεσης 89,747 sec που δεν είναι πλέον αποδεκτός από το χρήστη. Ελέγχεται η αξιοποίηση μίας μεθόδου κατακερματισμού του δικτύου σε επιμέρους υπογραφήματα, όπως αυτή που παρουσιάστηκε στην Ενότητα 5.9. για τη μείωση του μεγέθους του δικτύου.

8.6.ΑΞΙΟΛΟΓΗΣΗ ΜΕΘΟΔΩΝ ΥΠΟΛΟΓΙΣΜΟΥ ΒΕΛΤΙΣΤΩΝ ΔΙΑΔΡΟΜΩΝ ΥΠΟ ΠΕΡΙΟΡΙΣΜΟΥΣ ΔΙΑΘΕΣΙΜΩΝ ΠΟΡΩΝ



Εικόνα 8.5.Διαδρομή ελαχίστου κόστους : $w_p = \sum_{e \epsilon p_*} w(e) = \min \{ \sum_{e \epsilon p} w(e) \}$, ∀ p η οποία ικανοποιεί ορισμένους περιορισμούς διαθέσιμων πόρων (περιβαλλοντικούς, οικονομικούς κ.α.) : $\sum_{e \epsilon p_*} R^i_{(e)} \leq b_i$, ∀ i = 1, 2, .., n

∗Σε αυτή την ενότητα ελέγχεται εάν είναι εφικτός από άποψη χρόνου εκτέλεσης ο υπολογισμός της διαδρομής ελαχίστου κόστους η οποία πληρεί και έναν περιορισμό. Ο περιορισμός αυτός μπορεί να είναι περιβαλλοντικής φύσεως (αποτύπωμα άνθρακα, κατανάλωση βενζίνης κ.α.). Για το σκοπό αυτό και επειδή τα συστήματα πλοήγησης έχουν περιορισμένες υπολογιστικές δυνατότητες προγραμματίστηκε ο αλγόριθμος LARAC και συγκρίνονται τα αποτελέσματα του χρόνου εκτέλεσής του με τα αντίστοιχα του αλγόριθμο Dijkstra.

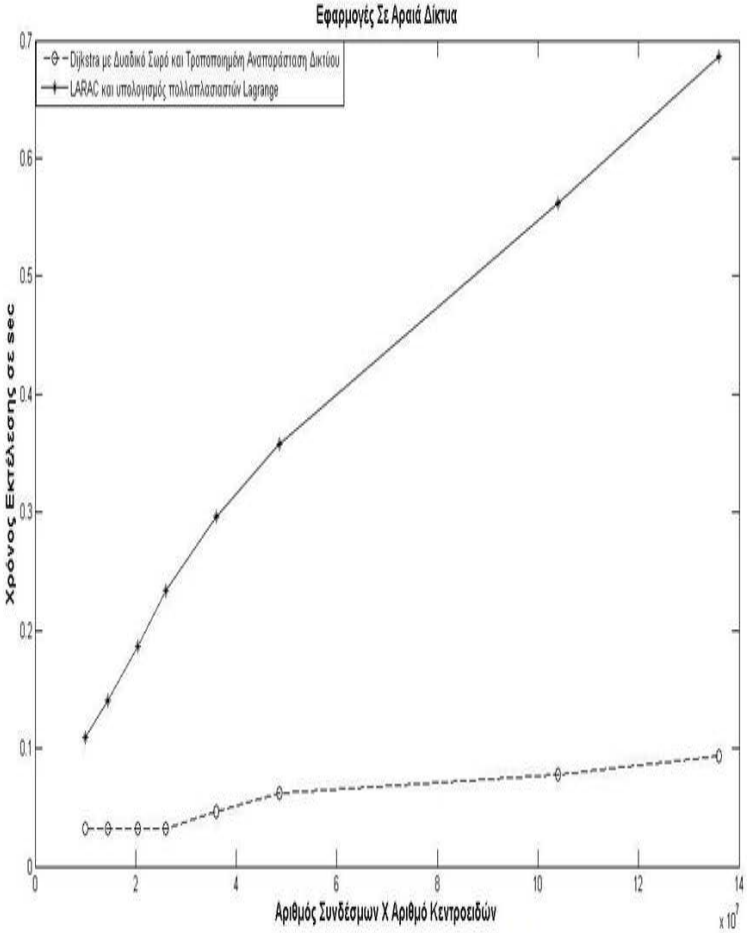
Αναπαράσταση Συγκοινωνιακών Δικτύων μέσω Λίστας Γειτνίασης

Δίκτυα με συνηθισμένη αναλογία κόμβων / συνδέσμων

Αριθμός Κόμβων / Αριθμό Συνδέσμων ≈ 0.10

				ΜΕΘΟΔΟΣ LARAC				ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA
ΚΕΝΤΡΟΕΙΔΗ	ΣΥΝΔΕΣΜΟΙ	ΠΕΡΙΟΣΜΟΣ ΔΙΑΘΕΣΙΜΟΥ ΠΟΡΟΥ	ΕΛΑΧΙΣΤΟ ΚΟΣΤΟΣ ΔΙΑΔΡΟΜΗΣ ΧΩΡΙΣ ΠΕΡΙΟΡΙΣΜΟΥΣ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ σε sec	ΑΡΙΘΜΟΣ ΕΠΑΝΑΛΗΨΕΩΝ ΜΕΧΡΙ ΤΗ ΒΕΛΤΙΣΤΗ ΣΥΓΚΛΙΣΗ	ΚΟΣΤΟΣ ΑΝΩ ΟΡΙΟΥ ΚΑΙ ΚΑΤΑΝΑΛΩΣΗ ΔΙΑΘΕΣΙΜΟΥ ΠΟΡΟΥ	ΚΟΣΤΟΣ ΚΑΤΩ ΟΡΙΟΥ ΚΑΙ ΚΑΤΑΝΑΛΩΣΗ ΔΙΑΘΕΣΙΜΟΥ ΠΟΡΟΥ	ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ σε sec
1000	10010	700	5176	0,1092	9	5253 716	5358 687	0,0321
1200	12010	800	7166	0,1404	10	7189 801	7220 779	0,0321
1400	14610	600	7352	0,187	9	9697 604	9936 584	0,0321
1600	16210	700	9107	0,234	11	10583 707	10658 699	0,0321
1800	20000	800	9023	0,2964	10	10893 803	10951 797	0,0468
2000	24290	750	8779	0,358	12	11172 761	11291 749	0,0624
2600	40000	950	10889	0,5616	11	14391 956	14493 946	0,0781
2800	48580	900	9725	0,6864	11	13926 910	14167 889	0,0936

Πίνακας 8.8.Αποτελέσματα μεθόδου εύρεσης της διαδρομής ελαχίστου κόστους που ικανοποιεί έναν περιορισμό



Διάγραμμα 8.8.Συγκριτικά Αποτελέσματα χρόνων εκτέλεσης των αλγορίθμων LARAC, Dijkstra

Συμπεράσματα

Η μέθοδος που χρησιμοποιήθηκε έχει παρουσιαστεί στην Ενότητα 5.7. και έχει θεωρητικό χρόνο εκτέλεσης :

LARAC με Υπορουτίνα τον αλγόριθμο Dijkstra σε τροποποιημένο δίκτυο :
$O(\text{k} \times (\text{N+E}) log_2 N)$), όπου k ο αριθμός των επαναλήψεων μέχρι να επιτευχθεί η σύγκλιση.
Dijkstra με Βέλτιστη Αναπαράσταση Δικτύου : $O((\text{N+E}) log_2 N)$

Παρατηρήσεις μέσω των εφαρμογών :

1) Στο παραπάνω πρόβλημα εξετάστηκε η μέθοδος LARAC διότι έχει μικρότερο χρόνο εκτέλεσης σε σύγκριση με τις άλλες μεθόδους που παρουσιάστηκαν στην Ενότητα 5.7. Η μέθοδος αυτή βέβαια επιλύει προβλήματα με έναν περιορισμό διαθέσιμων πόρων και δε μπορεί να εφαρμοστεί σε γενικότερες περιπτώσεις.

2) Η μέθοδος LARAC ακολουθεί μία επαναληπτική διαδικασία χρησιμοποιώντας σε κάθε βήμα τον αλγόριθμο Dijkstra για την επίλυση του επιμέρους υποπροβλήματος. Ο χρόνος εκτέλεσής της εξαρτάται από τον αριθμό των επαναλήψεων που απαιτούνται ώστε να επιτευχθεί η σύγκλιση. Σύμφωνα με τις εφαρμογές ο αριθμός των επαναλήψεων κυμαίνεται από 9-12 σε συνήθη δίκτυα. Έτσι προκύπτει χρόνος εκτέλεσης ≈ $O(\ 11 \times (\text{N+E}) log_2 N)$). Ο αριθμός αυτός είναι ενδεικτικός και μπορεί να διαφέρει σε άλλες εφαρμογές, ωστόσο φαίνεται ότι το μέγεθός του δε μπορεί να εμποδίσει τη χρήση αυτής της μεθόδου από τα συστήματα πλοήγησης. Για το λόγο αυτό, το μεγαλύτερο πρόβλημα πλέον έγγειται στον προσδιορισμό του διαθέσιμου πόρου που καταναλώνει κάθε σύνδεσμος.

3) Ο χρόνος εκτέλεσης αυτής της μεθόδου αυξάνεται όσο μεγαλώνει το δίκτυο σε σύγκριση με το χρόνο εκτέλεσης του αλγορίθμου Dijkstra. Σύμφωνα με τις εφαρμογές ο λόγος των χρόνων εκτέλεσης των δύο αλγορίθμων είναι ≈ 6,898. Αυτό όμως σημαίνει ότι κάθε αύξηση του χρόνου εκτέλεσης του αλγορίθμου Dijkstra (μέσω της αύξησης του αριθμού των κόμβων και των συνδέσμων του δικτύου) συνεπάγεται επταπλάσια περίπου αύξηση του χρόνου εκτέλεσης του αλγορίθμου LARAC. Το πρόβλημα αυτό είναι εντονότερο όσο το μέγεθος του συγκοινωνιακού δικτύου μεγαλώνει, όπως παρουσιάζεται και στο διάγραμμα. Για το λόγο αυτό καλό θα ήταν να εφαρμοστεί μία μέθοδος κατακερματισμού του δικτύου πριν εφαρμοστεί ο αλγόριθμος σε μεγάλα δίκτυα.

8.7.ΔΙΕΡΕΥΝΗΣΗ ΤΟΥ ΚΑΤΑΜΕΡΙΣΜΟΥ ΤΗΣ ΖΗΤΗΣΗΣ ΣΕ ΣΥΓΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ ΜΕΣΩ ΤΗΣ ΧΡΗΣΗΣ ΣΥΣΤΗΜΑΤΩΝ ΠΛΟΗΓΗΣΗΣ

Όπως έχει αναφερθεί ήδη, η κυκλοφοριακή συμφόρηση αποτελεί κυρίαρχο πρόβλημα στο χώρο των συγκοινωνιακών συστημάτων προκαλώντας ανώφελη κατανάλωση οικονομικών και ενεργειακών πόρων, επιβαρύνοντας την οικονομία και το περιβάλλον. Το κατά πόσο μπορούν να χρησιμοποιηθούν τα συστήματα πλοήγησης για τον περιορισμό αυτών των προβλημάτων διερευνήθηκε στα Κεφάλαια 7, 8. Στη συνέχεια πραγματοποιούνται διάφορες εφαρμογές μέσω της προσομοίωσης των κυκλοφοριακών συνθηκών που επικρατούν σε συγκοινωνιακά δίκτυα, διερευνώντας πέντε περιπτώσεις.

α) Τα αποτελέσματα που έχει η χρήση των συστημάτων πλοήγησης σε στατικά δίκτυα. Για το σκοπό αυτό γίνεται η παραδοχή ότι όλοι οι χρήστες του δικτύου διαθέτουν συστήματα πλοήγησης και κάθε χρήστης ακολουθεί τη συντομότερη διαδρομή που του προτείνεται. Τα κόστη διάνυσης των συνδέσμων δε μεταβάλλονται λόγω της κυκλοφοριακής συμφόρησης και για το λόγο αυτό μπορεί να χρησιμοποιηθεί ο καταμερισμός της ζήτησης με τη μέθοδο του όλα ή τίποτα. Περισσότερες πληροφορίες σχετικά με αυτή την περίπτωση υπάρχουν στην Ενότητα 6.2.

β) Τα αποτελέσματα που έχει ο καταμερισμός της ζήτησης σε δυναμικά δίκτυα σύμφωνα με τη στοχαστική ισορροπία του χρήστη. Στην περίπτωση αυτή θεωρείται ότι μόνο ένα μικρό ποσοστό των χρηστών διαθέτει σύστημα πλοήγησης και το ποσοστό αυτό δεν επηρεάζει τον καταμερισμό της ζήτησης στο δίκτυο. Γίνεται επίσης η παραδοχή ότι οι χρήστες επιλέγουν διαδρομή με βάση το λογιστικό μοντέλο επιλογής και μπορούν να αντιληφθούν το μέγεθος της κυκλοφοριακής συμφόρησης, κάτι που υπονοεί ότι η μορφή του λογιστικού μοντέλου επιλογής μεταβάλλεται όσο μεταβάλλονται και οι κυκλοφοριακές συνθήκες. Περισσότερες πληροφορίες σχετικά με αυτή την περίπτωση υπάρχουν στην Ενότητα 7.5.2.

γ) Τα αποτελέσματα που έχει η χρήση των συστημάτων πλοήγησης σε δυναμικά δίκτυα, υπό τις συνθήκες που επικρατούν σήμερα. Για το σκοπό αυτό θεωρείται ότι όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης, τα οποία χρησιμοποιούν την αυτόνομη δρομολόγηση με μη ενημερωμένα στοιχεία σχετικά με τις κυκλοφοριακές συνθήκες. Έτσι οι χρήστες ακολουθούν τις οδηγίες του συστήματος πλοήγησης και σε κάποιο σημείο, στο οποίο αντιλαμβάνονται ότι η προτεινόμενη διαδρομή δεν είναι η βέλτιστη, ακολουθούν μία άλλη διαδρομή σύμφωνα με την κρίση τους. Για την προσομοίωση της παραπάνω κατάστασης θεωρείται ότι το δίκτυο φορτίζεται σε τέσσερα στάδια. Αρχικά το 60% των χρηστών ακολουθεί την προτεινόμενη διαδρομή από το σύστημα πλοήγησης, το 30% τη δεύτερη καλύτερη εναλλακτική και το 10% την τρίτη καλύτερη εναλλακτική. Έπειτα από το πρώτο στάδιο φόρτισης οι κυκλοφοριακές συνθήκες μεταβάλλονται και τα ποσοστά αυτά τροποποιούνται ως :

50%, 40%, 10%. Έπειτα ως 40%, 50%, 10% και κατά το τέταρτο στάδιο της φόρτισης ως 30%, 50%, 20%. Σταθόπουλος (2010).

δ) Τα αποτελέσματα που έχει η χρήση των συστημάτων πλοήγησης σε δυναμικά δίκτυα χρησιμοποιώντας την κεντρική, προβλεπτική δρομολόγηση που έχει προταθεί. Αυτός ο τύπος δρομολόγησης είναι πιλοτικός και απαιτεί τη συνεργασία των συστημάτων πλοήγησης και του κεντρικού διαχειριστή. Μέσω αυτής της μεθόδου ο κεντρικός διαχειριστής υπολογίζει τη συντομότερη διαδρομή για κάθε μεταφορικό μέσο ξεχωριστά υπό τις παρούσες κυκλοφοριακές συνθήκες και αναβαθμίζει τα στοιχεία του. Περισσότερες πληροφορίες σχετικά με αυτή την περίπτωση υπάρχουν στις Ενότητες 7.4.3.3.,7.5.3.

ε) Τα αποτελέσματα που έχει η χρήση των συστημάτων πλοήγησης σε δυναμικά δίκτυα χρησιμοποιώντας την κεντρική, προβλεπτική δρομολόγηση που έχει προταθεί με διαφορετικό προσανατολισμό. Πιο συγκεκριμένα ο κεντρικός διαχειριστής δεν προτείνει πλέον τη συντομότερη διαδρομή σε κάθε μεταφορικό μέσο ξεχωριστά με βάση της υπάρχουσες συνθήκες. Αντίθετα προτείνει διαδρομές σε κάθε μέσο με στόχο την ισορροπία του συστήματος και αναβαθμίζει τα δεδομένα του σχετικά με την κυκλοφοριακή κατάσταση του δικτύου. Περισσότερες πληροφορίες σχετικά με αυτή την περίπτωση υπάρχουν στις Ενότητες 7.4.3.3., 7.5.4.

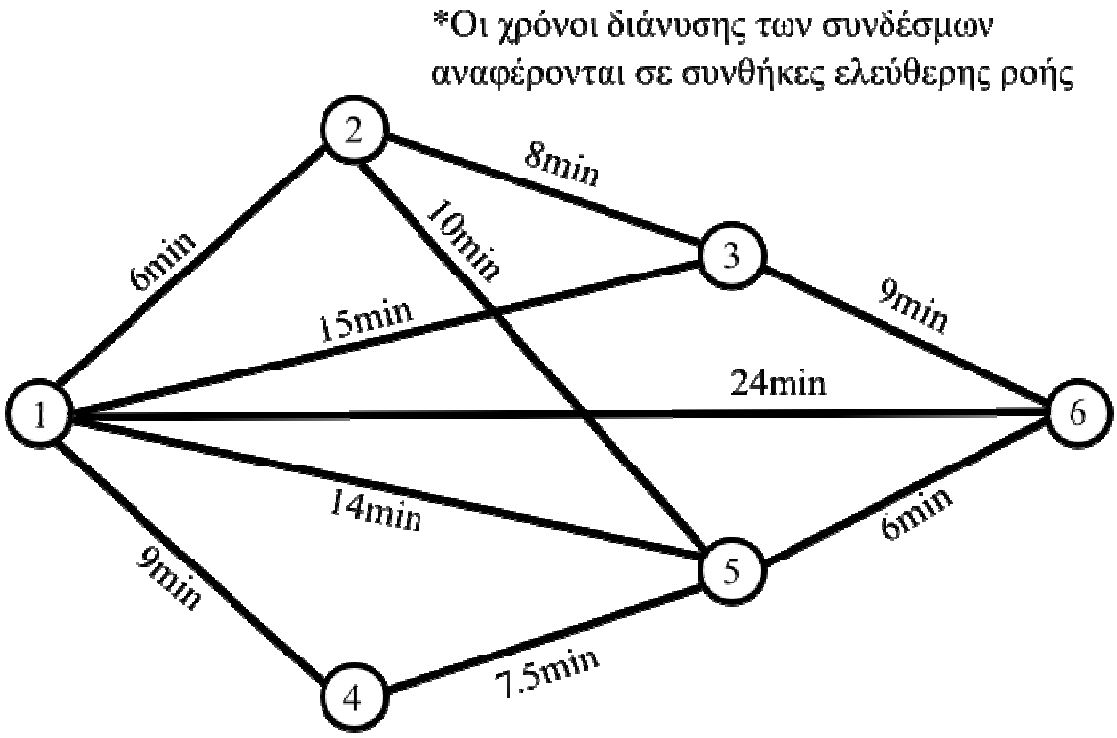
Για την υλοποίηση των εφαρμογών σε συγκοινωνιακά δίκτυα έχουν προγραμματιστεί πέντε αλγόριθμοι οι οποίοι παρουσιάζονται στο Παράρτημα της εργασίας. Το κρίσιμο σημείο που ελέγχεται είναι ο συνολικός χρόνος διάνυσης όλων των μεταφορικών μέσων που φορτίζουν το δίκτυο μέχρι την ολοκλήρωση του καταμερισμού της ζήτησης. Εάν το δίκτυο φορτίζεται με k_i ,i=1,..,z μεταφορικά μέσα με κόμβο προέλευσης r_i και κόμβο προορισμού s_i ο συνολικός χρόνος διάνυσης είναι :

$$\sum_{i=1,m} \sum_{r_i}^{s_i} w_i(e)\delta(e)$$

όπου δ(e) = 1 εάν ο σύνδεσμος e ανήκει στη διαδρομή $r_i \rightarrow s_i$, αλλιώς δ(e) = 0. Επίσης $w_i(e)$ είναι ο χρόνος διάνυσης του συνδέσμου e, όταν εισέρχεται σε αυτόν το i=1,2,..,m μεταφορικό μέσο.

