20/7/2015

Πολυτεχνείο Κρήτης

ΤΜΗΜΑ:
ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΠΡΟΗΓΜΕΝΟΙ ΑΛΓΟΡΙΘΜΟΙ ΕΛΕΓΧΟΥ ΟΔΙΚΗΣ ΚΥΚΛΟΦΟΡΙΑΣ ΣΕ ΑΣΤΙΚΑ ΔΙΚΤΥΑ

Διπλωματική εργασία :
Δημήτρης Χαραλαμπίδης

Επιβλέπων:

- Παπαγεωργίου Μάρκος

Μέλη επιτροπής:

- Παπαμιχαήλ Ιωάννης
- Νικολός Ιωάννης

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον κ. Μάρκο Παπαγεωργίου, επιβλέπων καθηγητή της διπλωματικής μου εργασίας, για την άψογη συνεργασία μας και για την υποστήριξη που μου προσφέρθηκε κατά την εκπόνηση της εργασίας αυτής.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη τους καθόλο το διάστημα των σπουδών μου και ιδιαίτερα την αδερφή μου Κατερίνα, η οποία με στήριξε όποτε αυτό χρειάστηκε χωρίς ενδοιασμούς.

Ακόμη μια αναφορά θα ήθελα να κάνω στην φίλη μου Ειρήνη γιατί μου έδωσε το τελικό «σπρώξιμο» που χρειαζόμουν.

Περιεχόμενα

Περίληψη

Ο σύγχρονος τρόπος ζωής και η περεταίρω οικοδόμηση πόλεων με κακές οδικές υποδομές ευθύνονται επί το πλείστον για τη δημιουργία της κυκλοφοριακής συμφόρησης. Η κυκλοφοριακή συμφόρηση είναι ένα σύγχρονο πρόβλημα που εμφανίζεται σε πολλά μέρη του κόσμου, ιδιαιτέρως στα πιο μεγάλα αστικά κέντρα, αποτελώντας έτσι ένα παγκόσμιο πρόβλημα. Έχει αντιμετωπιστεί στο παρελθόν με διαφόρους τρόπους (σήμανση, τροχονόμους κτλ) αλλά δεν εφαρμόστηκε ποτέ κάποιος αρκετά ικανός, ώστε να εξαλείψει το πρόβλημα ολοσχερώς. Προσαρμόζοντας την σήμανση στη ροη της κυκλοφορίας και παίρνοντας σωστές αποφάσεις για την προτεραιότητα των οχημάτων, έχει αποδεχτεί ότι είναι πιο αποτελεσματικός, πιο οικονομικός και συνάμα ο πιο "πράσινος" τρόπος αντιμετώπισης.

Σε αυτή την πτυχιακή εργασία μελετήθηκε, μια τέτοια μέθοδο δυναμικού έλεγχου της κυκλοφορίας. Η SchIC (Schedule-driven intersection control), μια μέθοδο η οποία δημιουργήθηκε το 2012 μέσω της συνεργασίας του Robotics Institute του Carnegie Mellon University και του Department of Industrial Engineering and Logistics Management του Hong Kong University of Science and Technology, αξιοποιώντας ότι πληροφορίες προκύπτουν από την δομή μίας μη ομοιόμορφα κατανεμημένης ροής της κυκλοφορίας, δίνει λύση στον εκθετικό χαρακτήρα του συνόλου των συνδυασμών των φάσεων μιας διασταύρωσης σε έναν ορισμένο ορίζοντα βελτιστοποίησης. Βασικό κομμάτι της μεθόδου είναι ο εναλλακτικός τρόπος μοντελοποίησης της βελτιστοποίησης του έλεγχου μιας διασταύρωσης, σαν ένα πρόβλημα προγραμματισμού εργασιών, πετυχαίνοντας έτσι σημαντικές μειώσεις στο πλήθος των λύσεων. Αποτέλεσμα είναι σχεδόν βέλτιστες λύσεις με πολυωνυμική πολυπλοκότητα στον ορίζοντα πρόβλεψης χωρίς να παρουσιάζεται ευαισθησία προς τον χρόνο. Για την δομή του αλγορίθμου δημιουργήθηκε ένα πρόγραμμα προσομοίωσης μιας διασταύρωσης, με διάφορα επίπεδα ροής , ώστε να παρατηρηθεί η αποτελεσματικότητα του αλγορίθμου στο κάθε ένα και ίσως τα όριά του.

Εισαγωγή

Ο έλεγχος της κίνησης των οχημάτων στα δίκτυα αυτοκινητοδρόμων είναι ένα πρόβλημα πολλών διαστάσεων. Η κυκλοφορική συμφόρηση, πέρα από ότι προφανώς καθυστερεί τα ραντεβού και τις υποχρεώσεις του καθενός, έχει επιπλέον αντίκτυπο στην ψυχολογία των ανθρώπων, στα οικονομικά τους αλλά και στο περιβάλλον. Όταν κάποιος καθυστερεί για μια υποχρέωσή του, νιώθει αποτυχία κατηγορώντας ίσως έτσι τον εαυτό του , ρίχνοντας έτσι την ψυχολογία του αδίκως. Αρκετά σημαντική επίσης είναι και η οικονομική πλευρά του προβλήματος, καθώς με τις αυξήσεις στις τιμές του πετρελαίου και την αυξημένη κατανάλωση καύσιμου από το σταμάτα-ξεκίνα στα κεντρικά σημεία των πόλεων, πολύ περισσότερα χρήματα καταναλώνονται, υπερχρεώνοντας τους οδηγούς για την επιπλέον καθυστέρηση και τις χαμηλότερες ταχύτητες.

Τέλος όσο αφορά το οικολογικό κομμάτι, κάθε άλλο παρά καλό είναι η συγκέντρωση οχημάτων για αρκετό χρόνο σε ένα σημείο, που ειδικότερα στα κέντρα είναι χωρίς πράσινο, το οποίο θα απορροφούσε ένα μέρος των ρύπων που κάνουν την

ατμόσφαιρα αποπνικτική, ακόμα και τοξική. Η πιο συχνή λύση που εφαρμόζεται για την επίλυση τέτοιων προβλημάτων είναι η κατασκευή περιφερειακών δρόμων και η προσαρμογή της σήμανσης για την τροφοδοσία αυτών. Τέτοια έργα προφανώς όμως και δεν είναι απλά, ενώ σε αρκετές περιπτώσεις δεν είναι πραγματοποιήσιμα λόγο χώρου και ταυτοχρόνως κοστίζουν αρκετά χρήματα για την εκτέλεση τους και αρκετό χρόνο για να περατωθούν.

Γι' αυτό και συχνά διακόπτονται, ενώ μερικά δε τελειώνουν ποτέ, δημιουργώντας περεταίρω προβλήματα και χρέη. Προς αυτό το σκοπό όμως έχουν αναπτυχτεί τεχνολογίες, μέθοδοι και τεχνικές τέτοιες ώστε να κάνουν το πρόβλημα της χειραγώγησης της κυκλοφορίας, ένα πρόβλημα με λύση, όπου κάποιες φορές αυτή η λύση θα επιφέρει απλώς ένα πολύ καλύτερο αποτέλεσμα αλλά, κάποιες άλλες φορές θα επιφέρει ολική απαλοιφή της συμφόρησης ενός δικτύου. Μια τέτοια μέθοδο είναι και η SchIC, όπου αναλύεται παρακάτω σε αυτή τη διπλωματική εργασία.

1.Παρουσίαση προβλήματος

Οι βασικές τεχνικές επίλυσης τέτοιων προβλημάτων είναι γενικά μέθοδοι εύρεσης βέλτιστου χρόνου αλλαγής φάσεων των ελεγκτών κυκλοφορίας, όπου κλειδί είναι το εύρος της αναζήτησης και ο χρόνος εκτέλεσης της. Όλες οι μοντελοποιήσεις που επιχειρούν να λύσουν τέτοια προβλήματα έχουν ως βασικό πρόβλημα την διαχείριση υπολογιστικού φόρτου. Η καθολική αναζήτηση πιθανών λύσεων σε έναν αρκετά εκτενή ορίζοντα πρόβλεψης (Hp) είναι πρακτικά ανέφικτη καθώς το σύνολο του πλήθους των λύσεων (state-space) μεγαλώνει εκθετικά ως προς τον αριθμό των βημάτων (time steps) που χωρίζουμε τον Hp (Papageorgiou et al., 2003).