Ελέγχεται επίσης και η κατάσταση του συγκοινωνιακού δικτύου μετά τον καταμερισμό της ζήτησης και πιο συγκεκριμένα οι χρόνοι διάνυσης και η φόρτιση των συνδέσμων. Αυτός ο έλεγχος πραγματοποιείται διότι μπορεί ένας τύπος δρομολόγησης να οδηγεί στο μικρότερο συνολικό χρόνο διάνυσης αλλά να χρησιμοποιεί το δίκτυο μη αποδοτικά. Έτσι εάν σε επόμενο βήμα το δίκτυο φορτιστεί περαιτέρω ο συνολικός χρόνος διάνυσης μέσω αυτής της δρομολόγησης θα αυξηθεί έντονα λόγω του φαινομένου της κυκλοφοριακής συμφόρησης. Για το σκοπό αυτό υπολογίζεται και ο μέσος χρόνος διάνυσης κάθε διαδρομής, όπως έχει προκύψει έπειτα από τον καταμερισμό της ζήτησης.

Η πρώτη εφαρμογή υλοποιείται για λόγους επίδειξης σε ένα μικρό δίκτυο, το οποίο παρουσιάζεται στο ακόλουθο σχήμα :



Σχήμα 8.2.Συγκοινωνιακό δίκτυο εφαρμογής

Θεωρείται για λόγους απλότητας ότι το δίκτυο αρχικά παραμένει αφόρτιστο. Επίσης όλες οι μετακινήσεις προέρχονται από τον κόμβο 1 και το μητρώο προέλευσης-προορισμού είναι :

Από \ Προς	2	3	4	5	6	→Κόμβοι Προορισμού
1	72	67	74	82	112	→Μεταφορικά Μέσα / Χρόνο

Θεωρείται επίσης ο χρόνος διάνυσης των συνδέσμων συνδέεται με το φόρτο μέσω της συνάρτησης BPR, ως : $t_a(q_a) = t_a^0$ [1+ 0,15 $(q_a / 100)^4$].

Στη συνέχεια παρουσιάζεται ο καταμερισμός της ζήτησης στο δίκτυο σύμφωνα με τις πέντε περιπτώσεις που αναπτύχθηκαν παραπάνω :

α) $\sum_{i=1,m}\sum_{r_i}^{s_i} w_i(e)\delta(e)$=5424 min ,

$\sum$ Χρόνος Διάνυσης Διαδρομής × Φόρτιση = 5424 min

Διαδρομή	Χρόνος Διάνυσης	Φόρτιση	Σύνδεσμος	Χρόνος Διάνυσης	Φόρτιση
1-2	6	72	1-2	6	139.
1-2-3	14	67	1-3	15	0.
1-3	15	0	1-4	9	74.
1-4	9	74	1-5	14	194.
1-4-5	16.5	0	1-6	24	0.
1-5	14	82	2-3	8	67.
1-2-5	16	0	2-5	10	0.
1-6	24	0	3-6	9	0.
1-3-6	24	0	4-5	7.5	0.
1-2-3-6	23	0	5-6	6	112.
1-5-6	20	112	Μέσος Χρόνος Διάνυσης	10.85	Συνολική Φόρτιση = 586
1-2-5-6	22	0			
1-4-5-6	22.5	0			

Πίνακας 8.9.Καταμερισμός της ζήτησης με χρήση συστημάτων πλοήγησης σε στατικά δίκτυα

β) $\sum$ Χρόνος Διάνυσης Διαδρομής × Φόρτιση = 6827,588 min

Διαδρομή	Χρόνος Διάνυσης	Φόρτιση	Σύνδεσμος	Χρόνος Διάνυσης	Φόρτιση
1-2	10.23913	72	1-2	10.23913	147.3189
1-2-3	18.31783	33.768116	1-3	15.13787	49.753365
1-3	15.13787	33.2319	1-4	11.1676	112.5671
1-4	11.1676	74	1-5	14.14865	51.58045
1-4-5	18.69249	24.51059	1-6	24.036	31.61393
1-5	14.14865	32.775	2-3	8.0787	50.6045
1-2-5	20.27343	24.7145	2-5	10.0343	38.8805
1-6	24.036	31.61393	3-6	9.016716	33.3579
1-3-6	24.1546	16.5215	4-5	7.52489	38.5672
1-2-3-6	27.33455	16.8364	5-6	6.044023	47.0282
1-5-6	20.192673	18.80553	Μέσος Χρόνος Διάνυσης	11.5428	Συνολική Φόρτιση = 601.272
1-2-5-6	26.31745	14.166			
1-4-5-6	24.7365	14.0566			

Πίνακας 8.10.Στοχαστικός καταμερισμός της ζήτησης σε δυναμικά δίκτυα, όταν ένα μικρό ποσοστό των χρηστών διαθέτει συστήματα πλοήγησης και οι χρήστες αντιλαμβάνονται τις κυκλοφορικές συνθήκες που επικρατούν στο δίκτυο κάθε χρονική στιγμή

γ) $\sum$ Χρόνος Διάνυσης Διαδρομής × Φόρτιση = 7250,203 min

Διαδρομή	Χρόνος Διάνυσης	Φόρτιση	Σύνδεσμος	Χρόνος Διάνυσης	Φόρτιση
1-2	12.1377	72	1-2	12.1377	161,6
1-2-3	20.15	31.825	1-3	15.0344	35,175
1-3	15.0344	35.175	1-4	11.5211	116.9
1-4	11.5211	74	1-5	15.2198	87.3
1-4-5	19.0592	16.4	1-6	24	5.6
1-5	15.2198	36.9	2-3	8.0123	31.825
1-2-5	22.3098	28.7	2-5	10.1721	58.2
1-6	24	5.6	3-6	9	0
1-3-6	24.0344	0	4-5	7.5381	42.9
1-2-3-6	29.15	0	5-6	7.1535	106.4
1-5-6	22.3733	50.4	Μέσος		Συνολική
1-2-5-6	29.4633	33.6	Χρόνος	11.9789	Φόρτιση
1-4-5-6	26.2127	22.4	Διάνυσης		= 645.9

Πίνακας 8.11.Καταμερισμός της ζήτησης σε δυναμικά δίκτυα, όταν τα συστήματα πλοήγησης χρησιμοποιούν την αυτόνομη δρομολόγηση

δ) $\sum_{i=1,m} \sum_{r_i}^{s_i} w_i(e)\delta(e)$=5861.5522 min ,

$\sum$ Χρόνος Διάνυσης Διαδρομής × Φόρτιση = 6511,88 min

Διαδρομή	Χρόνος Διάνυσης	Φόρτιση	Σύνδεσμος	Χρόνος Διάνυσης	Φόρτιση
1-2	7.8048	72	1-2	7.8048	119
1-2-3	15.817	32	1-3	15.05205	39
1-3	15.05205	35	1-4	10.29684	99
1-4	10.29684	74	1-5	17.80234	116
1-4-5	17.801	0	1-6	24.0481	34
1-5	17.80234	82	2-3	8.0126	32
1-2-5	17.8056	0	2-5	10.0008	15
1-6	24.0481	34	3-6	9.	4
1-3-6	24.0521	4	4-5	7.5044	25
1-2-3-6	24.8174	0	5-6	6.2699	74
1-5-6	24.07224	34	Μέσος		Συνολική
1-2-5-6	24.0755	15	Χρόνος	11.57917	Φόρτιση
1-4-5-6	24.07114	25	Διάνυσης		= 557

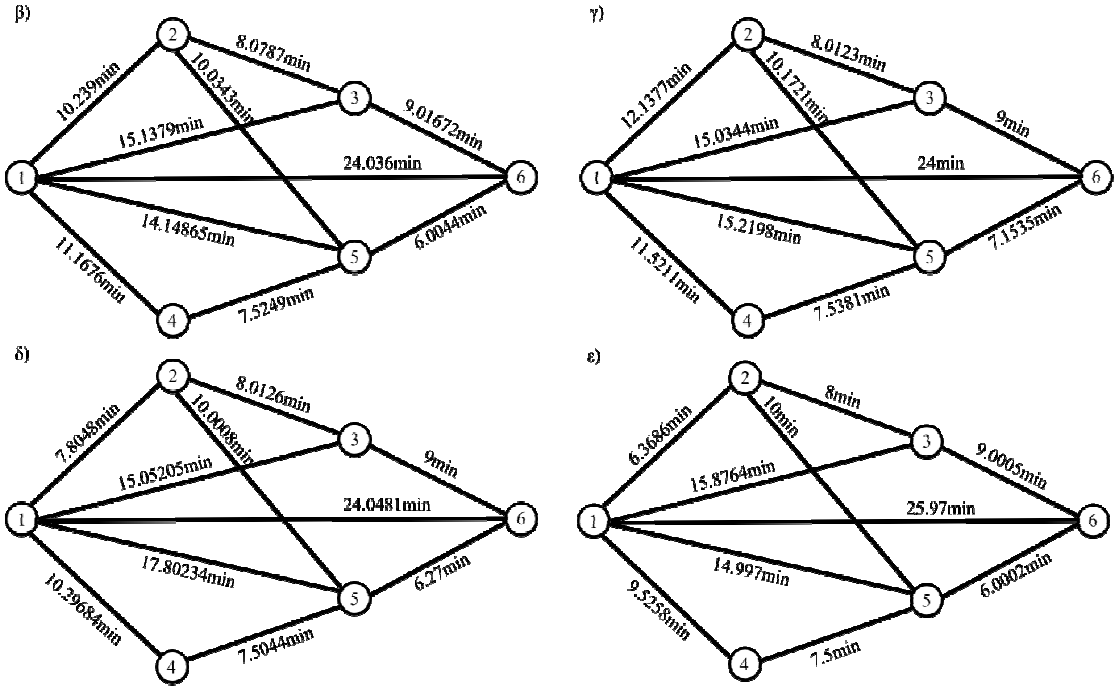
Πίνακας 8.12.Καταμερισμός της ζήτησης σε δυναμικά δίκτυα, όταν χρησιμοποιείται η κεντρική, προβλεπτική δρομολόγηση και ο κεντρικός διαχειριστής προτείνει ξεχωριστά σε κάθε μεταφορικό μέσο μία διαδρομή σύμφωνα με την ισορροπία του χρήστη αναβαθμίζοντας κάθε φορά τα δεδομένα του

ε) $\sum_{i=1,m}\sum_{r_i}^{s_i} w_i(e)\delta(e)$=5989.62 min ,

$\sum$ Χρόνος Διάνυσης Διαδρομής × Φόρτιση = 6305,942 min

Διαδρομή	Χρόνος Διάνυσης	Φόρτιση	Σύνδεσμος	Χρόνος Διάνυσης	Φόρτιση
1-2	6.3686	72	1-2	6.3686	80
1-2-3	14.3686	0	1-3	15.8764	79
1-3	15.8764	67	1-4	9.5258	79
1-4	9.5258	74	1-5	14.997	83
1-4-5	17.0258	2	1-6	25.97	86
1-5	14.997	77	2-3	8.	2
1-2-5	16.3686	3	2-5	10.	6
1-6	25.97	86	3-6	9.0005	14
1-3-6	24.8769	12	4-5	7.5	5
1-2-3-6	23.3691	2	5-6	6.0002	12
1-5-6	20.9972	6	Μέσος		Συνολική
1-2-5-6	22.3688	3	Χρόνος	11.32374	Φόρτιση
1-4-5-6	23.026	3	Διάνυσης		= 446

Πίνακας 8.13.Καταμερισμός της ζήτησης σε δυναμικά δίκτυα, όταν χρησιμοποιείται η κεντρική, προβλεπτική δρομολόγηση και ο κεντρικός διαχειριστής προτείνει ξεχωριστά σε κάθε μεταφορικό μέσο μία διαδρομή σύμφωνα με την ισορροπία του συστήματος αναβαθμίζοντας κάθε φορά τα δεδομένα του



Σχήμα 8.3.Χρόνοι Διάνυσης των συνδέσμων έπειτα από τον καταμερισμό της ζήτησης σύμφωνα με τις τέσσερις περιπτώσεις που εξετάζονται

Έπειτα από τα αποτελέσματα της εφαρμογής, κατά τις τέσσερις διαφορετικές μορφές καταμερισμού της ζήτησης που προκύπτουν, συγκεντρώνονται σε ένα συγκριτικό πίνακα τα στοιχεία : Συνολικός χρόνος κυκλοφορίας των μεταφορικών μέσων μέχρι στιγμής, Μέσος χρόνος υλοποίησης κάθε μετακίνησης μετά τον καταμερισμό της ζήτησης, Μέσος χρόνος διάνυσης συνδέσμου μετά τον καταμερισμό της ζήτησης.

		α	**β**	**γ**	**δ**	**ε**
Συνολικός χρόνος κυκλοφορίας των μεταφορικών μέσων μέχρι στιγμής (min)		5424.	-	-	5861.5522	5989.62
Μέσος Χρόνος Υλοποίησης Κάθε Μετακίνησης Μετά τον καταμερισμό της ζήτησης	1-2	6	10.23913	12.1377	7.8048	6.3686
	1-3	14	16.72785	17.5922	15.4345	15.1225
	1-4	9	11.1676	11.5211	10.2968	9.5258
	1-5	14	17.70486	18.86293	17.803	16.1305
	1-6	20	24.46196	25.51233	24.0638	23.44067
Μέσος χρόνος διάνυσης συνδέσμου μετά τον καταμερισμό της ζήτησης (min)		10.85	11.5428	11.9789	11.57917	11.32374

Πίνακας 8.14.Συγκεντρωτικά στοιχεία σύγκρισης των μεθόδων δρομολόγησης μέσω της εφαρμογής

Παρατηρήσεις

1) Όπως είναι φυσικό τα συστήματα πλοήγησης είναι αναγκαία σε στατικά δίκτυα και μπορούν να οδηγήσουν στους μικρότερους δυνατούς χρόνους κυκλοφορίας για τα μέσα μεταφοράς. Έτσι εξοικονομούνται χρηματικοί και ενεργειακοί πόροι, καθώς αποφεύγεται η επιλογή εσφαλμένων εναλλακτικών λόγω ελλειπούς γνώσης του δικτύου και της αδυναμίας απαρίθμησης όλων των εναλλακτικών επιλογών.

2) Η δρομολόγηση με στόχο την ισορροπία του συστήματος αναγκάζει ορισμένους χρήστες να διανύσουν μεγαλύτερη απόσταση από την ελάχιστη δυνατή. Για το λόγο αυτό άλλωστε οι εναλλακτικές διαδρομές που μπορούν να χρησιμοποιηθούν για την ίδια μετακίνηση έχουν έντονες διαφορές στο χρόνο διάνυσής τους. Πρέπει όμως να αναφερθεί ότι η διαφορά στο συνολικό χρόνο κυκλοφορίας των μέσων μεταφοράς κατά την περίπτωση που τους προτείνεται συνεχώς η συντομότερη διαδρομή με βάση τις παρούσες συνθήκες και κατά την περίπτωση που τους προτείνεται μία διαδρομή με στόχο την ισορροπία του συστήματος δεν είναι μεγάλη : (5861.5522 min και 5989.62 min αντίστοιχα). Η διαφορά αυτή εξετάζεται στη συνέχεια σε μεγαλύτερες εφαρμογές για την εξαγωγή ασφαλέστερων συμπερασμάτων. Πλεονεκτήματα κατά την ισορροπία ως προς το σύστημα παρουσιάζονται στην τελική κατάσταση του δικτύου, η οποία είναι καλύτερη από πλευράς κυκλοφοριακής συμφόρησης. Αυτό είναι εμφανές από τη φόρτιση των συνδέσμων (Μέγιστη φόρτιση = 86 μετφ.μέσα, ενώ στην ισορροπία ως προς το χρήστη = 119 μετφ.μέσα). Ως αποτέλεσμα οι χρόνοι διάνυσης των συνδέσμων παραμένουν σε χαμηλά επίπεδα και δε δημιουργούνται

συνθήκες αιχμής σε μεμονωμένα τμήματα του δικτύου. Το στοιχείο αυτό έχει μεγάλη σημασία καθώς εάν ο κεντρικός διαχειριστής συνεχίσει να προτείνει συντομότερες διαδρομές το δίκτυο δε χρησιμοποιείται αποδοτικά και μπορεί τελικά ο συνολικός χρόνος διάνυσης να είναι μεγαλύτερος σε σχέση με τη δρομολόγηση που στοχεύει στην ισορροπία του συστήματος. Τέλος πρέπει να αναφερθεί ότι οι χρήστες πρέπει να πειστούν ώστε να ακολουθήσουν μη αποδοτικές διαδρομές που στοχεύουν στην ισορροπία του συστήματος. Για παράδειγμα μπορεί να προταθεί σε ένα χρήστη διαδρομή με χρόνο διάνυσης 25.792min για τη μετακίνηση 1→6 τη στιγμή που η μετακίνηση αυτή μπορεί να υλοποιηθεί και με χρόνο διάνυσης 20.9972min. Πολλά εξαρτώνται δηλαδή από τη συνεργασία τους.

3) Η χρήση των συστημάτων πλοήγησης με αυτόνομη δρομολόγηση σε δυναμικά δίκτυα οδηγεί σε δυσμενέστερες συνθήκες κυκλοφοριακής συμφόρησης σε σύγκριση ακόμα και με την περίπτωση που δε χρησιμοποιούνται συστήματα πλοήγησης. Αυτό προκύπτει ακόμα και αν θεωρείται ότι όσο περισσότερο διαφέρει η διαδρομή που προτείνεται από τη βέλτιστη, τόσο αυξάνεται το ποσοστό των χρηστών που ακολουθούν άλλες εναλλακτικές. Ακόμα και σε αυτήν τη μικρή εφαρμογή προκύπτει ότι οι συνολικοί χρόνοι κυκλοφορίας των μεταφορικών μέσων είναι αυξημένοι. Επίσης σε ορισμένα τμήματα του δικτύου εμφανίζεται έντονη κυκλοφοριακή συμφόριση (τμήμα 1-2 : 162 μετ.μέσα) ενώ σε άλλα τμήματα η κυκλοφοριακή ροή είναι πολύ χαμηλή. Εμφανίζονται δηλαδή έντονες ανισοκατανομές στον καταμερισμό της ζήτησης. Ως αποτέλεσμα των παραπάνω οι χρόνοι διάνυσης των συνδέσμων παρουσιάζονται αυξημένοι (κατά μέσο όρο 11.9789min και 11.5428min αντίστοιχα όταν δε χρησιμοποιούνται συστήματα πλοήγησης). Τέλος δημιουργούνται έντονα προβλήματα τοπικής κυκλοφοριακής συμφόρησης σε τμήματα των διαδρομών που προτείνονται από τα συστήματα πλοήγησης επειδή πολλοί χρήστες ακολουθούν την ίδια διαδρομή. Το φαινόμενο αυτό έχει επισημανθεί σε πολλές έρευνες και παρατηρείται ακόμα και σε περιπτώσεις που τα κυκλοφοριακά δεδομένα αναβαθμίζονται συνεχώς.

Στη συνέχεια παρουσιάζονται αποτελέσματα και από άλλες, μεγαλύτερες εφαρμογές.

Εφαρμογές σε τυχαία δίκτυα

[$\sum_{e_\epsilon E} w(e)$] / L : Μέσος όρος χρόνων διάνυσης των συνδέσμων μετά τον καταμερισμό της ζήτησης (min)

$\sum_{i=1,m} \sum_{r_i}^{s_i} w_i(e)\delta(e)$:

Συνολικός χρόνος κυκλοφορίας των μεταφορικών μέσων (min)

Κόμβοι	Σύνδεσμοι	α	β	γ
22	56	5.977 51968	10.7348 _	14.9662 _
16	40	5.9075 25495	7.877909 _	9.39008 _
6	20	8.89 20163	13.936 _	15.904 _

Πίνακας 8.15.Αποτελέσματα εφαρμογών στην περίπτωση που χρησιμοποιούνται συστήματα πλοήγησης με αυτόνομη δρομολόγηση (γ) και στην περίπτωση που δε χρησιμοποιούνται (β)

[$\sum_{e_\epsilon E} w(e)$] / L : Μέσος όρος χρόνων διάνυσης των συνδέσμων μετά τον καταμερισμό της ζήτησης (min)

$\sum_{i=1,m} \sum_{r_i}^{s_i} w_i(e)\delta(e)$:

Συνολικός χρόνος κυκλοφορίας των μεταφορικών μέσων (min)

Κόμβοι	Σύνδεσμοι	δ	ε
47	130	6.46713 9109	6.4486 12011
36	100	6.702 6777	6.69217 8239
30	80	6.61594 4298	6.60245 5200
26	68	6.2502 3430	6.23363 4298
22	56	5.9929 2348	5.9796 2790
16	40	5.91042 1256	5.9089 1361

Πίνακας 8.16.Αποτελέσματα εφαρμογών στην περίπτωση που η κεντρική δρομολόγηση στοχεύει στην ισορροπία του χρήστη (δ) ή στην ισορροπία του συστήματος (ε)

Από τις παραπάνω εφαρμογές η επιλογή διαδρομών από τον κεντρικό διαχειριστή με στόχο την ισορροπία του χρήστη δίνει καλύτερα αποτελέσματα σε σύγκριση με την επιλογή διαδρομών με στόχο την ισορροπία του συστήματος. Μάλιστα ο συνολικός χρόνος διάνυσης διαφέρει έως και (2902 min) κατά τις δύο περιπτώσεις (9109 min, 12011min αντίστοιχα). Τέλος, ως προς το χρόνο εκτέλεσης, ο κεντρικός διαχειριστής εξυπηρέτησε ταυτόχρονα 348 αιτήσεις για συντομότερη διαδρομή μία προς μία, αναβαθμίζοντας κάθε φορά τα δεδομένα του σε χρόνο 0,256 sec.

ΚΕΦΑΛΑΙΟ 9ο

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΙΣΗΓΗΣΕΙΣ ΓΙΑ ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ

9.1.ΣΥΜΠΕΡΑΣΜΑΤΑ ΣΥΜΦΩΝΑ ΜΕ ΤΟΥΣ ΑΡΧΙΚΟΥΣ ΣΤΟΧΟΥΣ ΠΟΥ ΕΘΕΣΕ Η ΕΡΕΥΝΑ

Για τον υπολογισμό της βέλτιστης διαδρομής από ένα σύστημα πλοήγησης προτείνεται ο αλγόριθμος Dijkstra με βελτιώσεις στις δομές δεδομένων. Είναι πολύ σημαντικό όμως να τροποποιηθεί η αναπαράσταση του δικτύου με χρήση κάποιου αλγόριθμου, όπως αυτού που παρουσιάστηκε στην Ενότητα 5.8., αλλιώς ο χρόνος εκτέλεσής του αυξάνεται σημαντικά και τυχόν βελτιώσεις στις δομές δεδομένων δε φέρουν ουσιαστικά αποτελέσματα. Τα παραπάνω επιβεβαιώνονται και σε θεωρητικό επίπεδο καθώς η δομή δεδομένων που προτάθηκε από τους Fredman and Tarjan (1987) για τον αλγόριθμο Dijkstra επιλύει το πρόβλημα υπολογισμού της βέλτιστης διαδρομής σε χρόνο $O(\mathrm{E+N}\times\log\mathrm{N})$ που αποτελεί το θεωρητικό όριο μέχρι σήμερα όταν τα βάρη των συνδέσμων είναι θετικοί πραγματικοί αριθμοί. Επίσης ευρετικοί αλγόριθμοι όπως ο A* και ο διπλής κατεύθυνσης δε μειώνουν δραστικά το εύρος αναζήτησης στα συγκοινωνιακά δίκτυα και ως αποτέλεσμα δε βελτιώνουν το χρόνο εκτέλεσης. Τέλος δεν είναι δυνατό να χρησιμοποιηθούν από τα συστήματα πλοήγησης αλγόριθμοι εύρεσης όλων των εναλλακτικών διαδρομών για μία μετακίνηση λόγω του αυξημένου χρόνου εκτέλεσης. Εξαίρεση αποτελεί η περίπτωση υπολογισμού μόνο των κυριότερων εναλλακτικών διαδρομών με χρήση του αλγορίθμου Eppstein (1997) που έχει σχεδόν ίδιο χρόνο εκτέλεσης με τον αλγόριθμο Dijkstra, όπως αποδείχθηκε μέσω των εφαρμογών.

Οι νέες εύχρηστες μέθοδοι που έχουν προταθεί για τον υπολογισμό των εκπομπών αεριών που ευθύνονται για το φαινόμενο του θερμοκηπίου, όπως το αποτύπωμα άνθρακα, μπορούν να χρησιμοποιηθούν από τα συστήματα πλοήγησης ώστε να περιοριστούν οι περιβαλλοντικές επιπτώσεις που οφείλονται στις μεταφορές. Για το σκοπό αυτό τα συστήματα πλοήγησης πρέπει να προτείνουν διαδρομές στους χρήστες που είναι ταυτόχρονα σύντομες και οδηγούν σε περιορισμένες εκπομπές CO_2. Για να υπολογιστούν όμως αυτές οι διαδρομές υπάρχει υπολογιστική επιβάρυνση. Αποδείχθηκε ωστόσο μέσω εφαρμογών σε τυχαία συγκοινωνιακά δίκτυα ότι το πρόβλημα εύρεσης της συντομότερης διαδρομής που ικανοποιεί ταυτόχρονα έναν περιορισμό μπορεί να επιλυθεί με τη μέθοδο των πολλαπλασιαστών Lagrange και το απαιτούμενο υπολογιστικό κόστος είναι περίπου επταπλάσιο σε σύγκριση με τον αλγόριθμο Dijkstra. Έτσι ακόμα και σε συγκοινωνιακά δίκτυα με πυκνή δόμηση ο απαιτούμενος χρόνος εκτέλεσης $\approx O(11\times(\mathrm{N+E})log_2 N))$ είναι αποδεκτός.

Όσο το ποσοστό των χρηστών που διαθέτει συστήματα πλοήγησης παραμένει σε χαμηλά επίπεδα δε μπορεί να επηρεαστεί ο καταμερισμός της ζήτησης στα συγκοινωνιακά δίκτυα. Οφέλη μπορεί να προκύψουν μόνο για τους χρήστες που διαθέτουν συστήματα πλοήγησης. Πρέπει ωστόσο να διαχωριστούν δύο περιπτώσεις. Εάν τα συστήματα πλοήγησης ακολουθούν την αυτόνομη δρομολόγηση και χρησιμοποιούν στοιχεία που δεν αντιπροσωπεύουν την τρέχουσα κυκλοφοριακή κατάσταση στο δίκτυο, τότε προτείνουν διαδρομές που δεν είναι βέλτιστες και οι κάτοχοί τους ζημιώνονται καθώς ακολουθούν μη αποδοτικές διαδρομές. Εάν όμως ακολουθούν την αποκεντρωτική δρομολόγηση και λαμβάνουν συνεχώς αναβαθμισμένα στοιχεία σχετικά με τις κυκλοφοριακές συνθήκες που επικρατούν στο δίκτυο, τότε υπολογίζουν διαδρομές που προσεγγίζουν τις βέλτιστες.

Εάν αυξηθεί το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης η αποδοτικότητα του δικτύου επιδεινώνεται, εκτός κι αν χρησιμοποιηθούν νέες μέθοδοι δρομολόγησης. Πιο συγκεκριμένα εάν χρησιμοποιείται η αυτόνομη δρομολόγηση η κατάσταση στο δίκτυο επιδεινώνεται καθώς σε όλους τους χρήστες προτείνονται παρόμοιες διαδρομές για την ίδια μετακίνηση και ορισμένοι σύνδεσμοι φορτίζονται υπερβολικά. Ακόμα και αν αρκετοί χρήστες αγνοήσουν τις διαδρομές που προτείνονται από τα συστήματα πλοήγησης, ορισμένες διαδρομές αντιμετωπίζουν προβλήματα κυκλοφοριακής συμφόρησης και ο συνολικός χρόνος μετακινήσεων είναι μεγαλύτερος ακόμα και από την περίπτωση που κανένας χρήστης δε χρησιμοποιεί σύστημα πλοήγησης. Η κατάσταση βελτιώνεται εάν χρησιμοποιηθεί η αποκεντρωτική δρομολόγηση και αναβαθμίζονται συνεχώς τα στοιχεία των κυκλοφοριακών συνθηκών. Ωστόσο, επειδή τα ίδια κυκλοφοριακά δεδομένα διανέμονται σε όλα τα συστήματα πλοήγησης, προτείνονται ξανά οι ίδιες διαδρομές σε όλους τους χρήστες και η ζήτηση καταμερίζεται τελικά στο δίκτυο σύμφωνα με τη μέθοδο του όλα ή τίποτα μέχρι την επόμενη αναβάθμιση των κυκλοφοριακών στοιχείων. Τα παραπάνω συμπεράσματα που προκύπτουν από τις εφαρμογές επιβεβαιώνονται και από άλλες έρευνες σχετικά με τη χρήση των συστημάτων πλοήγησης Kaufmann et al. (1991), Wunderlich and Smith (1992) στις οποίες προτείνεται το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης να μην ξεπερνά το 30%. Το πρόβλημα επιδείνωσης της αποδοτικότητας του δικτύου μπορεί να περιοριστεί με χρήση νέων αλγορίθμων οι οποίοι προτείνουν στους χρήστες διαδρομές που ικανοποιούν διάφορους περιορισμούς σχετικά με το σκοπό της μετακίνησής τους. Έτσι, δε θα καθοδηγούνται όλα τα μεταφορικά μέσα με στόχο τη μείωση του χρόνου μετακίνησης και ο καταμερισμός της ζήτησης θα βελτιωθεί. Μία ακόμα λύση είναι τα συστήματα πλοήγησης να ενημερώνουν για σημεία του δικτύου στα οποία επικρατούν έκτακτες συνθήκες (πορείες, έργα κ.α.) χωρίς να προτείνουν διαδρομή στο χρήστη ώστε να μην επηρεάζουν τον καταμερισμό της ζήτησης. Τέλος, ο μόνος τρόπος για τη βελτίωση της απόδοσης του δικτύου όταν αυξηθεί το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης είναι χρησιμοποίηση της κεντρικής δρομολόγησης, στην οποία ο κεντρικός διαχειριστής έχει επίγνωση των κυκλοφοριακών συνθηκών που επικρατούν και επιλέγει τις διαδρομές που θα ακολουθήσουν οι χρήστες.

9.2.ΕΙΣΗΓΗΣΕΙΣ ΓΙΑ ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ

- Έρευνα για τη χρήση των συστημάτων πλοήγησης ως συστήματα εντοπισμού θέσης των μεταφορικών μέσων. Σκοπός είναι η χρήση τους ως συστήματα μέτρησης της κυκλοφοριακής ροής. Η τεχνική αυτή πλεονεκτεί διότι η παρακολούθηση της κίνησης των μεταφορικών μέσων είναι συνεχής σε αντίθεση με τη χρήση άλλων μεθόδων μέτρησης όπου το μεταφορικό μέσο πρέπει να διέλθει από ένα σταθερό σημείο ώστε να προσμετρηθεί.

- Έρευνα σχετικά με τις εκπομπές CO_2 των μεταφορικών μέσων ανάλογα με το είδος του οδικού τμήματος, την κλίση του, τη μέση ταχύτητα, την κατανάλωση καυσίμου κ.α. Σκοπός είναι ο προσεγγιστικός υπολογισμός των εκπομπών CO_2 κατά τη διάνυση οδικών τμημάτων. Έτσι μπορούν να υπολογιστούν από τα συστήματα πλοήγησης διαδρομές οι οποίες αποτελούνται από επιμέρους οδικά τμήματα, στα οποία οι εκπομπές CO_2 κατά τη διάνυσή τους δεν είναι σημαντικές.

- Έρευνα για νέες μεθόδους με αποδεκτούς χρόνους εκτέλεσης οι οποίες εστιάζουν στην πρόταση διαδρομής σε κάθε χρήστη η οποία να πληροί τις προσωπικές του προτιμήσεις. Διερεύνηση κυρίως του καταμερισμού της ζήτησης όταν η χρήση των συστημάτων πλοήγησης είναι μαζική και σε κάθε χρήστη προτείνεται διαφορετική διαδρομή σύμφωνα με τους περιορισμούς που έχει θέσει. Στόχος αυτής της διερεύνησης είναι η μελέτη της επιβάρυνσης στην αποδοτικότητα των συγκοινωνιακών δικτύων λόγω αυτής της συνθήκης.

- Έρευνα με σκοπό την τροποποίηση της αποκεντρωτικής δρομολόγησης ώστε να μην επιδεινώνεται η αποδοτικότητα των συγκοινωνιακών δικτύων όταν αυξηθεί το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης. Μία ενδιαφέρουσα πρόταση είναι να τροποποιηθούν τα αναβαθμισμένα κυκλοφοριακά δεδομένα που διανέμονται στα συστήματα πλοήγησης. Για το λόγο αυτό, κάθε σύστημα πλοήγησης μπορεί να προσθέτει έναν τυχαίο συντελεστή σφάλματος στα δεδομένα. Έτσι οι διαδρομές που προτείνουν τα συστήματα πλοήγησης για την ίδια μετακίνησης θα διαφέρουν. Αυτή η προσέγγιση βέβαια δεν οδηγεί στο βέλτιστο καταμερισμό της ζήτησης, αλλά μπορεί να βελτιώσει την αποδοτικότητα του δικτύου ακόμα και όταν το ποσοστό των χρηστών που διαθέτουν συστήματα πλοήγησης αυξηθεί.

- Έρευνα με σκοπό την εφαρμογή της κεντρικής δρομολόγησης. Ιδιαίτερα πρέπει να μελετηθούν τα αποτελέσματα αυτού του τύπου δρομολόγησης σε πραγματικά δίκτυα και τα πρακτικά προβλήματα που πρέπει να ξεπεραστούν ώστε να εφαρμοστεί. Τα πλεονεκτήματα αυτού του τύπου δρομολόγησης είναι πολλαπλά και ενδέχεται να βελτιώσει αισθητά τον καταμερισμό της ζήτησης στα συγκοινωνιακά δίκτυα, με μεγάλα οφέλη στους τομείς της οικονομίας και του περιβάλλοντος. Πρέπει να σημειωθεί επίσης ότι είναι ο μόνος τύπος

δρομολόγησης που μπορεί να διαχειριστεί τη μεταφορική ζήτηση οδηγώντας σε ισορροπία του συστήματος. Για να συμβεί αυτό το ποσοστό των χρηστών που διαθέτει συστήματα πλοήγησης πρέπει να είναι μεγάλο, ωστόσο δε θα απαιτείται από αυτά να έχουν σημαντικές υπολογιστικές δυνατότητες καθώς όλοι οι υπολογισμοί πραγματοποιούνται από τον κεντρικό διαχειριστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΞΕΝΟΓΛΩΣΣΕΣ

Ahuga, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E. (1990) *¨Faster Algorithms for the Shortest Path Problem¨*. Journal of the Association of Computing Machinery, Vol. 37, No. 2, pp. 213-223.

Allsop, R.E., Charlesworth, J.A. (1977) *¨Traffic in a Signal-Controlled Road Network: An Example of Different Signal Timings Inducing Different Routeings¨*. Traffic Engineering and Control 18(5), pp. 262-265.

Alon, N., Galil, Z., Margalit, O. (1997) *¨On the Exponent of the All Pairs Shortest Path Problem¨*. Journal of Computing and System Sciences (54), pp. 255–262.

Andreatta, G., Romeo, L. (1988) *¨Stochastic shortest path problems with recource¨*. Networks (18), pp. 193–204.

Avella, P., Boccia, M., Sforza, A. (2004) *¨Resource Constrained Shortest Path Problems in Path Planning for Fleet Management¨*. Journal of Mathematic Modeling Algorithms, Vol. 3, pp. 1-17.

Banhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P. (1998) *¨Branch-and-price: Column generation for solving huge integer problems¨*. Operations Research, 46(3), pp. 316-329.

Beasley, J., Christofides., N. (1989) *¨An Algorithm for the Resource Constrained Shortest Path Problem¨*. Networks (19), pp. 379–394.

Beckmann, M., McGuire, C.B., Winsten, C.B. (1956) *¨Studies In the Economics of Transportation ¨*. New Haven, Connecticut: Yale University Press.

Bellman, R. (1958) *¨On a Routing Problem¨*. Quarterly of applied mathematics, Vol. 16, p. 87.

Bellman, R., Kalaba, R. (1960) *¨On Kth best policies¨*. Journal of SIAM, Vol. 8, No. 4, pp. 582-588.

Bellman, R.E. (2003) *¨Dynamic Programming¨*. New York: Dover Publications.

Ben-Akiva, M.E., Bierlaine, M., Bottom, J., Koutsopoulos, H.N., Mishalani, R. (1997) *¨Development of a Route Guidance Generation System for Real-Time Application¨*. Proceedings of the 8th IFAC/IFIP/IFORS symposium on transportation systems, Chania, Greece.

Ben-Akiva, M.E., Bierlaine, M., Koutsopoulos, H.N., Mishalani, R. (1998) *¨DynaMIT-A simulation based system for traffic prediction¨*. Paper presented at the DACCORD short term forecasting Workshop, Delft, The Netherlandes.

Bertsekas, D. (1993) *¨A Simple and Fast Label Correcting Algorithm for Shortest Paths¨*. Networks, Vol. 23, pp. 703-709.

Borndorfer, R., Grotschelm, M., Lobel, A. (2001) *¨Scheduling duties by adaptive column generation¨*. Technischer Bericht, ZIB-Report #01-01, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, Berlin.

Boyce, D.E.(1989) *¨Route Guidance Systems for managing urban transportation networks: Review and prospects¨*. Presented at tenth Italian Regional Science Conference, Rome.

Braess, D. (1968) *¨Uber ein Paradox aus der Verkehrsplanung¨*. Unternehmensforschung 12, pp. 258-268.

Chabini, I. (1998) *¨Discrete Dynamic Shortest Path Problems in Transportation Applications: Complexity and Algorithms with Optimal Run Time¨*. Transportation Research Record 1645, pp. 170-175.

Cherkassky, B.V., Goldberg, A.V., Silverstein, C. (1997) *¨Buckets, Heaps, Lists and monotone priority queues¨.* SIAM Journal on Computing 28(4), pp. 1326–1346.

Cho, H.J., Lan, C.L. (2009) *¨Hybrid shortest path algorithm for vehicle navigation¨*. The Journal of Supercomputing (49), pp. 234-247.

Cormen, T., Leiserson, C., Rivest, R., Stein C. (2001) *¨Introduction to Algorithms¨*. 2nd Edition, Massachusettes: MIT Press, pp. 23-57.

Croucher, J.S. (1978) *¨A Note of stochastic shortest-route problem¨*. Naval Research Logistics Quarterly (25), pp. 729–732.

Crowder, H. (1976) *¨Computational Improvements for subgradient optimization¨.* In Symposia mathematica*,* Vol. 19, London: Academic press.

Dafermos, S.C., Sparrow, F.T. (1969) *¨The traffic assignment problem for a general network¨.* Journal of Research of the National Bureau of Standards, Series B, 73B(2), pp. 91–118.

Dafermos, S.C. (1980) *¨Traffic Equilibrium and variational inequalities¨.* Transportation Science 14 (1), pp. 42–54.

Daganzo, C.F., Sheffi, Y. (1977) *¨On Stochastic models of traffic assignment¨*. Transportation Science 11(3), pp. 253-274.

Daganzo, C.F. (1978) *¨On the traffic assignment problem with flow dependent costs-II¨.* Transportation Research (11), pp. 439-441.

Damberg, O., Lundgren, J.T., Patriksson, M. (1996) *¨An algorithm for the stochastic user equilibrium problem¨.* Transportation Research 30B, pp. 115-131.

Dantzig, G.B. (1960) *¨On the shortest route through a network¨.* Management Science 6 (2), pp. 187 - 190.

Dantzig, G.B., Wolfe P. (1960) *¨Decomposition Principle for Linear Programs¨*. Operations Research (8), pp. 101-111.