Μέθοδοι εξαντλητικής αναζήτησης και branch-and-bound (Robertson and Brethendon, 1974 / Porche and Lafortune, 1999 / Shelby, 2004) απαιτούν υπερβολικά αρκετό χρόνο για ρεαλιστικά σενάρια. Έρευνες έχουν δείξει πως για να γίνουν αρκετές υπάρχοντες μέθοδοι εφαρμόσιμες θα πρέπει να καταφύγουν σε λύσεις όπως σμίκρυνση του Hp, σε ελάττωση του αριθμού αλλαγής φάσεων, ή και σε παραπλήσιες μεθόδους επίλυσης (απαλοιφή παραπλήσιων λύσεων κτλ), όπως προσεγγιστικές λύσεις της συνάρτησης αξιολόγησης και σε αιρετικές αναζητήσεις, που όμως και οι δυο έχουν αποδείξει ότι επηρεάζουν αρκετά την ποιότητα των λύσεων (σχετικά μακρινά αποτελέσματα από τη βέλτιστη λύση).

-1.1 Δυναμική Λύση(SchIC)

Με δεδομένα όλα τα παραπάνω για την μέθοδο SchIC επιδόθηκε μεγάλη βαρύτητα στην παραγωγή σχεδόν βέλτιστων λύσεων σε πραγματικό χρόνο. Αξιοποιώντας δομικές πληροφορίες της ροής επιτυγχάνει αποτελεσματική μείωση στο σύνολο του πλήθους των λύσεων εντός Hp. Επίσης η SchIC μετασχηματίζει το πρόβλημα ελέγχου διασταύρωσης σε πρόβλημα εκτέλεσης εργασιών, βασισμένο σε μια συνολική αναπαράσταση της ροής δεδομένων και χρησιμοποιεί την μη-ομοιόμορφη κατανομή της ροής δεδομένων για την δημιουργία ενός μικρότερου πλήθους λύσεων, το οποίο παρόλα αυτά περιέχει ακόμα ικανοποιητικές λύσεις. Εν τέλη η πολυπλοκότητα του παραγόμενου μοντέλου είναι πολυωνυμική ως προς τον Hp, γεγονός που δεν ανεβάζει τον υπολογιστικό φόρτο τόσο όσο άλλες μέθοδοι. Για την εύρεση της βέλτιστης ανάμεσά στις πιθανές λύσεις, χρησιμοποιείται ένα αποτελεσματικό κριτήριο ελαχιστοποίησης των λύσεων που ερευνούνται μειώνοντας τον φόρτο σημαντικά.

2.Μοντελοποίηση

-2.1 Πίνακας αναπαράστασης ("a")

Η πρώτη μοντελοποίηση που πρέπει να αναλυθεί είναι του ορίζοντα πρόβλεψης και της διασταύρωσης. Αρχικά, γνωρίζοντας την απόσταση των φωρατών από το stop_line και την ταχύτητα που θεωρητικά ακολουθούν τα οχήματα που κυκλοφορούν εντός συστήματος, μπορούμε με απλή διαίρεση να βρούμε τον ορίζοντα πρόβλεψης(HP = Distance / Speed). Ο ορίζοντας πρόβλεψης Hp είναι ο χρόνος μέσα στον οποίο παίρνουμε μετρήσεις από το σύστημα. Τα δεδομένα που λαμβάνονται δεν είναι συνεχείς. Η λήψη των μετρήσεων γίνεται ανά μια σταθερή μονάδα χρόνου, την μεταβλητή samp. Έτσι, τα χρονικά πεδία που μπορούμε να αντιληφτούμε ένα όχημα είναι Hp / samp.

Η βασική μοντελοποίηση της εισόδου των οχημάτων γίνεται σε clusters. Η αρχική αποθήκευση όμως γίνεται στον πίνακα αναπαράστασης ροής (πίνακας "a") ο οποίος αποθηκεύεται και ανανεώνεται σε κάθε επανάληψη ώστε να χρησιμοποιηθεί ξανά έπειτα λειτουργώντας ως μνήμη της ροής μας. Είναι ένας απλός δυσδιάστατος πίνακας όπου η κάθε γραμμή αναπαριστά μια φάση (η φάση μπορεί να αποτελείται από μια οδό ή από παραπάνω οδούς οι οποίες λειτουργούν ταυτόχρονα και έτσι θεωρητικά έχουν κοινή είσοδο, έξοδο και ουρά) του συστήματος και έχει αριθμό στηλών ίσο με τα steps που χωρίσαμε τον ορίζοντα Hp. Κάθε στοιχειό του δείχνει τον

αριθμό των οχημάτων ανά φάση που απέχουν αριθμό steps (τα steps είναι σε seconds) από την έξοδο του συστήματος που βρίσκεται ο ελεγκτής της κυκλοφορίας, ίσο με τον αριθμό στήλης τους .

Π.χ.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0

1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0

Εδώ βλέπουμε 3 φάσεις ενός συστήματος όπου η πρώτη είναι εντελώς άδεια, ενώ η δεύτερη έχει αρκετά οχήματα και ιδιαίτερα 3 στην πρώτη στήλη. Όπως είπαμε η πρώτη σημαίνει απόσταση 0 από την έξοδο άρα είναι σαν τρεις οδηγοί που περιμένουν να ξεκινήσουν. Σημαντικό είναι ότι για να συμπεριληφθεί και ο ανθρώπινος παράγοντας στις MATLAB προσομοιώσεις που ακολουθούν, θεωρήθηκε ότι μπορεί να μην ξεκινήσει για κάποιο λόγο ένας οδηγός και να μην έχουμε ενδεχομένως έξοδο εκείνη τη χρονική στιγμή.

Σε κάθε ανανέωση (σε κάθε time-step), αφού θεωρείται σταθερή η ταχύτητα προσχωρούν ένα step τη φορά. Έτσι σε μια πιθανή επανάληψη ένα step μετά μπορεί να έχουμε :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

2 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0

2 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0

Όπου εδώ παρατηρείται πως τα οχήματα έχουν προχωρήσει όλα ένα time step και πιο συγκεκριμένα έχει φύγει ένα από την δεύτερη φάση, έχει μπει ένα στην πρώτη και στην τρίτη δεν έφυγε κανένα αλλά έγιναν 2 αυτά που αναμένουν στην έξοδο.

-2.2 Πίνακας των Clusters("C")

Η τελική μορφοποίηση των δεδομένων γίνεται από τα στοιχεία του πίνακα "a" στον πίνακα των clusters "C". Ο πίνακας αυτός είναι τρισδιάστατος, όπου κάθε δυσδιάστατος υπό-πίνακας του αφορά τα clusters μιας φάσης, όπου εκεί κάθε σειρά αποτελείται από πληροφορίες για κάθε ένα cluster και ο αριθμός των σειρών ισούται με τα steps που χωρίσαμε τον Hp (αυτό σημαίνει ότι ο κάθε δυσδιάστατος πίνακας έχει τον ίδιο αριθμό clusters, καθώς μετά τα clusters συμπληρώνεται με μηδενικές γραμμές (δεν υπάρχει cluster)).

Το μήκος των στηλών είναι σταθερά 5 καθώς τόσα είναι τα δεδομένα που χρειαζόμαστε για να προσδιοριστεί το κάθε ένα cluster. Με τη σειρά η πρώτη στήλη κρατά τον αριθμό των οχημάτων του cluster ("col"), η δεύτερη τα seconds που απέχει από την έξοδο ("arr") του συστήματος (τα όποια μπορούν να πάρουν προφανώς μόνο τιμές πολλαπλάσιες του τυχών time step), η τρίτη τα seconds που χρειάζονται για να περάσουν την έξοδο ("dep") του συστήματος, η τέταρτη την διαφορά dep-arr ("dur") που εκφράζει τη διάρκεια του cluster και η πέμπτη τον μέσο ρυθμό ροής του cluster ("fr") όπου είναι το πηλίκο col/dur. Το fr είναι σημαντικό λόγω του ότι clusters με μεγάλο fr είναι πιο φρόνιμο να εξυπηρετούνται πρώτα, αποφεύγοντας έτσι υψηλή συνολική καθυστέρηση στην αντίστοιχη ουρά (μεγάλα intergreen και απορρόφηση χρόνων start-up πάνω σε αλλαγές της ενεργούς φάσης).

Π.χ.