Desaulniers, G., Desrosiers, G., Ioachim, J., Solomon, I., Soumis, F., Villeneuve, D. (1988) *¨A Unified Framework for deterministic time constrained vehicle routing and crew scheduling problems¨*. Norwel, MA: Kluwer Publishing Company, Fleet Management and Logistics, pp. 57-93.

Desrochers, M. (1988) *¨An algorithm for the shortest path problem with Resource Constraints¨.* Technical Report G-88-27, GERAD.

Desrochers, M., Desrosiers, J., Solomon M. (1992) *¨A new optimization algorithm for the vehicle routing problem with time windows¨*. Operations Research (40), pp. 342-354.

Desrosiers, J., Dumas, Y., Solomon, M.M, Soumis, F. (1995) *¨Time Constrained Routing and Scheduling¨.* Handbooks in OR & MS, pp. 35–139.

Dial, R.B. (1971) *¨A Probabilistic Multipath Traffic Assignment Algorithm which obviates path enumeration¨*. Transportation Research 5(2), pp. 83-111.

Dial, R.B., Glover, F., Karney, D., Klingman D. (1979) ¨*A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees*¨. Networks (9), pp. 215–248.

Dijkstra, E.W. (1959) ¨*A Note on Two Problems in Connexion with Graphs*¨. Numerische Mathematik 1, pp. 269-271.

Dreyfus, S. (1969) ¨*An appraisal of some shortest paths algorithms*¨. Operations Research, Vol. 17, No. 3, pp. 395-412.

Dumitresu, I., Boland, N. (2003) ¨*Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem*¨. Networks (42), pp. 135-153.

Eppstein, D. (1998) ¨*Finding the k shortest paths*¨. SIAM Journal of Computing 28(2), pp. 652–673.

Euler, L. (1736) ¨*Solutio problematis ad geometriam situs pertinentis*¨, Commentarii Academiae Scientiarum Imperialis Petropolitanae 8, pp. 128–140.

Fan, Y., Kalaba, R., Moore, J. (2005) ¨*Arriving on Time*¨. Journal of Optimization Theory and Applications, No.3, pp. 497-513.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C. (2004) ¨*An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems*¨. Networks (44), pp. 216-229.

Fiduccia, C., Mattheyses, R. (1982) ¨A Linear time heuristic for proving network partitions¨. Proceedings of the 19th IEEE Design Automation Conference, Las Vegas, Nevada, pp. 175–181.

Fisk, C. (1979) ¨*More paradoxes in the equilibrium assignment problem*¨. Transportation Research (13)B, pp. 305–309.

Fisk, C. (1980) ¨*Some Developments in equilibrium traffic assignment*¨. Transportation Research (14)B, pp. 243–255.

Floyd, R.W. (1962) ¨*Algorithm 97: Shortest Path*¨. Communications of the Association for Computing Machinery 5, p. 345.

Ford, L.R., Fulkerson D.R. (1962) ¨*Flows in Networks*¨. Princeton: Princeton University Press.

Fox, B.L. (1975) ¨*K-th shortest paths and applications to the probabilistic networks*¨. ORSA/TIMS Joint National Mtg., Vol. 23, p. 263.

Frank, M., Wolfe, P. (1956) ¨*An algorithm for quadratic programming*¨. Naval Research Logistics Quarterly (3), pp. 95-110.

Frank, H. (1969) ¨*Shortest paths in probabilistic graphs*¨. Operations Research 17.

Frederickson, G.N. (1983) ¨*Implicit Data Structures For The Dictionary Problem*¨. Journal of the ACM 30, pp. 80-94.

Fredman, M., Tarjan, R. (1987) ¨*Fibonacci heaps and their uses in improved network optimization algorithms*¨. Journal of the ACM 34, pp. 596-615.

Fredman, M., Willard, D. (1993) ¨*Surpassing the information theoretic bound with fusion trees*¨. Journal of Computing and System Sciences (47), pp. 424–436.

Fredman, M., Willard, D. (1994) ¨*Trans-Dichotomous algorithms for minimum spanning trees and shortest paths*¨. Journal of Computing and System Sciences (48), pp. 533–551.

Friesz, T.L., Bernstein, D., Smith, T.E., Tobin, R.L., Wie, B.W. (1993) *¨A Variational inequality formulation of the dynamic network user equilibrium problem¨*. Operations Research (41), pp. 179-191.

Frigioni, D., Marchetti-Spaccamela, A., Nani, U. (1996) *¨Fully Dynamic Output Bounded Single Source Shortest Path Problem¨*. In proceedings ACM-SIAM Symposium on Discrete algorithms, pp. 212-221.

Fu, L., Rillet, L.R. (1997) *¨Expected shortest paths in dynamic and stochastic traffic networks¨*. Transportation Research Part B (32), pp. 499-516.

Gallo, G., Pallottino S. (1986) *¨Shortest Path Methods: A Unified Approach¨*. Mathematical Programming Study, Vol. 26, pp. 38-64.

Gallo, G., Pallottino, S. (1988) *¨Shortest Path Algorithms¨*. Annals of Operations Research, Vol. 7, pp. 3–79.

GAP, SEI, Eco-Logica (2006) *¨UK Schools Carbon Footprint Scoping Study¨*. Report by Global Action Plan, Stockholm Environment Institute and Eco-Logica Ltd for the Sustainable Development Commission, London.

Ghali, M., Smith, M. (1995) *¨A model for the dynamic system optimum traffic assignment problem¨*. Transportation Research 29B, pp. 155-170.

Glover, F., Glover, R., Klingman, R. (1986) *¨The Threshold Shortest Path Algorithm¨*. Networks, Vol. 14, No. 1.

Goldberg, A.V., Kaplan, H., Werneck, R.F. (2005) *¨Reach for A*: Efficient Point To Point shortest path algorithms¨*. Proceedings of the 8th Workshop on Algorithm Engineering and Experiments, SIAM, Philadelphia, pp. 129–143.

Golden, B. (1976). *¨Shortest-Path Algorithms: A Comparison¨*. Operations Research (24), pp. 1164-1168.

Greenshields, B.D.A. (1935) *¨Study of Traffic Capacity¨*. Highway Research Board Proceedings 14, pp. 448-477.

Han, Y., Thorup, M. (2002) *¨Integer sorting in O(n$\sqrt{loglogn}$) expected time and linear space¨*. In Proc. of 43rd FOCS, pp. 135–144.

Handler, G., Zang, I. (1980) *¨A Dual Algorithm for the Constrained Shortest Path Problem¨*. Networks (10), pp. 293-310.

Hart, P. E., Nilsson, N.J., Raphael B. (1968) *¨A Formal Basis for the Heuristic Determination of Minimum Cost Paths¨*. IEEE Transportation Systems Sci. and Cybernetics SSC-4(2), pp. 100-107.

Hartel, P.H., Glaser H. (1996) *¨The resource constrained shortest path problem implemented in a lazy functional language¨*. Journal of Functional Programming 6(1), pp. 29-45.

Hearn, D.W., Lawphongpanich, S., Ventura, J.A. (1987) *¨Restricted simplicial decomposition: computation and extensions¨*. Mathematical Programming Study (31), pp. 99-118.

Held, M.H., Wolfe, P., Crowder, H.D. (1974) *¨Validation of subgradient optimization¨*. Mathematic programming, Vol.6, No.1., pp. 62-88.

Heydecker, B.G. (1986) *¨On the definition of traffic equilibrium¨*. Transportation Research 20B(6), pp. 435-440.

Hoffman, W., Pavley, R. (1959) *¨A method for the solution of the N'th best path problem¨*. Journal of the Association for Computing Machinery (6), pp. 506–514.

Irnich, S., Desaulniers, G. (2004) *¨Shortest path problems with resource constraints¨.* Column Generation (2), pp. 33–65.

Jahn, O., Mohring, R.H., Schulz, A.,S. (1999) *¨Optimal Routing of Traffic Flows with Length Restrictions in Networks with Congestion¨.* Technical Report #658, Technische Universitt Berlin.

Janson, B.N. (1991) *¨Dynamic traffic assignment for urban road networks¨.* Transportation Research B 25, pp. 143-161.

Johnson, D.B. (1977) *¨Efficient algorithms for shortest paths in sparse networks¨.* Journal of the ACM (JACM), 24(1), pp. 1–13.

Johnson, D.B. (1988) *¨A priority queue in which initialization and queue operations take O(loglogD) time¨.* Theory of Computing Systems, Vol. 15, No. 1, pp. 295-309.

Juttner, A., Szviatovski, B., Mecs, I., Rajko, Z. (2001) *¨Lagrange relaxation based method for the qos routing problem¨*. INFOCOM, Vol. 2, pp. 859-868.

Katoh, N., Ibaraki, T., Mine, H. (1982) *¨An efficient algorithm for the k shortest simple paths¨*. Networks 12(4), pp. 411-427.

Kaufman, D.E., Smith, R.L., Wunderlich, K.E. (1991) *¨An Iterative Routing / Assignment Method for Anticipatory Real-Time Route Guidance¨*. SAE Vehicle Navigation and Information systems Conference Proceedings, P-253, pp. 701-707.

Kaufman, D.E., Smith, R.L. (1993) *¨Fastest paths in time dependent networks for intelligent vehicle-highway systems application¨.* Transportation Research B 29B, pp. 95-98.

Kernighan, B.W., Lin, S. (1970) *¨An efficient heuristic procedure for partitioning graphs¨*. The Bell System Technical Journal (49), pp. 201-307.

Korilis, Y.A., Lazar, A.A., Orda, A. (1997) *¨Capacity allocation under noncooperative routing¨.* IEEE Trans. Automat. Cont. 42 (3), pp. 309–325.

Korilis, Y.A., Lazar, A.A., Orda, A. (1999) *¨Avoiding the Braess paradox in noncooperative network¨*. Journal of Applied Probability 36 (1), pp. 211–222.

Koutsoupias, E., Papadimitriou, C. (1999) *¨Worst-case equilibria¨.* In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS), pp. 404–413.

Lawler, E.L. (1972) *¨A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem¨.* Management Science (18), pp. 401–405.

LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P. (1975) *¨An Efficient Approach to solving the Network Equilibrium Traffic Assignment Problem¨*. Transportation Research (9), pp. 309-318.

LeBlanc, L.J., Helgason, R.V., Boyce, D.E. (1985) *¨Improved Efficiency of the Frank-Wolfe algorithm for convex network programs¨*. Transportation Science (19), pp. 445-462.

LeBlanc, R., Boyce, D.E. (1986) *¨A bi-level programming algorithm for exact solution of the network design problem with user-optimal flows¨.* Transportation Research 20B(3), pp. 259-265.

Lee, C.K. (1994) *¨A Multiple-Path Routing Strategy for Vehicle Route Guidance Systems¨*. Transportation Research : C, Vol. 2, No.3, pp. 185-195.

Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E. (1998) *¨Deviation Algorithms for ranking shortest paths¨*. The International Journal of Foundations of Computer Science 10 (3), pp.247-263.

Martins, E.Q.V., Pascoal, M.M.B. (2003) *¨A new implementation of Yen's ranking loopless paths algorithm¨*. Technical report, Centro de Informatica e Sistemas.

Mavronicolas, M., Spirakis, P. (2001) *¨The price of selfish routing¨*. In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), Hersonissos, Crete, pp. 510–519.

McCarthy, P.S. (2001) *¨Transportation Economics, Theory and practice: a case study¨*. Massachusetts: Blackwell Publishers, p. 10.

Mehlhorn, K., Ziegelmann, M. (2000) *¨Resource Constrained Shortest Paths¨*. In 7th Annual European Symposium on Algorithms (ESA2000), LNCS 1879, pp. 326–337.

Mehlhorn, K., Ziegelmann, M. (2001) *¨CNOP – A Package for Constrained Network Optimization¨*. Algorithm Engineering and Experimentation: Third International Workshop (ALENEX 2001), Lecture Notes in Computer Science, Vol. 2153.

Mirchandani, P.B. (1976) *¨Shortest distance and reliability of probabilistic networks¨*. Comp. Oper. Res. (3), pp. 347-355.

Mofat, A., Takaoka, T. (1987) *¨An all pairs shortest path algorithm with expected time O(n2log n)¨*. SIAM Journal of Computing (16), pp. 1023-1031.

Moore, E.F. (1957) *¨The shortest path through a maze¨*. Proceedings of an International Symposium on the Theory of Switching, Annals of the Computation laboratory of Harvard University 30, pp. 285-292.

Murchland, J.D. (1970) *¨Braess's paradox of traffic flow¨* .Transport. Res. (4), pp. 391–394.

Nicholson, T.A. (1966) *¨Finding the Shortest Route between two points in a network¨*. Computer Journal, Vol. 9, pp. 276-280.

Nikolova, E., Brand, M., Karger, D.R. (2006) *¨Optimal root planning under uncertainty¨*. In Proceedings of the International Conference on Automated Planning & Scheduling (ICAPS), Lake District, England, pp. 131–141.

Oppenheim, N. (1995) *¨Urban Travel Demand Modeling¨*. A Wiley-Interscience Publication, John Willey and Sons, Inc.

Papadimitriou, C.H., Yannakakis, M. (1991) *¨Shortest paths without a map¨*. In 16th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 372, pp. 610-620.

Pape., U. (1974) *¨Implementation and Efficiency of Moore Algorithms for the Shortest Path Problem¨*. Mathematical Programming 7, pp. 212–222.

Patriksson, M. (1994) *¨The Traffic Assignment Problem: Models and Methods¨*. Utrecht, The Netherlands: VSP.

Pollack, M., Wiebenson, W. (1960) *¨Solution of the shortest route problem-a review¨*. Operations Research, Vol. 8, No. 2, pp. 187-190.

Pollack, M. (1961) *¨The Kth best route through a network¨*. Operations Research, Vol. 9, No. 4 (1961), p. 578.

Polychronopoulos, G.H., Tsitsiklis, J.N. (1996) *¨Stochastic shortest path problems with recource¨*. Networks (27), pp. 133-143.

Potts, R.B., Oliver, R.M. (1972) *¨Flows in Transportation Network¨*. Academic Press, New York.

Ramalingam, G., Reps, T. (1996) *¨An Incremental Algorithm for the generalization of the Shortest Path Problem¨*. Journal of Algorithms (21), pp. 267–305.

Raman, R. (1996) *¨Priority Queues: Small, Monotone and Trans-Dichotomous¨*. In Proc. 4th Annual European Symposium Algorithms, pp. 121–137.

Raman, R. (1997) *¨Recent results on the single-source shortest paths problem¨*. SIGACT News 28(2), pp. 81–87.

Ran, B., Boyce, D.E., LeBlanc, L.J. (1993) *¨A New class of instantaneous dynamic useroptimal traffic assignment models¨*. Operations Research (41), pp. 192-202.

Ran, B., Boyce, D.E. (1994) *¨Dynamic Urban Transportation Network Models-Theory and Implication for Intelligent Vehicle-Highway Systems¨*, In Lecture Notes in Economics and Mathematical Systems, Vol. 417, Springer-Verlag, New York.

Righini, G., Salani, M. (2005) *¨New dynamic programming algorithms for the resource-constrained elementary shortest path problem¨*. Technical Report #69, Universit degli Studi di Milano, submitted to Networks.

Roughgarden, T., Tardos, E. (2001) *¨How bad is selfish routing ?¨*. Journal of the ACM, Vol. 49, pp. 236-259.

Roughgarden, T. (2004) *¨Stackelberg Scheduling Strategies¨*. SIAM Journal of Computing, Vol. 33, pp. 332-350.

Roy B. (1959) *¨Transitivité et connexité¨*. Comptes Rendus de l'Academie des Sciences 249, pp. 216–218.

Sanders, P., Schultes, D. (2005) *¨Fast and Exact Shortest Path Queries Using Highway Hierarchies¨*. In 13th European Symposium on Algorithms, Vol. 3669 of LNCS, pp. 568–579.

Schulz, A.S., Stier-Moses, N.E. (2003) *¨On the performance of user equilibria in traffic networks¨*. Proc.14th Annual ACM-SIAM Symposium on Discrete algorithms (SODA). SIAM, Philadelphia, pp. 86-87.

SEI, WWF, CURE (2006) *¨Counting consumption - CO_2 emissions, material flows and ecological footprint of the UK by region and devolved country¨*. Published by WWF-UK, Godalming, Surrey, UK.

Seidel, R. (1992) *¨On the All-Pairs-Shortest Path problem¨*. Proceedings of the 24rd Annual ACM Symposium on Theory of Computing, Victoria, Canada, pp. 745–749.

Sheffi, Y. (1985) *¨Urban Transportation Networks, Equilibrium Analysis with Mathematical Programming Methods¨*. Englewood Cliffs, N.J.: Prentice-Hall.

Shor N.Z. (1985) *¨Minimization Methods for Non-defferentiable Functions¨*. Berlin: Springer series in computational mathematics.

Takaoka, T. (2005) *¨An O(n^3loglogn/logn) time algorithm for the all-pairs shortest path problem¨*. Information Processing Letters, Vol. 96, pp. 155-161.

Texas Transport Institute (2002) *¨Urban Mobility Study¨*

Thorup, M. (1996) *¨On RAM priority queues¨*. In Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms, pp. 59-67.

Thorup, M. (2003) *¨Integer priority queues with decrease key in constant time and the single source shortest paths problem¨*. In *Proc. of 35th STOC*, pp. 149-158.

Transportation Research Board (2000) *¨Highway Capacity Manual 2000¨*. National Research Council, Washigton D.C.

U.S. Department of Transportation (1998) *¨The National Intelligence Transportation Systems Architecture¨*. Ver 2.0., Washigton DC.

Wardrop, J.G. (1952) *¨Some Theoretical Aspects of Road Traffic Research¨*. Proc. of the Institution of Civil Engineers (ICE), Part II, Road Engineering Division, pp. 325-362.

Warshall, S. (1962) *¨A theorem on Boolean matrices¨*. Journal of ACM, 9 (1), pp. 11-12.

Wie, B.W., Tobin, R., Bernstein, D., Friesz, T. (1995) *¨A Comparison of system optimal and user optimal equilibrium dynamic traffic assignments with scheduled delays¨*. Transportation Research 3C, pp. 389-411.

Wiedmann, T., Minx, J. (2007) *¨A definition of "carbon footprint" ¨*. ISAUK Research Report #07-01.

Winkler, P.M. (1983) *¨Proof of the Squashed Cube Conjecture¨*. Vol. 3, No. 1, pp. 135-139.

Wunderlich, K.E., Smith, R.L. (1992) *¨Large Scale Traffic Modeling for Route-Guidance Evaluation: A Case Study¨*. University of Michigan IVHS Program Technical Report #92-08, Ann Arbor, Michigan.

Wunderlich, K., Kaufman, D., Smith, R.L. (1997) *¨Link travel time prediction techniques for convergent iterative anticipatory route guidance methods¨*. Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor.

Yen, J.Y. (1971) *¨Finding the K shortest loopless paths in a network¨*. Management Science (17), pp. 712–716.

Yen, J.Y. (1972) *¨Another algorithm for finding the K shortest-loopless network paths¨*. In Proc. of 41st Mtg. Operations Research Society of America, Vol. 20, p. 185.

Zhao, X., Luh, P.B., Wang, J. (1999) *¨Surrogate Subgradient Algorithm for Lagrangian Relaxation¨*. Journal of optimization theory and applications, Vol.100, No.3., pp. 699-712.

ΕΛΛΗΝΙΚΕΣ

Γεωργακόπουλος, Γ.Φ. (2008) *¨Δομές Δεδομένων - έννοιες τεχνικές και αλγόριθμοι¨*. Ηράκλειο: Πανεπιστημιακές Εκδόσεις Κρήτης, σελ. 135-162, 307-319.

Κάβουρας, Ι.Κ. (2003) *¨Προγραμματισμός με Java¨*. Αθήνα: Εκδόσεις Κλειδάριθμος, σελ.13-46.

Καρλαύτης, Μ.Γ., Λαγαρός, Ν.Δ. (2010) *¨Επιχειρησιακή Έρευνα και Βελτιστοποίηση για Μηχανικούς¨*. Αθήνα: Εκδόσεις Συμμετρία.

Σταθόπουλος, Α., Καρλαύτης, Μ. (2008) *¨Σχεδιασμός Μεταφορικών Συστημάτων¨*. Αθήνα: Εκδόσεις Παπασωτηρίου, σελ. 5-30, 84-88, 123-152.