1	0	2	2	0.5
1	4	6	2	0.5
1	6	8	2	0.5
0	0	0	0	0
0	0	0	0	0

Στο παραπάνω παράδειγμα βλέπουμε έναν υποπίνακα cluster μιας τυχαίας φάσης. Στη πρώτη γραμμή έχουμε 1 αμάξι που απέχει 0 seconds από την έξοδο και θα την περάσει σε 2 seconds με διάρκεια 2 seconds και fr 0.5 v/s (vehicle/second). Αντίστοιχα στην δεύτερη γραμμή ένα άλλο cluster με 1 αμάξι το οποίο απέχει όμως 4 seconds και παρόλα αυτά καταλήγει με ίδιο fr και ομοίως στη τρίτη άλλο ένα που απέχει 6 seconds. Το σύνολο των γραμμών είναι 5 που σημαίνει πως έχουμε χωρίσει

τον Hp σε 5 steps και βλέποντας τα υπόλοιπα στοιχεία το step είναι 2 seconds και άρα και ο Hp είναι 10 seconds.

Η μοντελοποίηση όμως δε τελειώνει εκεί καθώς αν συγκεντρωθούν οχήματα στην έξοδο, θεωρούμε ότι αποτελούν ουρά. Σε αυτή την περίπτωση εκείνο το cluster έχει arr=0, fl = "sfr" (saturation flow rate (δεδομένο)) dep = col/sfr και dur = col/sfr.

Επίσης οχήματα που βρίσκονται χρονικά κοντά μεταξύ τους, συμφέρει αρκετές φορές να ενώνονται σε ένα Cluster. Αυτό μπορεί να παραχθεί σε 2 περιπτώσεις, πρώτον αν η διαφορά του arr από 2 clusters είναι μικρότερη από το κατώφλι που έχουμε ορίσει "thc" (Threshold-based clustering) και δεύτερων αν ένα προσερχόμενο cluster φτάνει στην έξοδο πριν να αδειάσει όλη η ουρά που περιμένει να εξυπηρετηθεί (Anticipated queue clustering, δηλαδή αν το προσερχόμενο cluster έχει arr <= dep του cluster που είναι πρώτο).

1) Threshold-based clustering

Σε αυτή τη περίπτωση αξιοποιώντας μια οχηματο-κεντρική λογική (VA) (Papageorgiou et al., 2003) εξυπηρετούνται οχήματα τα όποια βρίσκονται χρονικά εντός ενός κρίσιμου χρονοδιαστήματος. Το νέο cluster θα έχει φυσικά col = Σ(col) των clusters, arr = min (arr(1), arr(2)), dep = max (dep(1), dep(2)) και τα υπόλοιπα προκύπτουν φυσιολογικά από τους κανονικούς τύπους.

Στο προηγούμενο παράδειγμα το δεύτερο και το τρίτο cluster βρίσκονται αρκετά κοντά. Έτσι αν θεωρηθεί thc = 2 μπορούμε να παράξουμε :

1	0	2	2	0.5
2	4	8	4	0.5
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

2) Anticipated queue clustering

Προϋποθέτοντας ότι υπάρχει μια ουρά που αναμένει εξυπηρέτηση, ένα προσδεχόμενο cluster ίσως έρθει αρκετά νωρίς ώστε να προλάβει την ουρά πριν αδειάσει και έτσι να ενωθούν (Lammer and Helbing, 2008). Εδώ όμως προκύπτουν 2 περιπτώσεις :

α) Αν το dep του προσερχόμενου (2) είναι μικρότερο ή ίσο του dep της ουράς(1), δηλαδή dep(2) <= dep(1) ή αν fr(2) >= fr(1), τότε το (2) ενώνεται εξ ολοκλήρου με την ουρά με c(1) = c (1) +(2).

β) Διαφορετικά μόνο ένα κομμάτι του (2) θα ενωθεί με το (1). Συγκεκριμένα υπολογίζουμε το δdur =(dep(1) - arr(2)) /(1 -(fr(2) / sfr)) με το οποίο ελέγχουμε αν, δdur >= dur(2). Αν ισχύει το (2) θα ενωθεί ξανά εξολοκλήρου με το (1) αλλιώς :

col(1) = col(1) + col(2) *(δdur / dur(2))

col(2) = col(2) - col(2) *(δdur / dur(2))

arr(2) = arr(2) + δdur

dep(2) = dep(2)

fr(2) = fr(2)

-2.3 Παραγωγή του πίνακα λύσεων (" S ")

Για την εξαγωγή όλων των πιθανών λύσεων χρησιμοποιούνται μόνο δεδομένα από τον πίνακα C που παράχθηκε παραπάνω. Η λογική είναι απλή καθώς το μόνο που πρέπει να γίνει είναι να δούμε χρονικά τα clusters ως διαφορετικές «εργασίες» που πρέπει να εξυπηρετηθούν από μια «μηχανή» (την διασταύρωση), χωρίς να μπορούν να εξυπηρετηθούν ταυτόχρονα. Κατά την επίλυση θεωρούμε ότι το κάθε cluster εξυπηρετείται ολόκληρο και στη συνέχεια ο προσομοιωτής ελέγχει τους περιορισμούς του προβλήματος (π.χ. μέγιστος κύκλος ελάχιστο και μέγιστο πράσινο μιας φάσης κλπ). Ο αλγόριθμός μας φτιάχνει όλους τους συνδυασμούς των cluster και έτσι δημιουργεί όλες οι πιθανές λύσεις (Scheduling).

Ειδική σημείωση μπορεί να γίνει εδώ για την "πολυπλοκότητα" του συστήματος. Το πόσες παράλληλες λύσεις μπορεί να έχουμε επηρεάζει την πολυπλοκότητα του προβλήματος και αφού μόνο μετά τη δημιουργία του πίνακα S μπορεί κανείς να ξέρει τον ακριβές αριθμό. Για αυτό το λόγο στον ορισμό του S χρησιμοποιείται ο μέγιστος

αριθμός παράλληλων λύσεων που μπορεί να δημιουργηθεί κάθε φορά και ισούται με το παραγοντικό του αριθμού των φάσεων (" I "), υψωμένο στον μέγιστο αριθμό των clusters ανά φάση. Ένας τρόπος σχετικής μείωσης του υπολογιστικού φόρτου είναι είτε να μειώσουμε τον αριθμό των cluster εκμεταλλευόμενοι την παράμετρο thc όπως θα δούμε και στα αποτελέσματά μας, είτε να κάνουμε χρήση ενός forward recursive algorithm για την επιλογή του καλύτερου πλάνου χωρίς να χάνουμε σε αποτελεσματικότητα.

Π.χ. αν έχουμε τα πλάνα

1 2 1 3

1 2 3 1

2 1 1 3

2 1 3 1

Αν το 1 2 μας οδηγεί σε μικρότερη καθυστέρηση από το 2 1, τα τελευταία δύο πλάνα δεν χρειάζεται να ερευνηθούν περισσότερο.

3.Βελτιστοποίηση

Για την εύρεση της βέλτιστης επιλογής ανάμεσα στις τόσες λύσεις πρέπει να υπολογιστεί η αξία της καθεμιάς. Προς αυτή τη διεύθυνση θα αναλυθεί αρχικά η δημιουργία κάποιων βοηθητικών πινάκων.

-3.1 Πίνακας χρόνου αλλαγής φάσης(" MinSwitch ")

Με τον πίνακα αυτόν αναζητάμε το χρονικό κόστος της αλλαγής, από τη μια φάση στην άλλη, λαμβάνοντας υπόψη τους χρόνους : ελάχιστο πράσινο ("Gmin") και χρόνο προειδοποίησης (" Y ").

Π.χ.

Έστω 2 φάσεις 1, 2.

MinSwitch= 0 Y

Y +(G + Y) 0

Όπου οριζόντια βρίσκονται οι φάσεις που είναι ενεργές και κάθετα οι φάσεις που πρέπει να ενεργοποιηθούν. Παρατηρούμε πόσο πιο χρονοβόρο είναι αν θέλουμε να δώσουμε ξανά πράσινο σε μια φάση, καθώς χρειαζόμαστε 2 χρόνους προειδοποίησης (" Y "), αφού θα ανοίξουμε την επόμενη φάση και θα της δώσουμε ένα ελάχιστο πράσινο (" Gmin ") και θα την κλείσουμε ξανά για την επιθυμητή.

-3.2 Πίνακας σύγκρισης λύσεων(" Sy ")

Αυτός ο πίνακας περιέχει τα χαρακτηριστικά της κάθε λύσης που έχουμε βάση των οποίων εκλέγεται η βέλτιστη. Ο πίνακας έχει τρεις διαστάσεις γιατί κρατά 3 χαρακτηριστικά του κάθε cluster της εκάστοτε λύσης. Τα 3 χαρακτηριστικά αφορούν το ποια φάση ενεργοποιείται (πρώτο ("s") χαρακτηριστικό), το χρόνο που χρειάζεται για να περατωθεί (δεύτερο ("t") χαρακτηριστικό) και τη συνολική καθυστέρηση όλων των cluster μέχρι να περατωθεί (τρίτο ("d") χαρακτηριστικό). Για να ξεκινήσει ο υπολογισμός όλων των τιμών είναι απαραίτητη η παρουσία μιας αρχικής καταχώρισης. Χρησιμοποιούνται τα στοιχεία της φάσης που βρίσκεται τώρα το σύστημα ("ic"), πριν αρχίσει η διαδικασία και εξ ορισμού είναι για κάθε λύση " n " Sy [0] [ ] [n] = { ic, 0, 0 }, όπου "n" μια τυχαία λύση στο σύνολο των "w" που έχουν υπολογιστεί.