Σταθόπουλος, Α. (2010) *¨Προσωπική επαφή¨*.

Φραντζεσκάκης, Ι.Μ., Πιτσιάβα-Λατινοπούλου, Μ.Χ., Τσαμπούλας, Δ.Α. (1997) *¨Διαχείριση Κυκλοφορίας¨*. Αθήνα: Εκδόσεις Παπασωτηρίου, σελ. 6-12, 269-277.

Φραντζεσκάκης, Ι.Μ., Γκόλιας, Ι.Κ., Πιτσιάβα-Λατινοπούλου Μ.Χ. (1997) *¨Κυκλοφοριακή Τεχνική¨*. Αθήνα: Εκδόσεις Παπασωτηρίου, σελ. 2-70, 91-211.

ΠΑΡΑΡΤΗΜΑ

Στη συνέχεια παρουσιάζεται η ονομασία των κυριότερων μεταβλητών που έχουν χρησιμοποιηθεί στα επιμέρους προγράμματα.

e : Αριθμός συνδέσμου.

i : Αριθμός κόμβου.

N, L : Συνολικός αριθμός κόμβων και συνδέσμων του συγκοινωνιακού δικτύου αντίστοιχα.

S(e), E(e) : Κόμβος προέλευσης και κόμβος προορισμού του συνδέσμου e ϵ L.

w(e) : Αντίσταση κατά τη διάνυση του κόμβου e ϵ L.

r, t : Κόμβοι προέλευσης και προορισμού μιας διαδρομής.

D(i) : Συντομότερη απόσταση κάθε κόμβου i ϵ N από τον αρχικό r.

SP(e ή i): Μονοδιάστατη μήτρα για την αποθήκευση των κόμβων ή των συνδέσμων από τους οποίους αποτελείται μία διαδρομή ελαχίστου κόστους.

u : Ο επόμενος πλησιέστερος κόμβος στον κόμβο προέλευσης σε κάθε επανάληψη.

Ακολούθως παρουσιάζονται σε γλώσσα προγραμματισμού Fortran 95 οι αλγόριθμοι που χρησιμοποιήθηκαν στις εφαρμογές του Κεφαλαίου 8., σύμφωνα με τη σειρά εξετάστηκαν.

Αλγόριθμος Dijkstra με Λίστα Αναμονής
Επίλυση : Single Source Shortest Paths Problem

```
      MODULE Dijkstra_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      METRHTHS=0
      u=r
      NDUF=0
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do k=1,L  !*RELAXATION STEP
        if(S(k).eq.u.and.D(E(k)).gt.D(S(k))+w(k))then
              D(E(k))=D(S(k))+w(k)
              Q(E(k))=D(E(k))
        endif
        enddo
      ShortestDistance=1.2E+38
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        NDIF=1
        endif
        enddo
      IF(NDIF.EQ.0)GO TO 115
        NDUF=NDUF+1
      Q(u)=0
      METRHTHS=METRHTHS+1
      NDIF=0
```

```
      ENDDO
  115 CONTINUE
      write(*,*)NDUF
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      END Subroutine DijkstraAlgorithm
      END MODULE

      PROGRAM Dijkstra
      use Dijkstra_Mod
      IMPLICIT NONE
      INTEGER N,L,r,t,i
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\4000 to 80000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D)
      Call DijkstraAlgorithm(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM Dijkstra
```

Αλγόριθμος Dijkstra με χρήση δυαδικού σωρού
Επίλυση : Single Source Shortest Paths Problem

```
      module Dijkstra_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N))
      Do i=1,N
      F(i)=0
      enddo
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      RETURN
      END Subroutine Initialization
```

```
*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorith(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q
      integer,dimension(N)::IL,IK
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      do i=1,N
      IL(i)=i
      IK(i)=i
      enddo
      Iz=IL(1)
      IL(1)=IL(r)
      IL(r)=Iz
      IK(r)=1
      IK(1)=r
      METRHTHS=1
      u=IL(METRHTHS)
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(IL(r))=0

      DO WHILE(METRHTHS/=N)
        do k=1,L  !*RELAXATION STEP
        if(S(k).eq.u.and.D(E(k)).gt.D(S(k))+w(k))then
         D(E(k))=D(S(k))+w(k)
              Q(IK(E(k)))=D(E(k))
              ipp=IK(E(u,k))
  124 CONTINUE
      IF(Q(ipp)<Q(ipp/2))THEN
      Iz=ipp
      IK(E(k))=Iz/2
      IK(IL(Iz/2))=Iz
      IRES=IL(Iz)
      IL(Iz)=IL(Iz/2)
      IL(Iz/2)=IRES
      H=Q(Iz/2)
      Q(Iz/2)=Q(Iz)
      Q(Iz)=H
      ipp=ipp/2
      GO TO 124
      ENDIF

      ipp=IK(E(k))
  125 CONTINUE
      if(Q(ipp)<Q(ipp-1))then
      Iz=ipp
      IK(E(k))=Iz-1
      IK(IL(Iz-1))=Iz
      IRES=IL(Iz)
      IL(Iz)=IL(Iz-1)
      IL(Iz-1)=IRES
      H=Q(Iz-1)
      Q(Iz-1)=Q(Iz)
      Q(Iz)=H
      ipp=ipp-1
      GO TO 125
      ENDIF
        endif
        enddo
```

```
      METRHTHS=METRHTHS+1
      u=IL(METRHTHS)
      ENDDO
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      END Subroutine DijkstraAlgorith
      end module

      PROGRAM Dijkstra
      use Dijkstra_Mod
      IMPLICIT NONE
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      INTEGER :: r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2800 to 48580.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(S,E,w,D)
      call DijkstraAlgorith(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      WRITE(*,*) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Dijkstra με τροποποιημένη αναπαράσταση δικτύου, Ενότητα 5.8
Επίλυση : Single Source Shortest Paths Problem

```
      MODULE Dijkstra_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N),F(L),g(N,L))
      Do i=1,N
      F(i)=0
      enddo

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
```

```
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D
      real,dimension(N)::Q
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      METRHTHS=0
      u=r
      NDUF=0
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do k=1,F(u)  !*RELAXATION STEP
        if(D(E( g(u,k ))).gt.D(S( g(u,k ))+w( g(u,k )))then
              D(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
              Q(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        NDIF=1
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      IF(NDIF.EQ.0)GO TO 115
        NDUF=NDUF+1
      Q(u)=0
      METRHTHS=METRHTHS+1
      NDIF=0
      ENDDO
  115 CONTINUE
      write(*,*)NDUF
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      do i=1,N
      write(7,*)D(i)
      enddo
      END Subroutine DijkstraAlgorithm
      END MODULE

      PROGRAM Dijkstra
      use Dijkstra_Mod
      IMPLICIT NONE
      INTEGER N,L,r,t,i
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
```

```
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\4000 to 80000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D,F,g)
      Call DijkstraAlgorithm(S,E,w,D,F,g)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM Dijkstra
```

Αλγόριθος Dijkstra με χρήση δυαδικού σωρού σε τροποποιημένο δίκτυο
Επίλυση : Single Source Shortest Paths Problem

```
      module Dijkstra_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:),F(:),g(:,:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N),F(L),g(N,L))
      Do i=1,N
      F(i)=0
      enddo
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorith(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D
      real,dimension(N)::Q
      integer,dimension(N)::IL,IK
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      do i=1,N
      IL(i)=i
      IK(i)=i
```

```
    enddo
    Iz=IL(1)
    IL(1)=IL(r)
    IL(r)=Iz
    IK(r)=1
    IK(1)=r
    METRHTHS=1
    u=IL(METRHTHS)
    do 123 i=1,N
123 Q(i)=1.2E+38
    Q(IL(r))=0

    DO WHILE(METRHTHS/=N)
      do k=1,F(u)  !*RELAXATION STEP
      if(D(E(g(u,k ))).gt.D(S(g(u,k ))+w(g(u,k )))then
       D(E(g(u,k )))=D(S(g(u,k )))+w(g(u,k ))
            Q(IK(E(g(u,k ))))=D(E(g(u,k )))
            ipp=IK(E(g(u,k )))
124 CONTINUE
    IF(Q(ipp)<Q(ipp/2))THEN
    Iz=ipp
    IK(E(g(u,k )))=Iz/2
    IK(IL(Iz/2))=Iz
    IRES=IL(Iz)
    IL(Iz)=IL(Iz/2)
    IL(Iz/2)=IRES
    H=Q(Iz/2)
    Q(Iz/2)=Q(Iz)
    Q(Iz)=H
    ipp=ipp/2
    GO TO 124
    ENDIF

    ipp=IK(E(g(u,k )))
125 CONTINUE
    if(Q(ipp)<Q(ipp-1))then
    Iz=ipp
    IK(E(g(u,k )))=Iz-1
    IK(IL(Iz-1))=Iz
    IRES=IL(Iz)
    IL(Iz)=IL(Iz-1)
    IL(Iz-1)=IRES
    H=Q(Iz-1)
    Q(Iz-1)=Q(Iz)
    Q(Iz)=H
    ipp=ipp-1
    GO TO 125
    ENDIF
      endif
      enddo
    METRHTHS=METRHTHS+1
    u=IL(METRHTHS)
    ENDDO
    WRITE(9,116)NAME1,NAME2
116 FORMAT(3X,A,10X,A)
    DO I=1,N
    WRITE(9,117)I,D(I)
117 FORMAT(3X,I4,24X,F12.3)
    enddo
    END Subroutine DijkstraAlgorith
    end module

    PROGRAM Dijkstra
    use Dijkstra_Mod
    IMPLICIT NONE
    REAL, DIMENSION(2) :: tarray
    REAL :: result
```

```
      INTEGER :: r,t
      integer, allocatable::S(:),E(:),F(:),g(:,:)
      real,allocatable::w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2800 to 48580.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(S,E,w,D,F,g)
      call DijkstraAlgorith(S,E,w,D,F,g)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      WRITE(*,*) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Dijkstra με τριχοτομημένη λίστα αναμονής σε τροποποιημένο δίκτυο
Επίλυση : Single Source Shortest Paths Problem

```
MODULE DijkstraAdvanced_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N),F(L),g(N,L))
      Do i=1,N
      F(i)=0
      enddo
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g(S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      write(*,*)r
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorith(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D
      real,dimension(L)::Q,QQ,Z,ZZ,X,XX
      integer::METRHTHS,u,rr,TRAMER
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
```

```
    METRHTHS=0
    u=SNode
    ll=0
    kk=0
    rr=0
    itr=0
    itrit=0
    TRAMER=0
    SOUNTER=0
    DO WHILE(METRHTHS/=Nodes)
      do k=1,F(u)  !*RELAXATION STEP
      if(D(E(g(u,k))).gt.D(S(g(u,k)))+w(g(u,k)
   1))then
            D(E(g(u,k)))=D(S(g(u,k)))+w(g(u,k))
             TRAMER=TRAMER+1
             SOUNTER=SOUNTER+D(E(g(u,k)))
             CoefficientA=SOUNTER/TRAMER
             if(D(E(g(u,k)))<CoefficientA*0.8)then
             ll=ll+1
             Q(ll)=D(E(g(u,k)))
             QQ(ll)=E(g(u,k))
             else if(D(E(g(u,k)))<(CoefficientA)*1.2)then
             kk=kk+1
             Z(kk)=D(E(g(u,k)))
             ZZ(kk)=E(g(u,k))
             else
             rr=rr+1
             X(rr)=D(E(g(u,k)))
             XX(rr)=E(g(u,k))
             endif
      endif
      enddo   !#RELAXATION STEP ENDS
    if(itr.eq.1)go to 138
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,ll
      if(Q(i)<ShortestDistance)then
    ShortestDistance=Q(i)
    u=QQ(i)
    iz=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    if(ShortestDistance>1.2E+37)then
    itr=1
    go to 138
    endif
    SOUNTER=SOUNTER-Q(iz)
    Q(iz)=1.2E+38
    go to 139
138 continue
    if(itrit.eq.1)go to 140
          ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,kk
      if(Z(i)<ShortestDistance)then
    ShortestDistance=Z(i)
    u=ZZ(i)
    iz=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    if(ShortestDistance>1.2E+37)then
    itrit=1
    go to 140
    endif
    SOUNTER=SOUNTER-Z(iz)
    Z(iz)=1.2E+38
    go to 139
140 continue
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
```

```
        do i=1,rr
        if(X(i)<ShortestDistance)then
      ShortestDistance=X(i)
      u=XX(i)
      iz=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      SOUNTER=SOUNTER-X(iz)
      X(iz)=1.2E+38
 139  continue
      METRHTHS=METRHTHS+1
      TRAMER=TRAMER-1
      ENDDO
      WRITE(*,*)ll,kk,rr
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      END Subroutine DijkstraAlgorith
      END MODULE

      PROGRAM DijkstraAdvanced
      use DijkstraAdvanced_Mod
      IMPLICIT NONE
      INTEGER :: Nodes,Links,SNode,ENode,i
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\D500 to 35066.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D,F,g)
      Call DijkstraAlgorith(S,E,w,D,F,g)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      WRITE(*,118) result
      END PROGRAM DijkstraAdvanced
```

Αλγόριθμος Moore-Bellman-Ford που τερματίζει εάν δεν αναβαθμιστούν τα βάρη συνδέσμων
Επίλυση : Single Source Shortest Paths Problem

```
      MODULE Moore_Mod
      CONTAINS
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
```

```
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE MooreAlgorithm(S,E,w,D)
      INTEGER :: N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:) :: S,E
      real, dimension(:) :: w,D
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      NDIF=0
      Iteration=0
      DO I=1,N
      Iteration=Iteration+1
      DO J=1,L
      IF(D(S(J))+w(J)<D(E(J))) THEN
      D(E(J))= w(J)+D(S(J))
      NDIF=NDIF+1
      ENDIF
      ENDDO
      write(*,*)NDIF,ITERATION
      IF (NDIF.EQ.0) GO TO 115
      NDIF=0
      ENDDO
  115 CONTINUE
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      END SUBROUTINE MooreAlgorithm
      END MODULE

      PROGRAM Moore
      use Moore_Mod
      IMPLICIT NONE
      INTEGER :: N,L,r,t
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\4000 to 80000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D)
      Call MooreAlgorithm(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM Moore
```

Αλγόριθμος Threshold
Επίλυση : Single Source Shortest Paths Problem

```
    module Threshold_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE ThresholdAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q,QQ
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      do i=1,N
      Q(i)=1.2E+38
      QQ(i)=1.2E+38
      enddo
      Q(r)=0
      threshold=30
  127 continue
        do k=1,N  !*RELAXATION STEP
        if(Q(k).eq.0)then
           do j=1,L
      if(S(j).eq.k.and.D(E(j)).gt.D(k)+w(j))then
              D(E(j))=D(k)+w(j)
              if(D(E(j)).gt.threshold)QQ(E(j))=0
              if(D(E(j))<threshold)Q(E(j))=0
           endif
           enddo
         Q(k)=1.2E+38
         go to 127
        endif
        enddo
  128 continue
        threshold=2*threshold
        ih=0
        do i=1,N
```

```
        if(QQ(i).eq.0.and.D(i)<threshold)then
        Q(i)=0
        QQ(i)=1.2E+38
        ih=1
        endif
        enddo
        if(ih.eq.1)go to 127
        if(threshold<1.2E+38)go to 128
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      END Subroutine ThresholdAlgorithm
      end module

      PROGRAM Threshold
      use Threshold_Mod
      IMPLICIT NONE
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      INTEGER N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\1000 to 10010.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(S,E,w,D)
      call ThresholdAlgorithm(S,E,w,D)
      CALL ETIME(tarray,result)
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Astar
Επίλυση : Point to Point Path Problem

```
      module Astar_Mod
      contains
*!    Apost : Η προβλεπόμενη απόσταση μεταξύ κάθε κόμβου και του κόμβου προορισμού
      SUBROUTINE Initialization(S,E,w,D,Apost)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:),Apost(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N),Apost(N))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
```

```
       Do i=N,1,-1
       READ(7,*)Apost(i)
       enddo
      CLOSE(7)
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE AstarAlgorithm(S,E,w,D,Apost)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D,Apost
      real,dimension(N)::Q
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      u=r
      NDIF=0
      do i=1,N
      Q(i)=1.2E+38
      enddo
      Q(r)=0
      DO While(u/=t)
      do i=1,L
           if(S(i).eq.u.and.D(E(i))>D(S(i))+w(i))then
                D(E(i))=D(S(i))+w(i)
           endif
      enddo
      ShortestDist=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)+Apost(i)<ShortestDist) then
      ShortestDist=D(i)+Apost(i)
      u=i
      endif
      enddo
                NDIF=NDIF+1
      Q(u)=0
      ENDDO
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      WRITE(9,*)u,D(u)
      write(*,*)NDIF
      END Subroutine AstarAlgorithm
      end module

      PROGRAM Astar
      use Astar_Mod
      IMPLICIT NONE
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      INTEGER :: r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:),Apost(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2800 to 48580.f')
      OPEN(7,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\PPSPP\
     1AstarInput.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\PPSPP\
     1PPSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(S,E,w,D,Apost)
      call AstarAlgorithm(S,E,w,D,Apost)
```

```
      CALL ETIME(tarray, result)
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Διπλής Κατεύθυνσης (Bidirectional) με βέλτιστη αναπαράσταση δικτύου Επίλυση : Point to Point Path Problem

```
module Bidirectional_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D,D2,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:),F(:),g(:,:)
      real,allocatable::w(:),D(:),D2(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N),D2(N),F(L),g(N,L))
      Do i=1,N
      F(i)=0
      enddo

      Do I=1,L
       READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO

      DO 114 I=1,N
      D(I)=1.2E+38
      D2(I)=1.2E+38
  114 continue
      D(r)=0
      D2(t)=0
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE BidnalAlgorithm(S,E,w,D,D2,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D,D2
      real,dimension(N)::Q,QQ
      integer::u,uu
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      u=r
      uu=t
      do 123 i=1,N
      Q(i)=1.2E+38
      QQ(i)= 1.2E+38
  123 continue
```

```
    Q(r)=0
    QQ(t)=0
    NDIF=0
120 continue
      do k=1,F(u)  !*RELAXATION STEP
    if(D(E(g(u,k))).gt.D(S(g(u,k)))+w(g(u,k)))then
      D(E(g(u,k)))=D(S(g(u,k)))+w(g(u,k))
            Q(E(g(u,k)))=D(E(g(u,k)))
      endif
      enddo
      do k=1,F(uu)
      if(D2(S(g(uu,k))).gt.D2(E(g(uu,k)))+w(g(uu,k)))then
      D2(S(g(uu,k)))=D2(E(g(uu,k)))+w(g(uu,k))
            QQ(S(g(u,k)))=D2(S(g(u,k)))
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance1=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
    ShortestDistance2=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance1)then
    ShortestDistance1=D(i)
    u=i
      endif
      if(QQ(i)/=0.and.D2(i)<ShortestDistance2)then
    ShortestDistance2=D2(i)
    uu=i
        endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    QQ(uu)=0
    NDIF=NDIF+1
    do i=1,N
    if (Q(i).eq.0.and.uu.eq.i.or.QQ(i).eq.0.and.u.eq.i)go to 119
    enddo
    go to 120
119 continue
    write(*,*)D(uu)+D2(uu)
    WRITE(9,*)D(uu)+D2(uu)
    write(*,*)NDIF
    END Subroutine BidnalAlgorithm
    end module

    PROGRAM Bidirectional
    use Bidirectional_Mod
    IMPLICIT NONE
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    INTEGER :: r,t
    integer, allocatable::S(:),E(:),F(:),g(:,:)
    real,allocatable::w(:),D(:),D2(:)
    OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
   1ks\4000 to 80000.f')
    OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\PPSPP\
   1PPSPPOutput.f')
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    CALL Initialization(S,E,w,D,D2,F,g)
    call BidnalAlgorithm(S,E,w,D,D2,F,g)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    END
```