-3.3 Πίνακας Θέσης(" Sx ")

Είναι ένας συμπληρωματικός πίνακας του S, όπου αποθηκεύει την οριζόντια θέση κάθε cluster στον πίνακα C της φάσης στην οποία ανήκει, συνδέοντας τους έτσι. Δημιουργείται παράλληλα με τον S και ως είναι λογικό έχουν το ίδιο μέγεθος.

-3.4 Αναζήτηση βέλτιστης λύσης (" sn ")

Όπως προαναφέρθηκε για την διαλογή της βέλτιστης λύσης θα πρέπει να συμπληρωθεί ο πίνακας Sy. Για αυτό πρέπει πρώτα να υπολογιστούν οι τιμές του επιτρεπτού χρόνου εκκίνησης ("pst"), του πραγματικού χρόνου εκκίνησης ("ast") και η τοπική συνολική καθυστέρηση ενός cluster ("δd").

Ξεκινώντας με μια πιθανή λύση, γνωρίζοντας την αρχική ("i", τελική φάση του προηγούμενου cluster) και την τελική φάση ("s") ώστε να φύγει το πρώτο της cluster, υπολογίζουμε

pst = t(i) + MinSwitch(i, s), ast = max(arr(του cluster), pst)

Εδώ αν το pst είναι μεγαλύτερο του arr (pst > arr) και ταυτοχρόνως δεν έχουμε συνέχεια της ιδίας φάσης (s i),

ast = ast + sult (χαμένος χρόνος κατά την εκκίνηση).

Τελικά

t = ast + dur(s) και δd = col(s) * max(ast - arr(s), 0) -> d = d + δd.

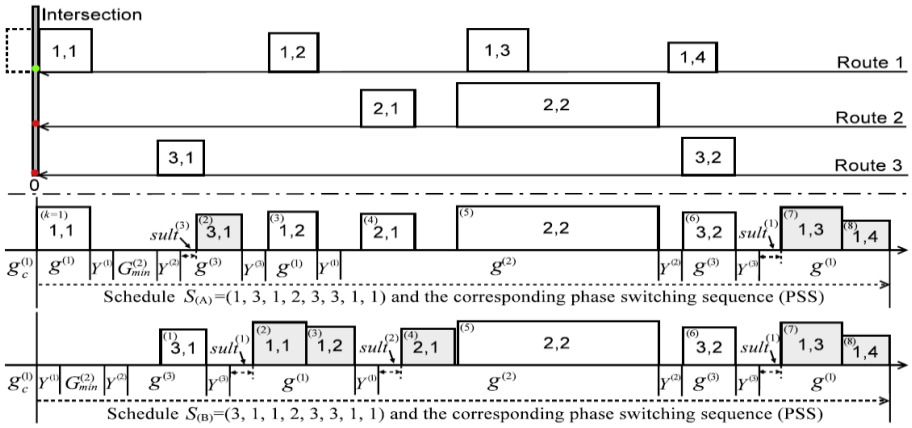
Επαναλαμβάνοντας όλα αυτά για κάθε cluster κάθε πιθανής λύσης εξάγονται όλα αυτά τα δεδομένα και μπορεί κανείς εύκολα να βρει την βέλτιστη. Αυτό γίνεται απλά ψάχνοντας στον πίνακα Sy για την λύση με το μικρότερο t και αν βρούμε παραπάνω λύσεις με το ελάχιστο t (Ελάχιστος χρόνος περάτωσης), τότε συγκρίνουμε τα d τους (ελάχιστη συνολική καθυστέρηση) και το μικρότερο είναι το βέλτιστο.

Εδώ βέβαια πρέπει να σημειωθεί ότι η βέλτιστη λύση δε θα εφαρμοστεί πλήρως. Από τη λύση θα εφαρμοστεί μόνο το πρώτο βήμα. Με λίγα λόγια εξυπηρετείται το πρώτο cluster και μετά γίνεται αναθεώρηση των δεδομένων. Αναλυτικότερα, η συνάρτηση καλείται ανά step για να γίνεται η αναθεώρηση των cluster. Όταν η τρέχουσα φάση φτάσει στο ελάχιστο πράσινο που δικαιούται (Gmin) ή έχει εφαρμοστεί το πρώτο βήμα της προηγούμενης απόφασής μας (έχει εξυπηρετηθεί το πρώτο cluster του προηγούμενου schedule), υπολογίζουμε το νέο βέλτιστο πρόγραμμα (schedule) φάσεων.

Αν η φάση που πρέπει να εξυπηρετηθεί σύμφωνα με τη λύση που προέκυψε είναι η τρέχουσα, τότε επεκτείνουμε την τρέχουσα φάση κατά τόσο χρόνο ώστε να εξυπηρετηθεί το πρώτο της εν αναμονή cluster, αλλιώς αλλάζουμε φάση και μόλις η νέα φάση φτάσει στο Gmin, ξανακάνουμε υπολογισμούς για να προκύψει το νέο schedule κ.ο.κ.

Ένα παράδειγμα scheduling και επιλογής του καλύτερου φαίνεται στο γράφημα 1 όπου φαίνονται οι 2 βέλτιστες λύσεις και το πώς έχουν προγραμματίσει χρονικά τα clusters.

Γράφημα 1: Scheduling



4.Περιορισμοί του αλγορίθμου

Όπως προαναφέρθηκε στην μοντελοποίηση του προβλήματος λήφθηκε υπόψη ο ανθρώπινος παράγοντας. Στα πλαίσια του να εξεταστεί ο αλγόριθμος σε -υπό ορούς πάντα- ρεαλιστικά σενάρια, παρθήκαν κάποια μέτρα ώστε να διαφυλαχτούν κάποιοι περιορισμοί που χαρακτηρίζουν την διαχείριση της ροής σήμερα.

Πρώτος εξ αυτών είναι η χρήση του ορίου Gmin, όπου ρόλος του είναι η διατήρηση της σειράς της εναλλαγής των φάσεων. Αυτό είναι σημαντικό διότι παρέχει την ασφάλεια ενός ελαχίστου χρόνου μέσα στον οποίο θα ενεργοποιηθεί μια φάση, ώστε ο επικείμενος οδηγός να ξέρει πως δε θα παραμεληθεί και ότι θα εξυπηρετηθεί κάποια στιγμή η φάση στην οποία βρίσκεται, ακόμα και αν δεν έχει μεγάλη πυκνότητα η ροη της. Επίσης με αυτόν τον τρόπο η μέθοδος διασφαλίζει ότι αν έχει μια φάση η οποία έχει αντικειμενικά αρκετά μεγαλύτερη ροη από της υπόλοιπες, θα εξυπηρετηθούν όλες ώστε να μην γίνουν μη λειτουργικές.

Ένας άλλος περιορισμός που διασφαλίζει πάλι αυτό το πρόβλημα είναι το Gmax. Το Gmax έχει του ρόλο του άνω ορίου στον greentime που μπορεί να έχει μια φάση. Έτσι πιο πυκνές φάσεις θα "δώσουν τη σκυτάλη" στις υπόλοιπες τελικά, διακόπτοντας τον πιθανό ατέρμονο κύκλο λειτουργίας τους.

Και όπως είναι γνωστό μεγάλο ρόλο παίζει η οδηγική συμπεριφορά στην ομαλή λειτουργία των δικτύων κυκλοφορίας. Γι' αυτό το λόγο συμπεριλήφθηκε, κατά την μοντελοποίηση των δεδομένων εισόδου, η υπόθεση πως παρόλο που ένα όχημα κινείται σταθερά και ακολουθεί της σημάνσεις μπορεί για οποιοδήποτε λόγο κάποια

στιγμή να μην γίνει αυτό. Έτσι θεωρήσαμε μια μεταβλητή "ex", η οποία ακολουθεί την κανονική κατανομή και η συνάρτηση πυκνότητας πιθανότητας της παίρνει τυχαίες τιμές . Έτσι θεωρήθηκε ότι με πιθανότητα 90% ο οδηγός θα εξέλθει κανονικά από την ροη, αλλιώς με πιθανότητα 10% θα παραμείνει στην ουρά για κάποιο τυχαίο λόγο.