Αλγόριθμος : Τροποποιημένος Dijkstra

```
      MODULE Dijkstra_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N),F(L),g(N,L))
      Do i=1,N
      F(i)=0
      enddo

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D
      real,dimension(N)::Q
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'
      METRHTHS=0
      u=r
      NDUF=0
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N.and.u.ne.t)
        do k=1,F(u)  !*RELAXATION STEP
        if(D(E( g(u,k ))).gt.D(S( g(u,k )))+w( g(u,k )))then
              D(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
              Q(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
```

```
      Q(u)=0
      METRHTHS=METRHTHS+1
                    NDUF=NDUF+1
      ENDDO
  115 CONTINUE
      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
      enddo
      write(*,*)NDUF
      END Subroutine DijkstraAlgorithm
      END MODULE

      PROGRAM Dijkstra
      use Dijkstra_Mod
      IMPLICIT NONE
      INTEGER N,L,r,t,i
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\4000 to 80000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\SSSPP\
     1SSSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D,F,g)
      Call DijkstraAlgorithm(S,E,w,D,F,g)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM Dijkstra
```

Αλγόριθμος Floyd-Warshall με υπορουτίνα τη μέθοδο Bellman-Ford που τερματίζει εάν δεν αναβαθμιστούν τα βάρη συνδέσμων
Επίλυση : All Pairs Shortest Paths Problem

```
      module Floyd_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:)
      real,allocatable::D(:,:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N,N))
      DO 114 I=1,N
      do 114 J=1,N
      D(I,J)=1.2E+38
  114 continue
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
```

```
      ENDDO
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE FloydAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w
      real,dimension(:,:)::D
      CHARACTER NAME1*12,NAME2*13
      character(38)::NAME3
      NAME1='Κόμβος Αρχής'
      NAME2='Κόμβος Τέλους'
      NAME3='Ελάχιστη Απόσταση από τον Αρχικό Κόμβο'
      do k=1,N
        D(k,k)=0
        do i=1,N
          NDIF=0
          do j=1,L
        if(D(k,E(j)) > D(k,S(j))+w(j))then
        D(k,E(j)) = D(k,S(j))+w(j)
        NDIF=NDIF+1
        endif
           ENDDO
           if(NDIF.eq.0)go to 115
        ENDDO
  115 continue
      ENDDO

      WRITE(9,116)NAME1,NAME2,NAME3
  116 FORMAT(3X,A,10X,A,10X,A)
      DO J=1,N
      DO I=1,N
      IF(D(J,I)/=1.2E+38)then
      WRITE(9,117)J,I,D(J,I)
  117 FORMAT(3X,I4,24X,I4,24X,F12.3)
      endif
      enddo
      ENDDO
      END Subroutine FloydAlgorithm
      end module

      PROGRAM Floyd
      use Floyd_Mod
      IMPLICIT NONE
      real etime
      REAL elapsed(2)
      real total
      INTEGER :: r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::D(:,:)
      real,allocatable::w(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\4000 to 80000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\ASSPP\
     1ASSPPOutput.f')
      CALL Initialization(S,E,w,D)
      call FloydAlgorithm(S,E,w,D)
      total=etime(elapsed)
      WRITE(9,*) 'total=',total,'user=',elapsed(1),'system',elapsed(2)
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Johnson
Επίλυση : All Pairs Shortest Paths Problem

```
      module Johnson_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L+N),E(L+N))
      Allocate(w(L+N),D(N+L))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N+L
  114 D(I)=1.2E+38
      r=N+1
      D(r)=0
      Increase=1
      do i=L+1,L+N
      w(i)=0
      S(i)=r
      E(i)=Increase
      Increase=Increase+1
      enddo
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE Moore(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      do k=1,N
        do i=1,L+N
           if(D(S(i))+w(i)<D(E(i)))then
           D(E(i))= D(S(i))+w(i)
           endif
        enddo
      enddo
        do i=1,L+N
        w(i)=D(S(i))+w(i)-D(E(i))
        enddo
      End Subroutine Moore

*     Main Loop (Step 2)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N+L)::Q
      integer::METRHTHS,u
```

```
    CHARACTER NAME1*12,NAME2*13
    character(38)::NAME3
    NAME1='Κόμβος Αρχής'
    NAME2='Κόμβος Τέλους'
    NAME3='Ελάχιστη Απόσταση από τον Αρχικό Κόμβο'
    WRITE(9,116)NAME1,NAME2,NAME3
116 FORMAT(3X,A,10X,A,10X,A)
    DO Iz=1,N
       do i=1,L+N
       D(i)=1.2E+38
       Q(i)=D(i)
       enddo
       D(Iz)=0
    METRHTHS=0
    u=r
    DO WHILE(METRHTHS/=N)
      do k=1,L
      if(S(k).eq.u.and.D(E(k)).gt.D(S(k))+w(k))then
            D(E(k))=D(S(k))+w(k)
            Q(E(k))=D(E(k))
      endif
      enddo
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
    DO I=1,N
    IF(D(I)/=1.2E+38)then
    WRITE(9,117)Iz,I,D(I)
117 FORMAT(3X,I4,24X,I4,24X,F12.3)
    endif
    enddo
    ENDDO
    END Subroutine DijkstraAlgorithm
    end module

    PROGRAM Johnson
    use Johnson_Mod
    IMPLICIT NONE
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    INTEGER :: r,t
    integer, allocatable::S(:),E(:)
    real,allocatable::w(:),D(:)
    OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
   1ks\1600 to 16210.f')
    OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\ASSPP\
   1ASSPPOutput.f')
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    CALL Initialization(S,E,w,D)
    call Moore(S,E,w,D)
    call DijkstraAlgorithm(S,E,w,D)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    END
```

Αλγόριθμος : Τροποποιημένος Dijkstra για το APSPP με βέλτιστη αναπαράσταση δικτύου
Επίλυση : All Pairs Shortest Paths Problem

```
      MODULE Dijkstra_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:)
      real,allocatable::D(:,:)
      integer, allocatable::  F(:),g(:,:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N,N),F(N),g(N,N))
      Do i=1,N
      F(i)=0
      enddo
      DO 114 I=1,N
      do 114 J=1,N
      D(I,J)=1.2E+38
  114 continue

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:,:)::D
      real,dimension(:)::w
      real,dimension(N)::Q
      integer::METRHTHS,u
      CHARACTER NAME1*8
      character(38)::NAME2
      NAME1='Κόμβος'
      NAME2='Ελάχιστη Απόσταση από τον Κόμβο Προέλευσης'

      WRITE(9,116)NAME1,NAME2
  116 FORMAT(3X,A,10X,A)
      Do iz=1,N
      METRHTHS=0
      u=iz
      D(u,u)=0
      NDUF=0
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do k=1,F(u)  !*RELAXATION STEP
```

```
      if(D(iz,E( g(u,k ))).gt.D(iz,S( g(u,k )))+w( g(u,k )))then
            D(iz,E( g(u,k )))=D(iz,S( g(u,k )))+w( g(u,k ))
            Q(E( g(u,k )))=D(iz,S( g(u,k )))+w( g(u,k ))
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,N
      if(Q(i)/=0.and.D(iz,i)<ShortestDistance)then
    ShortestDistance=D(iz,i)
    u=i
      NDIF=1
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    IF(NDIF.EQ.0)GO TO 115
      NDUF=NDUF+1
    Q(u)=0
    METRHTHS=METRHTHS+1
    NDIF=0
    ENDDO
115 CONTINUE
    DO I=1,N
    IF(D(iz,I)/=1.2E+38)then
    WRITE(9,117)I,D(iz,I)
117 FORMAT(3X,I4,24X,F12.3)
    endif
    enddo
    EnDDo

    END Subroutine DijkstraAlgorithm
    END MODULE

    PROGRAM Dijkstra
    use Dijkstra_Mod
    IMPLICIT NONE
    INTEGER N,L,r,t,i
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    integer, allocatable:: S(:),E(:),F(:),g(:,:)
    real, allocatable :: w(:),D(:,:)
    OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
   1ks\2800 to 48580.f')
    OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\ASSPP\
   1ASSPPOutput.f')
    WRITE(9,118) result
118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    Call Initialization(S,E,w,D,F,g)
    Call DijkstraAlgorithm(S,E,w,D,F,g)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    END PROGRAM Dijkstra
```

Αλγόριθμος Hoffman-Pavley με υπορουτίνα τον αλγόριθμο Dijkstra
Επίλυση : 2nd Shortest Path Problem

```
    MODULE HoffmanPavley_Mod
    CONTAINS

    SUBROUTINE Initialization(S,E,w,D,F,g)
    INTEGER N,L,r,t
    COMMON /AREA1/ N,L,r,t
    integer, allocatable:: S(:),E(:),F(:),g(:,:)
    real, allocatable :: w(:),D(:)
    CHARACTER EPIGRAFES*100
```

```
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N),F(N),g(N,N))
      Do i=1,N
      F(i)=0
      enddo

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE DijkstraAlgorithm(S,E,w,D,F,g)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
      real,dimension(:)::w,D
      real,dimension(N)::Q,V
      integer,dimension(L)::p,pl
      integer,dimension(N)::Parent
      integer::METRHTHS,u
      METRHTHS=0
      u=r
      NDUF=0
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do k=1,F(u)  !*RELAXATION STEP
        if(D(E( g(u,k ))).gt.D(S( g(u,k ))+w( g(u,k )))then
              D(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
              Q(E( g(u,k )))=D(S( g(u,k )))+w( g(u,k ))
              Parent(E( g(u,k ))) = g(u,k )
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        NDIF=1
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      IF(NDIF.EQ.0)GO TO 115
        NDUF=NDUF+1
      Q(u)=0
      METRHTHS=METRHTHS+1
      NDIF=0
      ENDDO
  115 CONTINUE
      write(*,*)NDUF
  116 FORMAT(3X,A,10X,A)
      DO I=1,N
      WRITE(9,117)I,D(I)
  117 FORMAT(3X,I4,24X,F12.3)
```

```
    enddo

    It=0
    k=t
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      pl(It)=Parent(k)
      k=S(Parent(k))
      go to 119
      endif

    i=It
    j=1
120 continue
    if(i.ne.0)then
    p(j)=S(pl(i))
    i=i-1
    j=j+1
    go to 120
    endif
    p(j)=E(pl(1))

    SecondSP=1.2E+38

    !Hoffman Pavley
    is=0
    Do i=j-1,1,-1
    is=is+1
    X=w(pl(is))
    w(pl(is))=1.2E+38
    u=p(i)
    do l=1,N
    V(l)=1.2E+38
    Q(l)=1.2E+38
    enddo
    Q(u)=0
    V(u)=0
    NDIF=0
    METRHTHS=0
    DO WHILE(METRHTHS/=N)
      do k=1,F(u)  !*RELAXATION STEP
      if(V(E( g(u,k ))).gt.V(S( g(u,k ))+w( g(u,k )))then
            V(E( g(u,k )))=V(S( g(u,k )))+w( g(u,k ))
            Q(E( g(u,k )))=V(S( g(u,k )))+w( g(u,k ))
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do l=1,N
      if(Q(l)/=0.and.V(l)<ShortestDistance)then
    ShortestDistance=V(l)
    u=l
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
    if(SecondSP > V(t)+D(p(i)))then
    SecondSP=V(t)+D(p(i))
    endif
    w(pl(is))=X
    enddo

    write(9,*)'Κόστος 2ης Συντομότερης Διαδρομής=',SecondSP
    END Subroutine DijkstraAlgorithm
    END MODULE
```

```
      PROGRAM HoffmanPavley
      use HoffmanPavley_Mod
      IMPLICIT NONE
      INTEGER N,L,r,t,i
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2600 to 40000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\kthSPP
     1\kthSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D,F,g)
      Call DijkstraAlgorithm(S,E,w,D,F,g)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM HoffmanPavley
```

Αλγόριθμος Eppstein
Επίλυση : 2nd Shortest Path Problem

```
      module Eppstein_Mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N))
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(t)=0
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE EppsteinAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E
      real,dimension(:)::w
      real,target::D(:)
      real,pointer::point(:)
      real,dimension(N)::Q,lp
      DIMENSION g(L)
      integer,allocatable::Z(:),G_Z(:),nextZ(:),u(:),Note(:),parent(:)
      integer::METRHTHS,ee
      METRHTHS=0
      ee=t
```

```
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(t)=0
      DO WHILE(METRHTHS/=N)
        do k=1,L  !*RELAXATION STEP
        if(E(k).eq.ee.and.D(S(k)).gt.D(E(k))+w(k))then
              D(S(k))=D(E(k))+w(k)
              Q(S(k))=D(S(k))
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      ee=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      Q(ee)=0
      METRHTHS=METRHTHS+1
      ENDDO

      k1=0            !Kataskeuh Dendrwn T kai G-T kai nextT(u) kathe komvou u pou
!                     !  anhkei sto T
      k2=0
      do i=1,L
      g(i)=w(i)+D(E(i))-D(S(i))
      if(g(i)<0.001)k1=k1+1
      if(g(i)>0.001)k2=k2+1
      enddo
      Allocate(Z(k1),G_Z(k2),u(k1),nextZ(k1),Note(L),parent(L))
      k1=0
      k2=0
      do i=1,L
      if(g(i)<0.001)then
      k1=k1+1
      Z(k1)=i
      u(k1)=S(Z(k1))
      nextZ(u(k1))=E(Z(k1))
      else
      k2=k2+1
      G_Z(k2)=i
      endif
      enddo
      write(9,*)'Σύνδεσμοι που ανήκουν στο δένδρο T'
      WRITE(9,*)(Z(i),i=1,k1)
      write(9,*)'Σύνδεσμοι που ανήκουν στο δένδρο G-T'
      WRITE(9,*)(G_Z(i),i=1,k2)
      SecondShortestPath=1.2E+38
      Do i=1,k2
      if(SecondShortestDistance > w(G_Z(i)))then
      SecondShortestDistance=w(G_Z(i))
      endif
      enddo
      SecondShortestDistance=SecondShortestDistance+D(r)
      write(9,*)'Κόστος Συντομότερης Διαδρομής',D(r)
      write(9,*)'Κόστος 2ης Συντομότερης Διαδρομής',SecondShortestDistance

      END Subroutine EppsteinAlgorithm
      end module

      PROGRAM Eppstein
      use Eppstein_Mod
      IMPLICIT NONE
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      INTEGER :: r,t
      integer, allocatable::S(:),E(:)
```

```
      real,allocatable::w(:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2600 to 40000.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\kthSPP
     1\kthSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(S,E,w,D)
      call EppsteinAlgorithm(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος Yen
Επίλυση : kth Loopless Shortest Paths Problem

```
      module Yen_mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer,allocatable::S(:),E(:)
      real,allocatable::w(:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(S(L),E(L),w(L),D(N))
  116 CONTINUE
      DO 113 I=1,L
  113 READ(8,*)S(I),E(I),Li,w(I)
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE YenAlgorithm(S,E,w,D)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer,dimension(:)::S,E
      real,dimension(:)::w,D
      integer,dimension(L)::Parent
      integer,dimension(N)::AA,TT
      integer,dimension(N,N)::A,B,Idios
      real,dimension(N)::Q,DD,DDD
      real,dimension(l)::ww
      real :: Minimize
      integer :: u
      k=1
      Read(*,*)KL ! Απαιτούμενος Αριθμός Συντομότερων Διαδρομών χωρίς κυκλικό βρόγχο
      lll=0
  124 CONTINUE
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      Do i=1,L
      Parent(i)=0
      enddo
```

```
      METRHTHS=0
      u=r
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do i=1,L
        if(S(i).EQ.u.AND.D(E(i)).gt.D(S(i))+w(i))then
              D(E(i))=D(S(i))+w(i)
              Parent(E(i)) = i
        endif
        enddo
      ShortestDistance=1.2E+38
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      Q(u)=0
      METRHTHS=METRHTHS+1
      ENDDO

      It=0
      kk=t
  119 continue
        if(kk.ne.r.and.Parent(kk).ne.0)then
        It=It+1
        AA(It)=E(Parent(kk))
        kk=S(Parent(kk))
        go to 119
        endif
        It=It+1
        AA(It)=r

      DDD(1)=0
      do i=1,It
      A(k,i)=AA(It+1-i)
          IF(i>1)THEN
          DO jR=1,L
      IF(S(jR).EQ.A(k,i-1).AND.E(jR).EQ.A(k,i))DDD(i)=DDD(i-1)+w(jR)
          ENDDO
          ENDIF
      enddo

      do i=1,It
      enddo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  130 CONTINUE

      k=k+1
      Do iDev=1,It
            iD = A(k-1,iDev)
            do i=1,L
            ww(i)=w(i)
            enddo
           do j=1,k-1
                   if (A(j,iDev+1).eq.A(k-1,iDev+1))then
                         Idios(j,iDev)=0
                   endif
      if(A(j,iDev+1).ne.A(k-1,iDev+1).and.Idios(j,iDev).eq.0)then
                               do ioi=1,L
      if (E(ioi).eq.A(j,iDev+1).and.S(ioi).eq.A(j,iDev))ww(ioi)=1.2E+38!!!
                               enddo
      endif
           enddo
```

```
          do ioi=1,L
      if(S(ioi).eq.A(k-1,iDev).and.E(ioi).eq.A(k-1,iDev+1))ww(ioi)=1.2E+
     138
          enddo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

          u=iD
      IF(u.ne.t)THEN
      DO I=1,N
      D(I)=1.2E+38
      Q(I)=1.2E+38
      enddo
      D(u)=0
      Q(u)=0
      if(iDev.ne.1)then          !!!!!!Mh Kyklikos Brogxos
      do i=1,iDev
      Q(A(k-1,i))=0
      enddo
      endif
      Do i=1,L
      Parent(i)=0
      enddo

      METRHTHS=0
      DO WHILE(METRHTHS/=N)
        do i=1,L
        if(S(i).EQ.u.AND.D(E(i)).gt.D(S(i))+ww(i))then
              D(E(i))=D(S(i))+ww(i)
              Parent(E(i)) = i
        endif
        enddo
      ShortestDistance=1.2E+38
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      Q(u)=0
      METRHTHS=METRHTHS+1
      ENDDO

      if(D(t)<1.2E+38)then
      write(*,*)'D(t)',D(t)
      lll=lll+1
      DD(lll)=0
      Itt=0
      kk=t
  120 continue
        if(kk.ne.iD.and.Parent(kk).ne.0)then
        Itt=Itt+1
        AA(Itt)=E(Parent(kk))
        DD(lll)=DD(lll)+ww(Parent(kk))
        kk=S(Parent(kk))
        go to 120
        endif
        Itt=Itt+1
        AA(Itt)=iD
      if(iDev.ne.1)then
      JJJ=iDev
  131 CONTINUE
      JJJ=JJJ-1
      Itt=Itt+1

      IF(Itt>N)THEN
      lll=lll-1
      Go TO 132
```

```
    endIF

    AA(Itt)=A(k-1,JJJ)
    IF(JJJ.NE.1)GO TO 131
    endif

    DD(lll)=DD(lll)+DDD(iDev)
    TT(lll)=Itt

    do i=1,Itt
    B(lll,i)=AA(Itt+1-i)
    enddo

    do i=1,Itt
    enddo
    endif

132 CONTINUE
    ENDIF
       ENDDO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    Minimize=1.2E+38
    do i=1,lll
    if(DD(i)<Minimize)then
    Minimize=DD(i)
    idds=i
    endif
    enddo
    It=TT(idds)

    DDD(1)=0
    do i=1,It
    A(k,i)=B(idds,i)
        IF(i>1)THEN
        DO jR=1,L
    IF(S(jR).EQ.B(idds,i-1).AND.E(jR).EQ.B(idds,i))DDD(i)=DDD(i-1)+w(j
   1R)
        ENDDO
        ENDIF
    enddo
    WRITE(9,*)k,'ή Συντομότερη Διαδρομή',(A(k,i),i=1,It),'Κόστους =',DD(idds)
    DD(idds)=1.2E+38
    IF(k < KL)GO TO 130

    END Subroutine YenAlgorithm
    END module

    PROGRAM Yen
    use Yen_Mod
    IMPLICIT NONE
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    INTEGER :: N,L,r,t
    integer,allocatable::S(:),E(:)
    real,allocatable::w(:),D(:)
    OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
   1ks\1000 to 10010.f')
    OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\KthSPP
   1\kthSPPOutput.f')
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    CALL Initialization(S,E,w,D)
    call YenAlgorithm(S,E,w,D)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    END
```