Τέλος ακόμα ένα θέμα που μπορεί να επηρεάσει την ρεαλιστική οπτική του προβλήματος είναι η συμφόρηση στο δίκτυο. Ένα άδειο δίκτυο σίγουρα δε θα είναι τόσο δύσκολο στην διαχείριση του και για αυτό κατασκευάστηκαν διάφορα δεδομένα εισόδου. Από μια χαμηλή ροη, με 5% (αντιστοιχεί σε 200 vehicles/h) πιθανότητας εμφάνισης αυτοκινήτου ανά step, μέχρι μια υψηλή με 70%.

5.Δημιουργία Προσομοιωτή

Στη προσπάθεια να εξεταστεί η μέθοδος SchIC κατασκευάστηκε ένας αλγόριθμος προσομοίωσης. Βασικά στοιχεία του είναι ότι κρατά το ρολόι της προσομοίωσης, τους χρόνους " greenTime ", " phaseTime ", δίνει την είσοδο στον βασικό αλγόριθμο και ότι από την έξοδο του, αλλάζει ή όχι την φάση.

Η προσομοίωση γίνεται στην εικονική περίοδο της μιας ώρας (3600 επαναλήψεις), χρησιμοποιώντας όμως time step = 2 second η βασική λούπα θα εκτελεστεί 1800 φορές. Έτσι κάθε 2 επαναλήψεις ο προσομοιωτής θα καλέσει την βασική συνάρτηση δίνοντας της 2 βασικά στοιχεία, την είσοδο ("in") και τη φάση που βρίσκεται το σύστημα ("phase"). Η είσοδος είναι ένα διάνυσμα με μήκος ίσο με τον αριθμό των φάσεων, το οποίο περιέχει μηδενικά παντού εκτός από το κελί της φάσης που έχει εισερχόμενο όχημα. Για λόγους απλοποίησης η είσοδος των οχημάτων είναι μοναδιαία ανά step αλλά τυχαία ως προς τη φάση που εισέρχεται και έτσι δε μπορεί να υπάρχει πάνω από 1 όχημα που να εισέρχεται στο σύστημα. Ο προσομοιωτής δεν λαμβάνει υπόψη τη βέλτιστη λύση που προκύπτει από τον αλγόριθμο βελτιστοποίησης ("ext") προτού επιτευχθεί το phasetime Gmin, αλλά παρόλα αυτά πρέπει να την καλεί γιατί εκεί δημιουργείται ο πίνακας " a ", ο οποίος εκτυπώνεται σε ένα αρχείο "output.txt " για να έχει μνήμη στην επόμενη επανάληψη.

Αφού καλεστεί η συνάρτηση αρχίζει να ελέγχεται ο συνολικός χρόνος που έχει διανύσει η παρούσα φάση (" phaseTime "). Αν έχει περάσει τον " greenTime " (διάνυσμα με μήκος όσο ο συνολικός αριθμός των φάσεων, που κρατά τον greentime που έχει υπολογιστεί για την κάθε φάση και ο οποίος στην αρχή ισούται με Gmin για

κάθε φάση) που της αναλογούσε τότε αυτός επεκτείνεται κατά ext και αποθηκεύεται επίσης στην μεταβλητή " decision ".

Τώρα, αν η επέκταση ("ext") δεν είναι μηδενική, δηλαδή το σύστημα έχει προτείνει επιμήκυνση του greentime, αυτή θα είναι ίση με το χρόνο που χρειάζεται το πρώτο cluster για να εξυπηρετηθεί πλήρως. Επίσης, θα πρέπει να ελεγχθεί αν είναι δυνατή μια τέτοια ενέργεια. Με λίγα λόγια, μόλις το phaseTime ξεπερνά το Gmax περνάμε σε αλλαγή φάσης και ας μην έχει ολοκληρωθεί η απόφαση, μηδενίζοντας ταυτοχρόνως τον χρόνο φάσης phaseTime.

Στη περίπτωση που το ext είναι μηδενικό θα χρειαστεί αλλαγή φάσης. Η αλλαγή γίνεται μόνο εφόσον η τρέχουσα φάση έχει συμπληρώσει την ελάχιστη διάρκεια πράσινου Gmin που προαναφέρθηκε

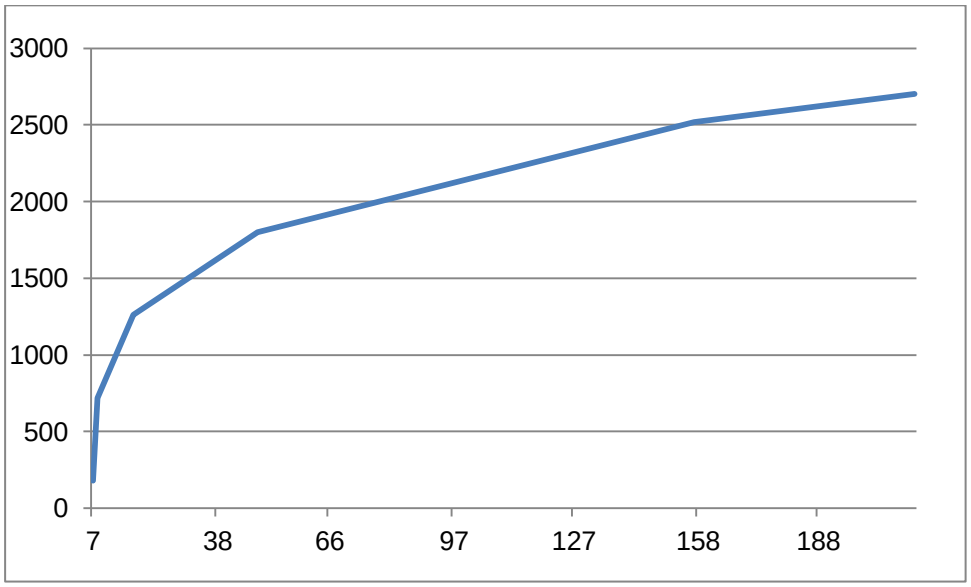
6.Εφαρμογή στον προσομοιωτή

Προς την εξερεύνηση των δυνατοτήτων της SchIC μεθόδου χρησιμοποιήθηκε μια διασταύρωση 3 φάσεων με ίδια ζήτηση. Εξετάστηκαν διάφορα σενάρια ροής ξεκινώντας από πιθανότητα εμφάνισης οχήματος ανά step (veh/timestep) 5% μέχρι 75%. Διερεύνηση πέρα του 75% κρίθηκε ανούσια, καθώς ο υπολογιστικός φόρτος εμπόδιζε την περάτωση της προσομοίωσης κάθε φορά. Πιο αναλυτικά, όπως έχει προαναφερθεί, ο χρόνος προσομοίωσης κάθε φορά ήταν 1 ώρα (3600 sec) με 22 timesteps των 2 sec σε Hp 45 sec. Επίσης θεωρήθηκε δημιουργία ουράς η συγκέντρωση πάνω από 3 οχήματα και τα υπόλοιπα χαρακτηριστικά του δικτύου είναι : Dist = 500 m (μήκος του HP), Speed = 40 km/h (σταθερή μέση ταχύτητα των οχημάτων), thc = 2 s, sfr = 2 s, sult = 2 s, Y =2 s, Gmin = 8 s.

Τα στοιχεία που εξετάστηκαν σε κάθε τρέξιμο ήταν ο χρόνος εκτέλεσης της προσομοίωσης, το μέγιστο μήκος ουράς που δημιουργείται και τέλος η ελάχιστη και η μεγίστη μνήμη σε Mb που απαιτείται, ως ένα δείκτη του φόρτου που χρειάζεται να υπολογιστεί.

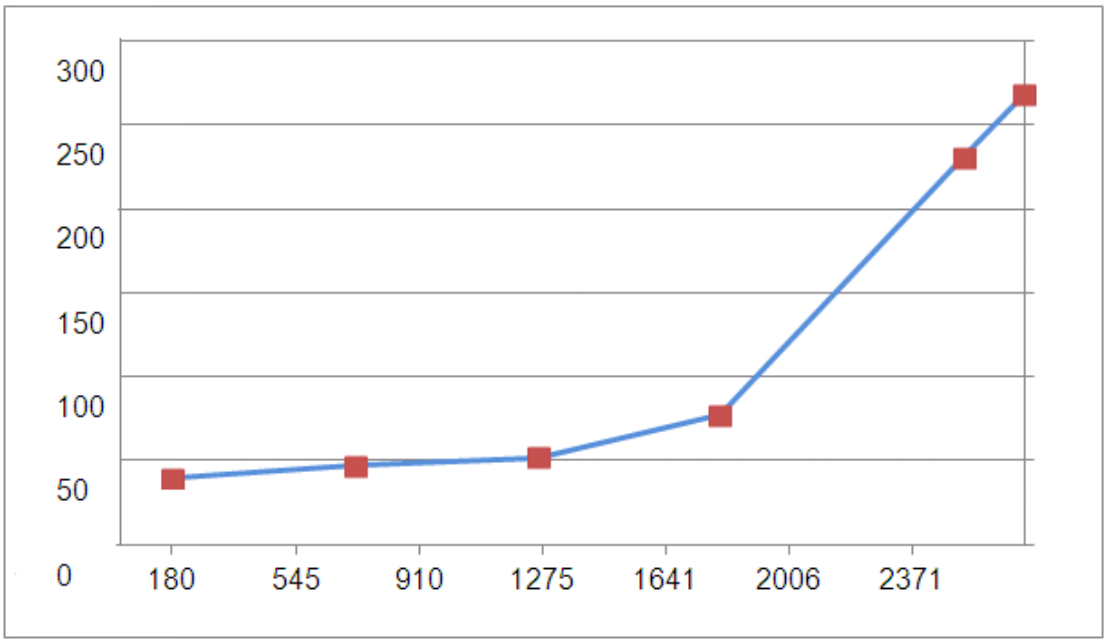
-6.1 Αποτελέσματα προσομοίωσης

Γράφημα 2: Ελάχιστη χρήση Ram(Mb) και veh/h



Κλίση ln(x)

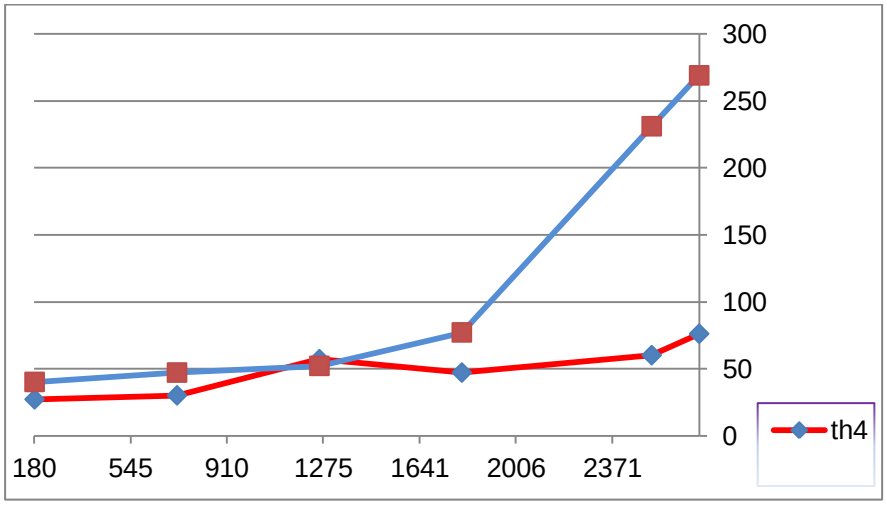
Γράφημα 3: Χρόνος εκτέλεσης(sec) και veh/h



Η αύξηση του χρόνου εκτέλεσης παρατηρούμε ότι ανεβαίνει εκθετικά σε σχέση με την ζήτηση στο σύστημα. Ο χρόνος εκτέλεσης επηρεάζεται πολύ από τον υπολογιστικό φόρτο, που με τη σειρά του επηρεάζεται άμεσα από την ζήτηση και ειδικότερα από τον αριθμό των clusters. Όσο αυξάνει η ζήτηση αυξάνει και ο αριθμός τους όπως είναι λογικό και αυτό μετά από ένα σημείο καθιστά δύσκολη την επίλυση του συστήματος καθώς η υπολογιστική ισχύς που χρειάζεται, αυξάνεται εκθετικά.

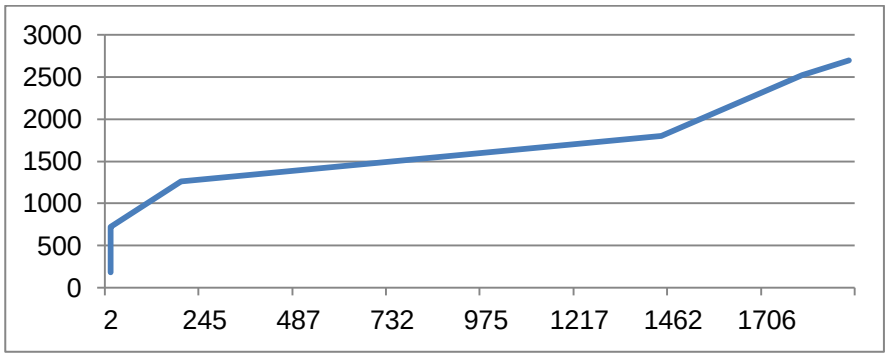
Μια τεχνική που μπορεί να γίνει ώστε να μειωθεί ο αριθμός των cluster, ανεξαρτήτως της ζήτησης, είναι η αύξηση του thc. Έτσι μεγαλώνοντας την χρονική απόσταση που χρειάζονται δύο cluster για να ενωθούν σε ένα cluster μειώνουμε δραματικά των αριθμό τους.

Γράφημα 4: Σύγκριση Χρόνου εκτέλεσης και veh/h με διαφορετικά thc



Απλά διπλασιάζοντας το thc βλέπουμε την διάφορα στο χρόνο επίλυσης, καθώς από εκθετικό έγινε σχεδόν γραμμικό(τουλάχιστον σε ζήτηση μέχρι 2700 v/h). Ταυτόχρονα δεν παρατηρήθηκαν αλλαγές στον μέσο χρόνο φάσης και στον μέγιστο μήκος ουράς.

Γράφημα 5: Μέγιστη χρήσης Ram(mb) και veh/h



Σε μικρές ροές δε σταθερή άνοδος μέχρι ένα σημείο και μετά απότομη αλλαγή της κλίσης.

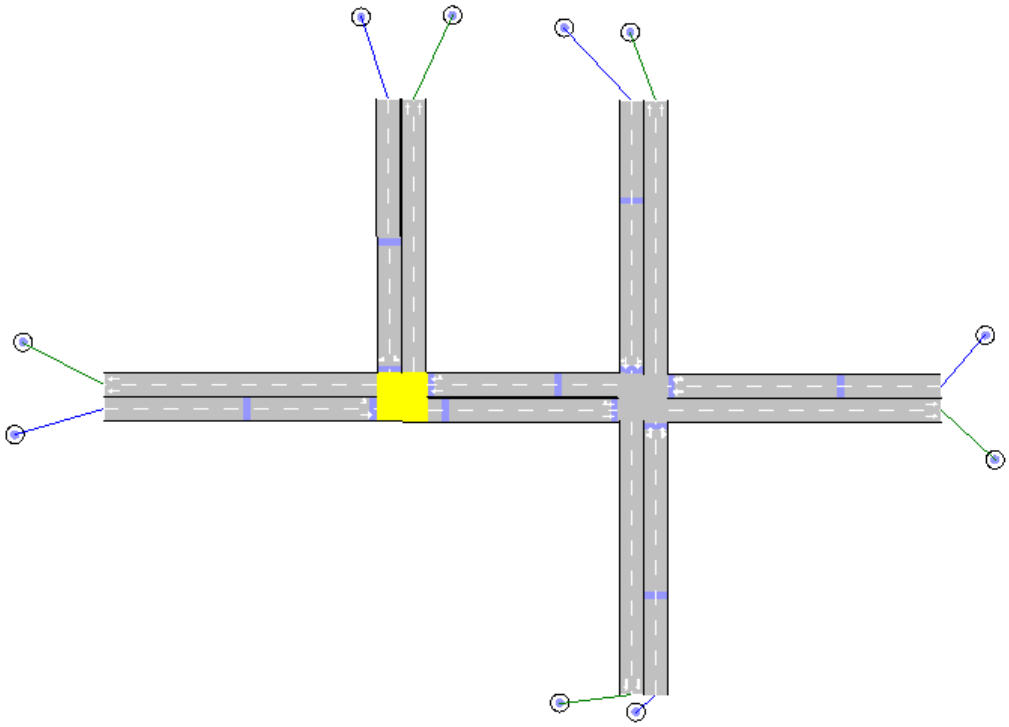
Η μνήμη που χρησιμοποιείται μετά τα 1800 veh/h είναι υψηλή για πιο πολύ χρονικό διάστημα, με αποτέλεσμα να ανεβάζει τον χρόνο εκτέλεσης. Στις μικρές ροές δε παρατηρούνται εναλλαγές στη χρήση μνήμης, αλλά σημαντικό είναι αυτό το φαινόμενο στις πιο υψηλές ροές, καθώς για πολύ χρόνο χρησιμοποιείται μεγάλα ποσά μνήμης ανεβάζοντας ακόμα πιο πολύ τον χρόνο εκτέλεσης. Όλο αυτό φτάνει συχνά τον υπολογιστή στα όρια μνήμης του, με αποτέλεσμα την προσομοίωση να σταματά, κάνοντας την συλλογή δεδομένων από δύσκολη μέχρι αδύνατη. Αυτός ήταν και ο λόγος που ελέγχθηκαν ροές μέχρι 75% καθώς πάνω από κει σταματούσε σχεδόν πάντα. Κάτι τέτοιο δεν παρατηρείται όμως στις αντίστοιχες περιπτώσεις όπου είχαμε θέσει thc = 4. Ειδικότερα η χρησιμοποιούμενη ram δεν ξεπέρασε τα 100 mb ακόμα και με ζήτηση 2700 v/h. Πιο αναλυτικά αυτό που αλλάζει το thc είναι το κατώφλι της απόστασης 2 οχημάτων μετά την οποία θεωρούνται 1 cluster.