Αλγόριθμος : Τροποποιημένος αλγόριθμος Yen με αποθήκευση δικτύου μέσω πίνακα γειτνίασης που χρησιμοποιήθηκε στις εφαρμογές
Επίλυση : kth Shortest Paths Problem

```
      module Yen_mod
      contains
*     (Step 0)
      SUBROUTINE Initialization(w,D)
      INTEGER N,SNode,ENode
      COMMON /AREA1/ N,SNode,ENode
      real,allocatable::w(:,:),D(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)SNode,ENode
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      Allocate(w(N,N),D(N))
      DO 116 i=1,N
      DO 116 j=1,N
      w(i,j)=1.2E+38
      IF(i.eq.j)w(i,j)=0
  116 CONTINUE
      DO 113 I=1,L
  113 READ(8,*)LStart,lEnds,Li,w(LStart,LEnds)
      CLOSE(8)
      RETURN
      END Subroutine Initialization

*     Main Loop (Step 1)
      SUBROUTINE YenAlgorithm(w,D)
      INTEGER N,SNode,ENode
      COMMON /AREA1/ N,SNode,ENode
      real,dimension(:,:)::w
      real,dimension(:)::D
      integer,dimension(N)::u,p,Sp,Spur,Z,SOUPE,ZZ
      integer,dimension(N,N)::A,C
      real,dimension(N)::Q,Dista2,V,Qer
      real :: Minimize
      DO 114 I=1,N
      D(I)=w(I,ENode)
      u(I)=ENode
  114 continue
      do k=1,N
         ITERM=0
         do i=1,N
         Minimize=D(i)
              do j=1,N
       if(w(i,j)+D(j)<D(i))then
       D(i)=w(i,j)+D(j)
       u(i)=j
       endif
              enddo
         if(Minimize.EQ.D(i))ITERM=ITERM+1
         enddo
         if(ITERM.EQ.N)GO TO 115
      enddo
  115 CONTINUE
      ks=1
         J=SNode
```

```
      p(ks)=SNode
      DO While(J.NE.ENode)
      ks=ks+1
      p(ks)=u(J)
      J=u(J)
      ENDDO

    lss=0
    iss=0
    V=+1.2E+38
    issst=0
    do while(lss/=ks-1)   !lss/=11
    write(*,*)'---------------------------------------) '
    irtrt=0
    lss=lss+1                      !lss=1
    H=w(p(ks-lss),p(ks-lss+1))  !Weight(11,12)=+Inf
    w(p(ks-lss),p(ks-lss+1))=+1.2E+38
    METRHTHS=0
    lp=p(ks-lss)
    lk=lp                         !lp=11
    write(*,*)'lk=',lk
    do 123 i=1,N
    Q(i)=+1.2E+38
    Dista2(i)=+1.2E+38
123 continue
    Q(lp)=0
    Dista2(lp)=0
    DO WHILE(METRHTHS/=ENode)   ! DIJKSTRA ALGORITHM
      do j=1,N
      if(j.ne.lp.and.Dista2(j).gt.Dista2(lp)+w(lp,j))then
        Dista2(j)=Dista2(lp)+w(lp,j)
        Q(j)=Dista2(j)
      endif
      enddo
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.Dista2(i)<ShortestDistance)then
    ShortestDistance=Dista2(i)
    lp=i
      endif
      enddo
    irtrt=irtrt+1
    Sp(irtrt)=lp
    write(*,*)'***',Sp(irtrt),irtrt
    if(lp.EQ.p(ks-lss).OR.lp.EQ.p(ks))go to 124
    Q(lp)=0
    METRHTHS=METRHTHS+1
    ENDDO                      ! DIJKSTRA ALGORITHM          !
124 CONTINUE
     write(*,*)'po',Dista2(1)
    do i=1,Nodes
    Q(i)=+1.2E+38
    enddo
    Q(ENode)=0
    lwww=ENode
    irt=1
    Spur(1)=ENode
    DO WHILE(METRHTHS/=ENode+1)
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.w(i,lwww)+Dista2(i)<ShortestDistance)then
    ShortestDistance=w(i,lwww)+Dista2(i)
    lwww=i
    irt=irt+1
    Spur(irt)=lwww
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
```

```
    write(*,*)'Spur',Spur(irt),irt,'lwww=',lwww,w(8,12)+Dista2(8)
    if(lwww.EQ.ENode.OR.lwww.EQ.lk)go to 127
    Q(lwww)=0
    METRHTHS=METRHTHS+1
    ENDDO                     ! DIJKSTRA ALGORITHM
127 CONTINUE

    w(p(ks-lss),p(ks-lss+1))=H
    B=Dista2(p(ks))-Dista2(p(ks-lss))
    write(*,*)'B=',B
    IF(B<+1E+38)then
    issst=issst+1
    V(issst)=B+D(SNode)-D(lk)
    iss=iss+1
    i=0
125 CONTINUE
    i=i+1
    A(iss,i)=p(i)
    write(*,*)p(i)
    if(A(iss,i).NE.lk)go to 125

    iZZZ=irt
126 continue
    i=i+1
    iZZZ=iZZZ-1
    A(iss,i)=Spur(iZZZ)
    IF(iZZZ.ne.0)go to 126
    Z(iss)=i-1
                endif
    enddo

    ieres=0
    do ipopop=1,iss
    Qer(ipopop)=1
    enddo
130 CONTINUE
    ieres=ieres+1
    ShortestDistance=+1.2E+38
    do iop=1,issst
    if(Qer(iop)/=0.AND.ShortestDistance>V(iop))then
    Iy=iop
    ShortestDistance=V(iop)
    endif
    enddo
    write(*,*)'Iy=',Iy
    do kios=1,N
    C(ieres,kios)=A(Iy,kios)
    enddo
    SOUPE(ieres)=V(Iy)
    Qer(Iy)=0
    ZZ(ieres)=Z(Iy)
    write(*,*)SOUPE(ieres),ZZ(ieres),ieres
    if(ieres.lt.iss)go to 130


    do irs=1,iss
    write(*,*)'ISS=',irs,'A=',(A(irs,ir),ir=1,Z(irs))
    enddo
    write(9,*)'ΟΙ ΣΥΝΤΟΜΟΤΕΡΕΣ ΔΙΑΔΡΟΜΕΣ ΕΙΝΑΙ','   ΜΕ ΑΠΟΣΤΑΣΗ'
    write(9,*)(p(i),i=1,ks),'                ',D(SNode)
    do irs=1,iss
    WRITE(9,*)(C(irs,ir),ir=1,ZZ(irs)),'          ',SOUPE(irs)
    enddo
    END Subroutine YenAlgorithm
    END module

    PROGRAM Yen
```

```
      use Yen_Mod
      IMPLICIT NONE
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      INTEGER :: N,SNode,ENode !%æ£™¦ª  ¨¦â¢œ¬©žª ¡˜   ¨¦¦¨ ©£¦ç ˜¤«å©«¦ ®˜
      real,allocatable::w(:,:),D(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\6 to 16.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\KthSPP
     1\kthSPPOutput.f')
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      CALL Initialization(w,D)
      call YenAlgorithm(w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      END
```

Αλγόριθμος LARAC (Lagrange Relaxation Based Aggregation Cost)
Επίλυση : Resource Constrained Shortest Path Problem (Εύρεση της συντομότερης διαδρομής που ικανοποιεί έναν περιορισμό)

```
      MODULE Larac_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D,F,g,RR)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, allocatable:: S(:),E(:),F(:),g(:,:)
      real, allocatable :: w(:),D(:),RR(:)
      CHARACTER EPIGRAFES*100
      READ(8,111)N
  111 FORMAT(15X,I10)
      READ(8,111)L
      READ(8,112)r,t
  112 FORMAT(16X,I6,16X,I6)
      READ(8,*)EPIGRAFES
      allocate(S(L),E(L),w(L),D(N),F(L),g(N,L),RR(L))
      Do i=1,N
      F(i)=0
      enddo
      Do i=1,L
      read(7,*)RR(I)
      enddo

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      F(S(I))=F(S(I))+1
      g( S(I),F(S(I)) ) = Number
      ENDDO
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE LaracAlgorithm(S,E,w,D,F,g,RR)
      INTEGER N,L,r,t
      COMMON /AREA1/ N,L,r,t
      integer, dimension(:)::S,E,F
      integer, dimension(:,:)::g
```

```
    real,dimension(:)::w,D,RR
    real,dimension(N)::Q
    real,dimension(L)::we
    integer,dimension(L)::Parent
    integer::METRHTHS,u
    real Wc,Wd,Wl
    real Lagrange,LowerBound
    Read(*,*) b    ! Εισήγαγε την τιμή του περιορισμού διαθέσιμου πόρου
    Wc=1.2E+38
    Wd=1E+38
    UpperBound=0
    LowerBound=1.2E+38
    IRRR=0
    Do i=1,L
    we(i)=w(i)
    enddo

126 continue
    IRRR=IRRR+1
    !YPOTOUTINA DIJKSTRA
    METRHTHS=0
    DO I=1,N
    D(I)=1.2E+38
    enddo
    D(r)=0
    u=r
    do 123 i=1,N
123 Q(i)=1.2E+38
    Q(r)=0
    DO WHILE(METRHTHS/=N)
      do k=1,F(u)  !*RELAXATION STEP
      if(D(E( g(u,k ))).gt.D(S( g(u,k )))+we( g(u,k )))then
            D(E( g(u,k )))=D(S( g(u,k )))+we( g(u,k ))
            Q(E( g(u,k )))=D(S( g(u,k )))+we( g(u,k ))
            Parent(E( g(u,k ))) = g(u,k )
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
     NDIF=1
      endif
      enddo
    IF(NDIF.EQ.0)GO TO 115
    Q(u)=0
    METRHTHS=METRHTHS+1
    NDIF=0
    ENDDO
115 CONTINUE
    WRITE(9,116)NAME1,NAME2
116 FORMAT(3X,A,10X,A)
    DO I=1,N
    WRITE(9,117)I,D(I)
117 FORMAT(3X,I4,24X,F12.3)
    enddo

    It=0
    k=t
    Cost=0
    Rp=0
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      Rp=Rp+RR(Parent(k))
      Cost=Cost+w(Parent(k))
```

```
      k=S(Parent(k))
      go to 119
      endif
      Wl=D(t)

    if(IRRR.eq.2)then
    if(Rp>b)then
    go to 125
    else
    Wd=Cost
    Rpd=Rp
    endif
    endif
    if(IRRR.eq.1)then
    if(Rp<b)then
    go to 125
    else
    Upperbound=Wl
    Wc=Cost
    Rpc=Rp
    endif
    endif
    !TELOS YPOROUTINAS

    if(IRRR.eq.1)then
    do i=1,L
    we(I)=RR(I)
    enddo
    endif

    if(IRRR.gt.3)then
    if(Rp.gt.b)then
    Wc=Cost
    Rpc=Rp
    endif
    if(Rp<b)then
    Wd=Cost
    Rpd=Rp
    endif
    endif

    Lagrange = (Wc-Wd)/(Rpd-Rpc)
    if(Lagrange<0)Lagrange=0
    if(IRRR.gt.2)then
    do i=1,L
    we(I)=w(I)+Lagrange*RR(I)
    enddo
    endif

    write(9,*)Wc,Wd
    if(UpperBound<Wc)UpperBound=Wc
    if(LowerBound>Wd)LowerBound=Wd
    if(UpperBound.eq.LowerBound.or.IRRR.eq.11)go to 125

    go to 126
125 continue

    END Subroutine LaracAlgorithm
    END MODULE

    PROGRAM Larac
    use Larac_Mod
    IMPLICIT NONE
    INTEGER N,L,r,t,i
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    integer, allocatable:: S(:),E(:),F(:),g(:,:)
```

```
      real, allocatable :: w(:),D(:),RR(:)
      OPEN(8,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\Networ
     1ks\2800 to 48580.f')
      OPEN(7,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\RCSPP\
     1RCSPP1000 to 10010.f')
      OPEN(9,FILE='C:\Users\kostas\Desktop\FORTRAN LIBRARY\FOLDER\RCSPP\
     1RCSPPOutput.f')
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D,F,g,RR)
      Call LaracAlgorithm(S,E,w,D,F,g,RR)
      CALL ETIME(tarray, result)
      WRITE(*,118) result
      END PROGRAM Larac
```

Αλγόριθμος για τη δρομολόγηση μεταφορικών μέσων και τη φόρτιση στατικών δικτύων σύμφωνα με τη μέθοδο όλα ή τίποτα.
Επίλυση : Καταμερισμός της ζήτησης σε στατικά δίκτυα.

```
      MODULE StaticNetwork_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      write(*,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE StaticNetworkAlgorithm(S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q
      real,dimension(N,N)::Demand
      real,dimension(L)::Flow,ww
      integer,dimension(L)::p
      integer,dimension(N)::Parent
      integer::METRHTHS,u
      H=0
      iz=0
      do i=1,L
      Flow(i)=0
      enddo
      do i=1,N
      read(7,*)(Demand(i,j),j=1,N) !Ζήτηση Μετακινήσεων
      enddo

      do i=1,L
```

```
    ww(i)=w(i)
    enddo

147 continue
    DO r=1,N

    DO 114 I=1,N
114 D(I)=1.2E+38
    D(r)=0

    METRHTHS=0
    u=r
    do 123 i=1,N
123 Q(i)=1.2E+38
    Q(r)=0
    DO WHILE(METRHTHS/=N)
      do i=1,L  !*RELAXATION STEP
      if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
            D(E( i))=D(S( i))+ww( i)
            Q(E( i))=D(S( i))+ww( i)
            Parent(E( i)) = i
            write(*,*)Parent(E(i))
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    DO I=1,N
    WRITE(9,117)I,D(I)
117 FORMAT(3X,I4,24X,F12.3)
    enddo
    write(*,*)'Telos'

    Do lll=1,N
    It=0
    k=lll
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      p(It)=Parent(k)
      k=S(Parent(k))
      go to 119
      endif
    do i=1,It
     Flow(p(i))=Flow(p(i))+Demand(r,lll)
     write(9,*)Flow(p(i)),p(i),lll
    enddo

    IF(D(lll).ne.1.2E+38)H=H+D(lll)*Demand(r,lll)

    Enddo
    ENDDO

    Do i=1,L
    write(9,*)S(i),'-',E(i),Flow(i),ww(i)
    AA=AA+ww(i)
    Enddo

    write(*,*)H,AA/L
```

```
      END Subroutine StaticNetworkAlgorithm
      END MODULE

      PROGRAM StaticNetwork
      use Dijkstra_Mod
      IMPLICIT NONE
      INTEGER N,L
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='XronosDianyshs6.f')
      OPEN(7,FILE='Zhthsh.f')
      OPEN(9,FILE='EquilibriumOutput.f')
      WRITE(9,118) result
  118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D)
      Call DijkstraAlgorithm(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM Dijkstra
```

Αλγόριθμος φόρτισης ενός δυναμικού δικτύου σύμφωνα με τη στοχαστική ισορροπία του χρήστη, όπου η επιλογή διαδρομής γίνεται βάσει του λογιστικού μοντέλου.
Επίλυση : Στοχαστικός καταμερισμός της ζήτησης σε δυναμικά δίκτυα, χωρίς τη χρήση συστημάτων πλοήγησης.

```
      MODULE StochasticUserEquilibrium_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      write(*,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE StochasticUserEquilibrium (S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q,SumP
      real,dimension(N,L)::FF
      real,dimension(N,N)::Demand,T
      real,dimension(L)::Flow,ww,ha
      integer::METRHTHS,u
      do i=1,N
      do j=1,L
```

```
      FF(i,j)=0
    enddo
    enddo
    H=0
    iz=0
    do i=1,L
    Flow(i)=0
    enddo
    do i=1,N
    read(7,*)(Demand(i,j),j=1,N)
    write(*,*)(Demand(i,j),j=1,N)
    enddo

    do i=1,L
    ww(i)=w(i)
    enddo

147 continue
    DO r=1,N

    DO 114 I=1,N
114 D(I)=1.2E+38
    D(r)=0

    do i=1,N
    SumP(i)=0
    enddo
    do i=1,L
    ha(i)=0
    enddo
    do i=1,N
    do j=1,N
    T(i,j)=0
    enddo
    enddo

    METRHTHS=0
    u=r
    do 123 i=1,N
123 Q(i)=1.2E+38
    Q(r)=0
    DO WHILE(METRHTHS/=N)
      do i=1,L  !*RELAXATION STEP
      if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
            D(E( i))=D(S( i))+ww( i)
            Q(E( i))=D(S( i))+ww( i)
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    enddo

    do i=1,L
    zz=0.1*(D(E(i))-D(S(i))-ww(i))
    if(D(E(i))>D(S(i)))ha(i)=ha(i)+2.71828**zz
    if(D(E(i))>D(S(i)))ha(i)=ha(i)+0
    SumP(E(i))=SumP(E(i))+ha(i)
    enddo
```

```
    DO lll=1,N
         if(lll.ne.r.and.Demand(r,lll).ne.0)then
                 do i=1,N
                 T(r,i)=0
                 enddo
         T(r,lll)=Demand(r,lll)/60

    do i=N,1,-1
    do j=1,L
      if(E(j).eq.i.and.T(r,E(j)).ne.0)then
      FF(lll,j)=FF(lll,j)+T(r,E(j))*(ha(j)/SumP(E(j)))
      Flow(j)=Flow(j)+T(r,E(j))*(ha(j)/SumP(E(j)))
      T(r,S(j))=T(r,E(j))*(ha(j)/SumP(E(j)))
      write(*,*)T(r,S(j)),T(r,E(j)),S(j),E(j)
      endif
    enddo
    enddo
         endif
      Do i=1,N
      T(r,i)=0
      Enddo
    ENDDO

    ENDDO
    do i=1,L
    ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4))
    enddo
    iz=iz+1
    if(iz.ne.60)go to 147
    AA=0
    Do i=1,L
    write(9,*)S(i),'-',E(i),Flow(i),ww(i) !Φόρτιση και χρόνος Διάνυσης Συνδέσμων
    AA=AA+ww(i)
    Enddo
    Do j=1,N
    Do i=1,L
    write(*,*)j
    write(*,*)S(i),'-',E(i),FF(j,i)
    enddo
    Enddo
    write(*,*)AA/L   ! Μέσος Χρόνος Διάνυσης Συνδέσμων
    END Subroutine StochasticEquilibrium
    END MODULE

    PROGRAM StochasticUserEquilibrium
    use StochasticUserEquilibrium_Mod
    IMPLICIT NONE
    INTEGER N,L
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    integer, allocatable:: S(:),E(:)
    real, allocatable :: w(:),D(:)
    OPEN(8,FILE='XronosDianyshs6.f')
    OPEN(7,FILE='Zhthsh.f')
    OPEN(9,FILE='EquilibriumOutput.f')
    WRITE(9,118) result
118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    Call Initialization(S,E,w,D)
    Call StochasticUserEquilibrium(S,E,w,D)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    END PROGRAM StochasticUserEquilibrium
```