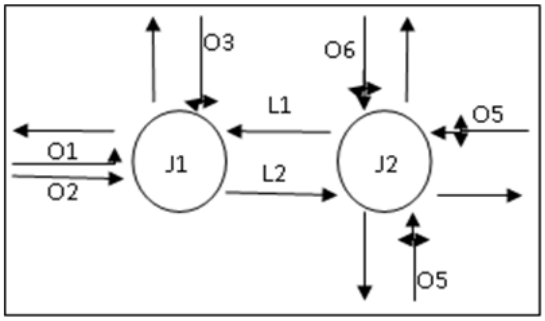
Έτσι αυξάνοντάς το πρακτικά, έχουμε λιγότερα cluster στο σύστημα για να ελέγξουμε και λιγότερες απαιτήσεις σε μνήμη. Μειώνεται όμως παρόλα αυτά η ευαισθησία του δικτύου και λόγο αυτού δε μπορεί να αυξάνει εις άπειρο ο δείκτης. Για θεωρητικά άπειρο thc και συνεχή ροή, στην ουσία θα είχαμε ένα cluster ανά φάση και θα προέκυπτε σταθερός κύκλος και μάλιστα ο μέγιστος δυνατός που θα ισούταν με το άθροισμα των Gmax.

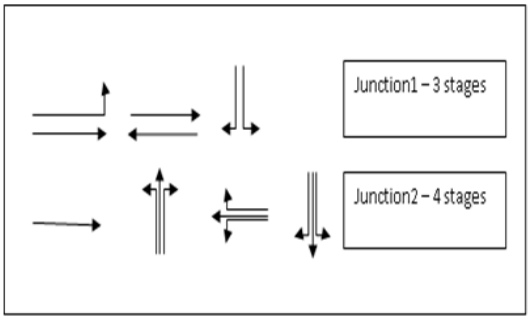
7.Προσομοίωση με χρήση μικροσκοπικού προσομοιωτή

Σε αυτό το σημείο θα παρουσιαστεί ένα παράδειγμα χρήσης της μεθόδου σε ένα ορισμένο δίκτυο 2 κόμβων σε περιβάλλον προσομοίωσης με τη χρήση του προγράμματος AIMSUM.

Σχήμα 1: Δίκτυο προσομοίωσης και οι φάσεις των κόμβων.







Ο πρώτος κόμβος αποτελείται από 3 οδούς διπλής κατεύθυνσης, που σημαίνει 3 εισόδους και 3 εξόδους από το σύστημα. Ο δεύτερος κόμβος αποτελείται από 4 οδούς διπλής κυκλοφορίας έχοντας μια οδό κοινή με τον πρώτο κόμβο. Οι μεταβλητές ορίστηκαν ως sfr = 0.8, sult = 3, Gmin = 6 και min_Cycle = 26. Η μεταβλητή Gmax, θέλοντας να εξεταστεί ένα πιο ρεαλιστικό σενάριο, ορίστηκε για τον πρώτο κόμβο ανά οδό 8 (οδός μειωμένης χρήσης), 66 (οδός μεγάλης χρήσης) ,30 και για τον δεύτερο 40, 24, 24, 24. Η απόσταση μεταξύ των κόμβων είναι στα 125 m και η αποστάσεις των φωρατών είναι στα 84 m στον πρώτο και 112 m στον δεύτερο αντίστοιχα. Μελετήθηκαν 3 σενάρια : thc = 2, thc = 4 και fixed_time.

Μια από τις βασικές παραδοχές που πρέπει να γίνουν για την εφαρμογή του αλγορίθμου Suptrac είναι ότι κάθε δρόμος εξυπηρετείται σε μία μόνο φάση. Αυτή η παραδοχή όμως δεν είναι κανόνας για τα αστικά δίκτυα. Όπως βλέπουμε στο σχήμα 1, ο δρόμος O2 εξυπηρετείται στις δύο πρώτες φάσεις του κόμβου 1. Αυτό που κάναμε για να μπορούμε να εφαρμόσουμε τον κώδικά μας είναι να υποθέσουμε ότι τα cluster του O2 αθροίζονται στα cluster της φάσης 2. Ο χρόνος πρασίνου της φάσης 1 αντιμετωπίζεται ως “bonus” του δρόμου O2. Αυτό δεν επηρεάζει τελικά τη λύση μας, μιας και οι μετρήσεις παίρνονται ανά δευτερόλεπτο και άρα στο τέλος της φάσης 1 ξέρουμε ακριβώς πόσα οχήματα βρίσκονται στον O2 ώστε να παρθεί η επόμενη μας απόφαση.

Στο συγκεκριμένο παράδειγμα επιλέξαμε μικρή ζήτηση στον O1 και την πρώτη φάση σταθερή και ίση με 6 δευτερόλεπτα που είναι αρκετή για να μην δημιουργηθούν ουρές. Ο λόγος που επιλέξαμε τον O2 να αθροίζεται ως προς τα cluster του με την φάση 2 είναι ότι, ο δρόμος O1 έχει μικρή ζήτηση σε σχέση με τους O2 και L1. Προσθέτοντας οδούς με μεγάλη ζήτηση ώστε να εξυπηρετούνται μαζί έχει νόημα, λόγω του ότι έτσι εξυπηρετούνται στο ίδιο GreenTime. Αντιθέτως αν ο O2 εξυπηρετούνταν στη φάση 1, ο δρόμος O1 θα είχε παραπάνω GreenTime από ότι θα μπορούσε να αξιοποιήσει.

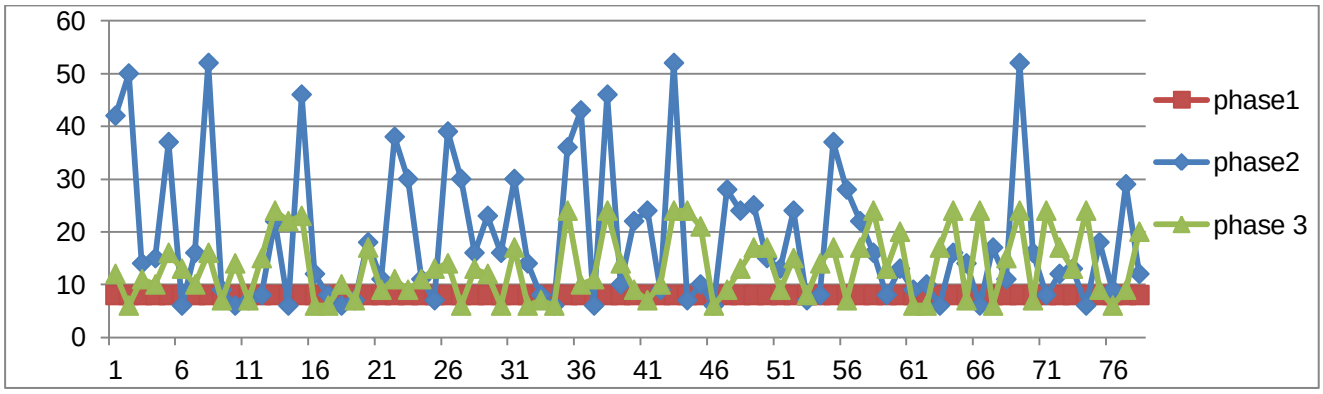
-7.1 Γραφήματα GreenTime / χρόνου

Παρακάτω παρουσιάζονται γραφήματα της διακύμανσης του GreenTime της προσομοίωσης στον κόμβο 1 ανά φάση με :