Αλγόριθμος φόρτισης ενός δυναμικού δικτύου υπό την προϋπόθεση ότι όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης, τα οποία ακολουθούν την αυτόνομη δρομολόγηση. Επίλυση : Προσομοίωση καταμερισμού της ζήτησης σύμφωνα με τη συνθήκη μαζικής χρήσης συστημάτων πλοήγησης, υποθέσεις Σταθόπουλος (2010).

```
      MODULE AytonomhDromologhsh_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      write(*,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE AytonomhDromologhsh(S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      integer,dimension(N,L)::p
      integer,dimension(N)::RR,Parent
      real,dimension(N)::Q,SumP
      real,dimension(4)::A
      real,dimension(N,N)::Demand,T
      real,dimension(L)::Flow,ww,ha
      integer::METRHTHS,u
      HH=0
      iz=0
      do i=1,L
      Flow(i)=0
      enddo
      do i=1,N
      read(7,*)(Demand(i,j),j=1,N)
      write(*,*)(Demand(i,j),j=1,N)
      enddo
      ik=1
      A(1)=0.6
      A(2)=0.5
      A(3)=0.4
      A(4)=0.3

      do i=1,L
      ww(i)=w(i)
      enddo

  147 continue
      DO r=1,N
      if(ik.eq.5)ik=1
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
      DO  I=1,N
      D(I)=1.2E+38
      enddo
      D(r)=0

      METRHTHS=0
      u=r
      do  i=1,N
      Q(i)=1.2E+38
      enddo
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do i=1,L  !*RELAXATION STEP
        if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+w( i))then
              D(E( i))=D(S( i))+w( i)
              Q(E( i))=D(S( i))+w( i)
              Parent(E( i)) = i
              write(*,*)Parent(E(i))
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        endif
        enddo  !#END FIND NEXT SHORTEST PATH NODE
      Q(u)=0
      METRHTHS=METRHTHS+1
      ENDDO

      Do lll=1,N
      It=0
      k=lll
  119 continue
        if(k.ne.r.and.Parent(k).ne.0)then
        It=It+1
        p(lll,It)=Parent(k)
        k=S(Parent(k))
        go to 119
        endif
      RR(lll)=It
      Enddo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0

      do i=1,N
      SumP(i)=0
      enddo
      do i=1,L
      ha(i)=0
      enddo
      do i=1,N
      do j=1,N
      T(i,j)=0
      enddo
      enddo

      METRHTHS=0
      u=r
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
```

```
    DO WHILE(METRHTHS/=N)
      do i=1,L  !*RELAXATION STEP
      if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
            D(E( i))=D(S( i))+ww( i)
            Q(E( i))=D(S( i))+ww( i)
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    enddo

    do i=1,L
    zz=0.1*(D(E(i))-D(S(i))-ww(i))
    if(D(E(i))>D(S(i)))ha(i)=ha(i)+2.71828**zz
    if(D(E(i))>D(S(i)))ha(i)=ha(i)+0
    SumP(E(i))=SumP(E(i))+ha(i)
    enddo

    DO lll=1,N
         if(lll.ne.r.and.Demand(r,lll).ne.0)then
                 do i=1,N
                 T(r,i)=0
                 enddo
         T(r,lll)=(1-A(ik))*(Demand(r,lll)/4)

    do i=N,1,-1
    do j=1,L
      if(E(j).eq.i.and.T(r,E(j)).ne.0)then
      IF(SumP(E(j)).ne.0)Flow(j)=Flow(j)+T(r,E(j))*(ha(j)/SumP(E(j)))
      if(SumP(E(j)).ne.0)T(r,S(j))=T(r,E(j))*(ha(j)/SumP(E(j)))
      write(*,*)T(r,S(j)),T(r,E(j)),S(j),E(j)
      endif
    enddo
    enddo
         endif
      Do i=1,N
      T(r,i)=0
      Enddo

    ENDDO
      do lll=1,N
      if(RR(lll).ne.0)then
      do i=1,RR(lll)
      Flow(p(lll,i))=Flow(p(lll,i))+A(ik)*(Demand(r,lll)/4)
      if(lll.eq.6)HH=HH+A(ik)*(Demand(r,lll)/4)
      write(*,*)HH,p(lll,i)
      enddo
      endif
      enddo
    ik=ik+1
    ENDDO

    do i=1,L
    ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4))
    enddo
    iz=iz+1
    if(iz.ne.4)go to 147
```

```
      Do i=1,L
      write(*,*)S(i),'-',E(i),Flow(i),ww(i)
      AA=AA+ww(i)
      Enddo
      write(*,*)HH,AA/L    !Φόρτιση και Μέσος Χρόνος Διάνυσης Συνδέσμων
      END Subroutine AytonomhDromologhsh
      END MODULE

      PROGRAM AytonomhDromologhsh
      use AytonomhDromologhsh_Mod
      IMPLICIT NONE
      INTEGER N,L
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='XronosDianyshs1.f')
      OPEN(7,FILE='Zhthsh.f')
      OPEN(9,FILE='EquilibriumOutput.f')
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D)
      Call AytonomhDromologhsh(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM AytonomhDromologhsh
```

Αλγόριθμος φόρτισης ενός δυναμικού δικτύου υπό την προϋπόθεση ότι όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης, τα οποία ακολουθούν την κεντρική δρομολόγηση και ο κεντρικός διαχειριστής προτείνει διαδρομές σύμφωνα με την ισορροπία του χρήστη.
Επίλυση : Καθοδήγηση μεταφορικών μέσων και προσομοίωση καταμερισμού της ζήτησης σύμφωνα με την ισορροπία του χρήστη σε δυναμικά δίκτυα.

```
      MODULE UsersOptimum_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE UsersOptimum (S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q
      real,dimension(N,N)::Demand
```

```
    real,dimension(L)::Flow,ww
    integer,dimension(L)::p
    integer,dimension(N)::Parent
    integer::METRHTHS,u,XronosDianyshs
    H=0
    iz=0
    do i=1,L
    Flow(i)=0
    enddo
    do i=1,N
    read(7,*)(Demand(i,j),j=1,N) !Ζήτηση μεταξύ κάθε κόμβου Π-Π
    enddo

    do i=1,L
    ww(i)=w(i)
    enddo

    do ii=1,N
      do jj=1,N
125 continue
      if(Demand(ii,jj).ne.0.and.ii.ne.jj)then
      Demand(ii,jj)=Demand(ii,jj)-1  !Καταμερισμός των Μετακινήσεων

    r=ii

    DO 114 I=1,N
114 D(I)=1.2E+38
    D(r)=0

    METRHTHS=0
    u=r
    do 123 i=1,N
123 Q(i)=1.2E+38
    Q(r)=0
    DO WHILE(METRHTHS/=N)
      do i=1,L  !*RELAXATION STEP
      if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
            D(E( i))=D(S( i))+ww( i)
            Q(E( i))=D(S( i))+ww( i)
            Parent(E( i)) = i
      endif
      enddo   !#RELAXATION STEP ENDS
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    DO I=1,N
    WRITE(9,117)I,D(I)
117 FORMAT(3X,I4,24X,F12.3)
    enddo
    write(*,*)'Telos'
    XronosDianyshs=XronosDianyshs+D(jj)
    It=0
    k=jj
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      p(It)=Parent(k)
      k=S(Parent(k))
      go to 119
```

```
      endif
    do i=1,It
     Flow(p(i))=Flow(p(i))+1
    enddo

    do i=1,L
    ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4)) !Χρήση συνάρτησης BPR
    enddo
         if(Demand(ii,jj).ne.0.and.ii.ne.jj)go to 125
    endif
    Enddo
    ENDDO

    write(9,*)'Συνολικός Χρόνος Διάνυσης μέχρι στιγμής=',XronosDianyshs
    Do i=1,L
    write(9,*)S(i),'-',E(i),Flow(i),ww(i)
    AA=AA+ww(i)
    Enddo
    write(9,*)'Διαμορφωμένος μέσος όρος χρόνων διάνυσης συνδέσμων',AA/L

    END Subroutine UsersOptimum
    END MODULE

    PROGRAM UsersOptimum
    use UsersOptimum_Mod
    IMPLICIT NONE
    INTEGER N,L
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    integer, allocatable:: S(:),E(:)
    real, allocatable :: w(:),D(:)
    OPEN(8,FILE='xRONOS dIANYSHS MIKRO DIKTYO.f')
    OPEN(7,FILE='zHTHSH MIKRO DIKTYO.f')
    OPEN(9,FILE='EquilibriumOutput.f')
    WRITE(9,118) result
118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    Call Initialization(S,E,w,D)
    Call UsersOptimum (S,E,w,D)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    END PROGRAM UsersOptimum
```

Αλγόριθμος φόρτισης ενός δυναμικού δικτύου υπό την προϋπόθεση ότι όλοι οι χρήστες διαθέτουν συστήματα πλοήγησης, τα οποία ακολουθούν την κεντρική δρομολόγηση και ο κεντρικός διαχειριστής προτείνει διαδρομές σύμφωνα με την ισορροπία του συστήματος.
Επίλυση : Καθοδήγηση μεταφορικών μέσων και προσομοίωση καταμερισμού της ζήτησης σύμφωνα με την ισορροπία του συστήματος σε δυναμικά δίκτυα.

```
    MODULE SystemOptimum_Mod
    CONTAINS

    SUBROUTINE Initialization(S,E,w,D)
    INTEGER N,L
    COMMON /AREA1/ N,L
    integer, allocatable:: S(:),E(:)
    real, allocatable :: w(:),D(:)

    READ(8,*)N
    READ(8,*)L
    allocate(S(L),E(L),w(L),D(N))
```

```
      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE SystemOptimum(S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q,DD
      real,dimension(N,N)::Demand
      real,dimension(L)::Flow,ww,www
      integer,dimension(L)::p
      integer,dimension(N)::Parent
      integer::METRHTHS,u
      real::XronosDianyshs
      iz=0
      do i=1,L
      Flow(i)=0
      enddo
      do i=1,N
      read(7,*)(Demand(i,j),j=1,N)
      write(*,*)(Demand(i,j),j=1,N)
      enddo

      do i=1,L
      ww(i)=w(i)
      www(i)=w(i)
      enddo

      do ii=1,N
        do jj=1,N
  125 continue
        if(Demand(ii,jj).ne.0.and.ii.ne.jj)then
        Demand(ii,jj)=Demand(ii,jj)-1

      r=ii
      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0
      DO  I=1,N
      DD(I)=1.2E+38
      enddo
      DD(r)=0

      METRHTHS=0
      u=r
      do 123 i=1,N
  123 Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do i=1,L  !*RELAXATION STEP
        if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
              D(E( i))=D(S( i))+ww( i)
              DD(E( i))=DD(S( i))+www( i)
              Q(E( i))=D(S( i))+ww( i)
              Parent(E( i)) = i
        endif
        enddo   !#RELAXATION STEP ENDS
      ShortestDistance=1.2E+38     !*FIND NEXT SHORTEST PATH NODE
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
```

```
    ShortestDistance=D(i)
    u=i
      endif
      enddo  !#END FIND NEXT SHORTEST PATH NODE
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    XronosDianyshs=XronosDianyshs+DD(jj)

    It=0
    k=jj
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      p(It)=Parent(k)
      k=S(Parent(k))
      go to 119
      endif
    do i=1,It
     Flow(p(i))=Flow(p(i))+1
     write(9,*)Flow(p(i)),p(i),lll
    enddo

    do i=1,L
    ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4))+Flow(i)*((1+0.15*(Flow(i)/1
   100)**4)+w(i)*(0.6*Flow(i)**3/(100**4)))
    enddo

    do i=1,L
    www(i)=w(i)*(1+0.15*((Flow(i)/100)**4))  !Χρήση συνάρτησης BPR
    enddo

    if(Demand(ii,jj).ne.0.and.ii.ne.jj)go to 125
    endif
    Enddo
    ENDDO

    Do i=1,L
    write(*,*)'*',S(i),'-',E(i),Flow(i),w(i)*(1+0.15*((Flow(i)/100)**4
   1))
    AAA=AAA+www(i)
    Enddo
    write(*,*)XronosDianyshs
    WRITE(*,*)'Διαμορφωμένος μέσος όρος χρόνων διάνυσης συνδέσμων',AAA/L

    END Subroutine SystemOptimum
    END MODULE

    PROGRAM SystemOptimum
    use SystemOptimum_Mod
    IMPLICIT NONE
    INTEGER N,L
    REAL, DIMENSION(2) :: tarray
    REAL :: result
    integer, allocatable:: S(:),E(:)
    real, allocatable :: w(:),D(:)
    OPEN(8,FILE='xRONOS dIANYSHS MIKRO DIKTYO.f')
    OPEN(7,FILE='zHTHSH MIKRO DIKTYO.f')
    OPEN(9,FILE='EquilibriumOutput.f')
    WRITE(9,118) result
118 FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
    Call Initialization(S,E,w,D)
    Call SystemOptimum(S,E,w,D)
    CALL ETIME(tarray, result)
    WRITE(9,118) result
    END PROGRAM SystemOptimum
```

Αλγόριθμος για τον καταμερισμό της ζήτησης σε δυναμικά δίκτυα με στόχο την ισορροπία του χρήστη μέσω σταδιακής φόρτισης.
Επίλυση : Ροές ως προς το χρήστη, χρησιμοποιείται για την επαλήθευση των αποτελεσμάτων που προκύπτουν εάν ο κεντρικός διαχειριστής προτείνει τη συντομότερη διαδρομή που θα ακολουθήσει κάθε μεταφορικό μέσο.

```
      MODULE UOIncrementalLoading_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE UOIncrementalLoading(S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q
      real,dimension(N,N)::Demand
      real,dimension(L)::Flow,ww
      integer,dimension(L)::p
      integer,dimension(N)::Parent
      integer::METRHTHS,u
      H=0
      iz=0
      do i=1,L
      Flow(i)=0
      enddo
      do i=1,N
      read(7,*)(Demand(i,j),j=1,N)
      enddo
      do i=1,L
      ww(i)=w(i)
      enddo

  147 continue
      DO r=1,N

      DO 114 I=1,N
  114 D(I)=1.2E+38
      D(r)=0

      METRHTHS=0
      u=r
      do 123 i=1,N
```

```
123 Q(i)=1.2E+38
    Q(r)=0
    DO WHILE(METRHTHS/=N)
      do i=1,L  !*RELAXATION STEP
      if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
            D(E( i))=D(S( i))+ww( i)
            Q(E( i))=D(S( i))+ww( i)
            Parent(E( i)) = i
      endif
      enddo
    ShortestDistance=1.2E+38
      do i=1,N
      if(Q(i)/=0.and.D(i)<ShortestDistance)then
    ShortestDistance=D(i)
    u=i
      endif
      enddo
    Q(u)=0
    METRHTHS=METRHTHS+1
    ENDDO
115 CONTINUE
    DO I=1,N
    WRITE(9,117)I,D(I)
117 FORMAT(3X,I4,24X,F12.3)
    enddo

    Do lll=1,N
    It=0
    k=lll
119 continue
      if(k.ne.r.and.Parent(k).ne.0)then
      It=It+1
      p(It)=Parent(k)
      k=S(Parent(k))
      go to 119
      endif
    do i=1,It
     Flow(p(i))=Flow(p(i))+Demand(r,lll)/60  !Φότιση δικτύου σε 60 στάδια
     write(9,*)Flow(p(i)),p(i),lll
    enddo

    IF(D(lll).ne.1.2E+38)H=H+D(lll)*Demand(r,lll)

    write(9,*)'Diadromh'
    write(9,*)(S(p(j)),E(p(j)),j=1,It),'It='
    do i=1,L
    ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4))
    enddo

    Enddo
    ENDDO

    iz=iz+1
    if(iz.ne.60)go to 147

    Do i=1,L
    write(9,*)S(i),'-',E(i),Flow(i),ww(i)
    AA=AA+ww(i)
    Enddo
    write(*,*)AA/L
    write(*,*)H/60

    END Subroutine UOIncrementalLoading
    END MODULE

    PROGRAM UOIncrementalLoading
    use UOIncrementalLoading_Mod
```

```
      IMPLICIT NONE
      INTEGER N,L
      REAL, DIMENSION(2) :: tarray
      REAL :: result
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)
      OPEN(8,FILE='xRONOS dIANYSHS MIKRO DIKTYO3.f')
      OPEN(7,FILE='zHTHSH MIKRO DIKTYO2.f')
      OPEN(9,FILE='EquilibriumOutput.f')
      WRITE(9,118) result
  118 FORMAT(39HΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
      Call Initialization(S,E,w,D)
      Call DijkstraAlgorithm(S,E,w,D)
      CALL ETIME(tarray, result)
      WRITE(9,118) result
      END PROGRAM UOIncrementalLoading
```

Αλγόριθμος για τον καταμερισμό της ζήτησης σε δυναμικά δίκτυα με στόχο την ισορροπία του συστήματος μέσω σταδιακής φόρτισης.
Επίλυση : Ροές ως προς το σύστημα, χρησιμοποιείται για την επαλήθευση των αποτελεσμάτων που προκύπτουν εάν ο κεντρικός διαχειριστής προτείνει τη διαδρομή που θα ακολουθήσει κάθε μεταφορικό μέσο με στόχο την ισορροπία του συστήματος.

```
      MODULE SOIncrementalLoading_Mod
      CONTAINS

      SUBROUTINE Initialization(S,E,w,D)
      INTEGER N,L
      COMMON /AREA1/ N,L
      integer, allocatable:: S(:),E(:)
      real, allocatable :: w(:),D(:)

      READ(8,*)N
      READ(8,*)L
      allocate(S(L),E(L),w(L),D(N))

      DO I=1,L
      READ(8,*)S(I),E(I),Number,w(I)
      ENDDO

      CLOSE(8)
      END SUBROUTINE Initialization

*     Main Loop (Step 1)
      SUBROUTINE SOIncrementalLoading(S,E,w,D)
      INTEGER N,L,r
      COMMON /AREA1/ N,L
      integer, dimension(:)::S,E
      real,dimension(:)::w,D
      real,dimension(N)::Q,DD
      real,dimension(N,N)::Demand
      real,dimension(L)::Flow,ww,www
      integer,dimension(L)::p
      integer,dimension(N)::Parent
      integer::METRHTHS,u
      H=0
      iz=0
      do i=1,L
      Flow(i)=0
      enddo
      do i=1,N
```

```
      read(7,*)(Demand(i,j),j=1,N)
      enddo
      do i=1,L
      ww(i)=w(i)
      www(i)=w(i)
      enddo

147   continue
      DO r=1,N

      DO 114 I=1,N
114   D(I)=1.2E+38
      D(r)=0
      DO  I=1,N
      DD(I)=1.2E+38
      enddo
      DD(r)=0

      METRHTHS=0
      u=r
      do 123 i=1,N
123   Q(i)=1.2E+38
      Q(r)=0
      DO WHILE(METRHTHS/=N)
        do i=1,L
        if(S(i).EQ.u.AND.D(E( i)).gt.D(S( i))+ww( i))then
              D(E( i))=D(S( i))+ww( i)
              DD(E( i))=DD(S( i))+www( i)
              Q(E( i))=D(S( i))+ww( i)
              Parent(E( i)) = i
        endif
        enddo
      ShortestDistance=1.2E+38
        do i=1,N
        if(Q(i)/=0.and.D(i)<ShortestDistance)then
      ShortestDistance=D(i)
      u=i
        endif
        enddo
      Q(u)=0
      METRHTHS=METRHTHS+1
      ENDDO
115   CONTINUE
      DO I=1,N
      WRITE(9,117)I,D(I)
117   FORMAT(3X,I4,24X,F12.3)
      enddo

      Do lll=1,N
      It=0
      k=lll
119   continue
        if(k.ne.r.and.Parent(k).ne.0)then
        It=It+1
        p(It)=Parent(k)
        k=S(Parent(k))
        go to 119
        endif
      do i=1,It
       Flow(p(i))=Flow(p(i))+Demand(r,lll)/60 !Φόρτιση δικτύου σε 60 στάδια
       write(9,*)Flow(p(i)),p(i),lll
      enddo

      write(9,*)'Diadromh'
      write(9,*)(S(p(j)),E(p(j)),j=1,It),'It='

      do i=1,L
```

```
     ww(i)=w(i)*(1+0.15*((Flow(i)/100)**4))+Flow(i)*((1+0.15*(Flow(i)/1
    100)**4)+w(i)*(0.6*Flow(i)**3/(100**4)))
     enddo

     do i=1,L
     www(i)=w(i)*(1+0.15*((Flow(i)/100)**4))
     enddo
     Enddo

     ENDDO
     iz=iz+1
     if(iz.ne.60)go to 147

     Do i=1,L
     write(*,*)'*',S(i),'-',E(i),Flow(i),w(i)*(1+0.15*((Flow(i)/100)**4
    1))
     AAA=AAA+www(i)
     Enddo
     WRITE(*,*)AAA/L  !Μέσος Όρος Χρόνων Διάνυσης Συνδέσμων

     END Subroutine SOIncrementalLoading
     END MODULE

     PROGRAM SOIncrementalLoading
     use SOIncrementalLoading_Mod
     IMPLICIT NONE
     INTEGER N,L
     REAL, DIMENSION(2) :: tarray
     REAL :: result
     integer, allocatable:: S(:),E(:)
     real, allocatable :: w(:),D(:)
     OPEN(8,FILE='XronosDianyshs6.f')
     OPEN(7,FILE='Zhthsh.f')
     OPEN(9,FILE='EquilibriumOutput.f')
     WRITE(9,118) result
118  FORMAT(39ΗΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ sec : ,F50.40)
     Call Initialization(S,E,w,D)
     Call SOIncrementalLoading(S,E,w,D)
     CALL ETIME(tarray, result)
     WRITE(9,118) result
     END PROGRAM SOIncrementalLoading
```