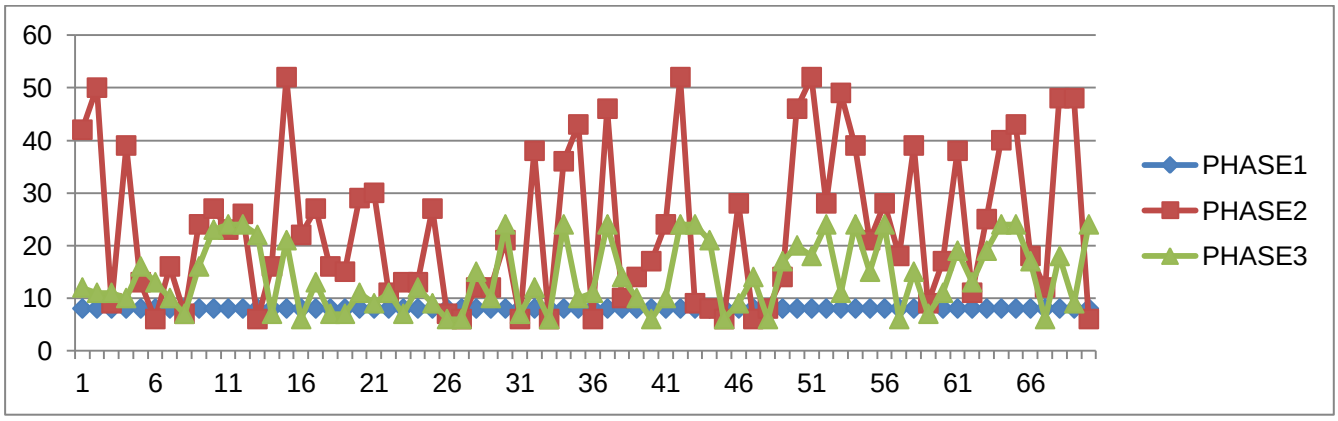
Πίνακας 1 : Gmax των φάσεων του κόμβου 1

	Phase 1	Phase 2	Phase 3
Gmax	8s	66s	30s

Γράφημα 9 : GreenTime της προσομοίωσης στον κόμβο 1 ανά φάση με **thc = 2**



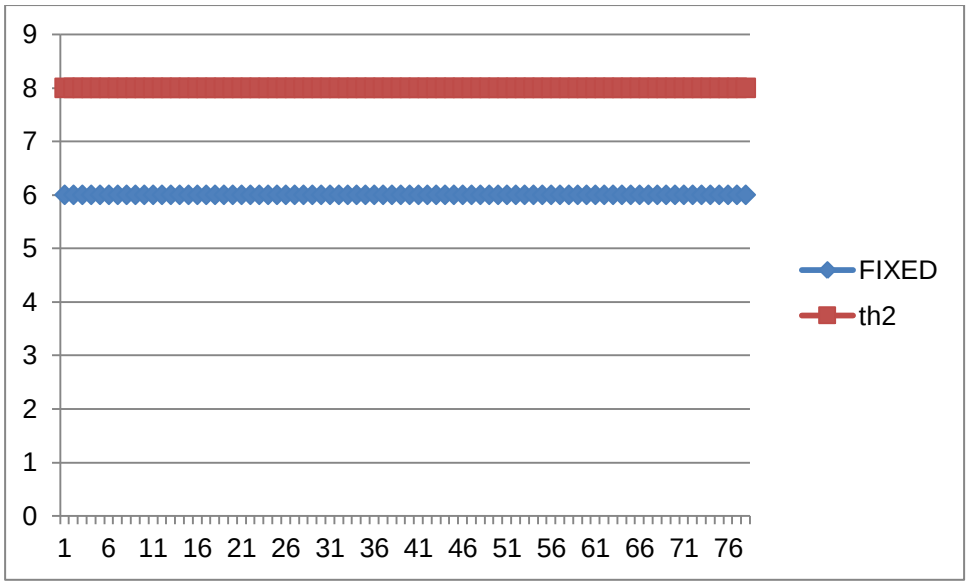
Γράφημα 9 : GreenTime της προσομοίωσης στον κόμβο 1 ανά φάση με **thc = 4**



Προχωρώντας γίνεται σύγκριση μεταξύ του αλγορίθμου SchIC και του Fixed_time ως προς το GreenTime που παραχωρεί η κάθε μέθοδος σε κάθε επιμέρους φάση.

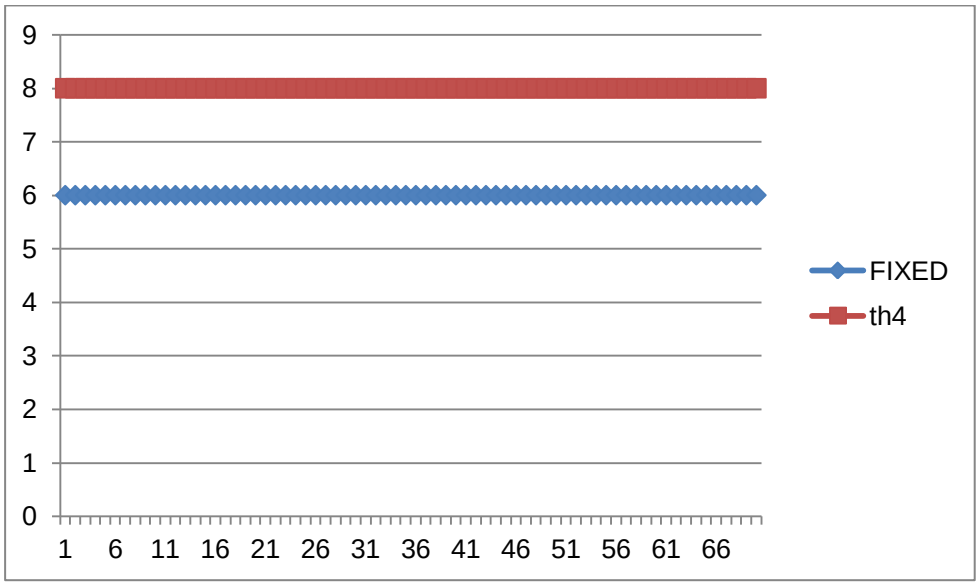
-Phase 1

Γράφημα 10 : GreenTime της SchIC και του Fixed_time της φάσης 1 με **thc = 2**



Η φάση 1 είναι σταθερή με Green_time 8 sec.

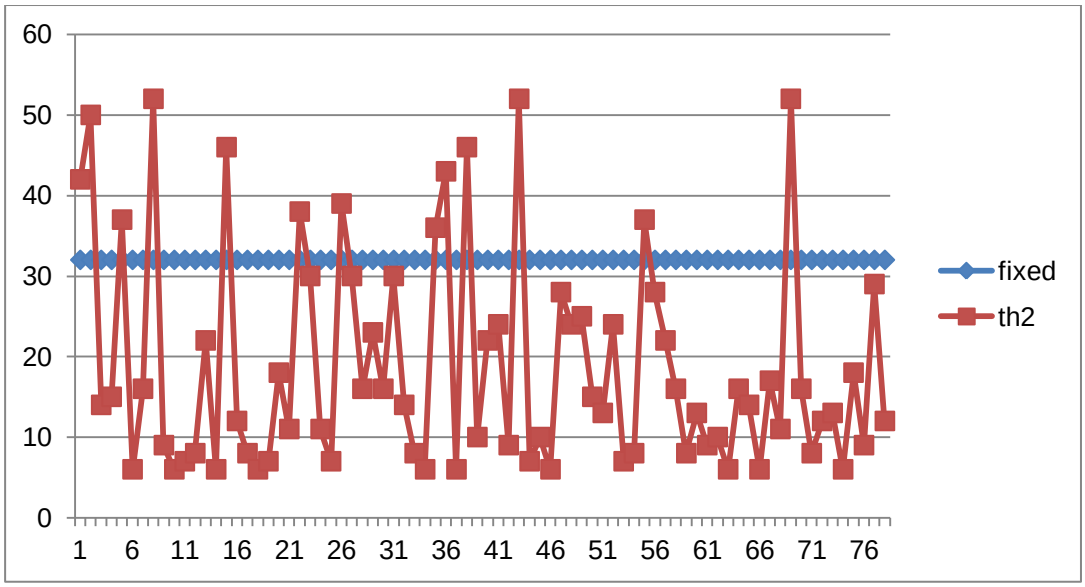
Γράφημα 11: GreenTime της SchIC και του Fixed_time της φάσης 1 με **thc = 4**



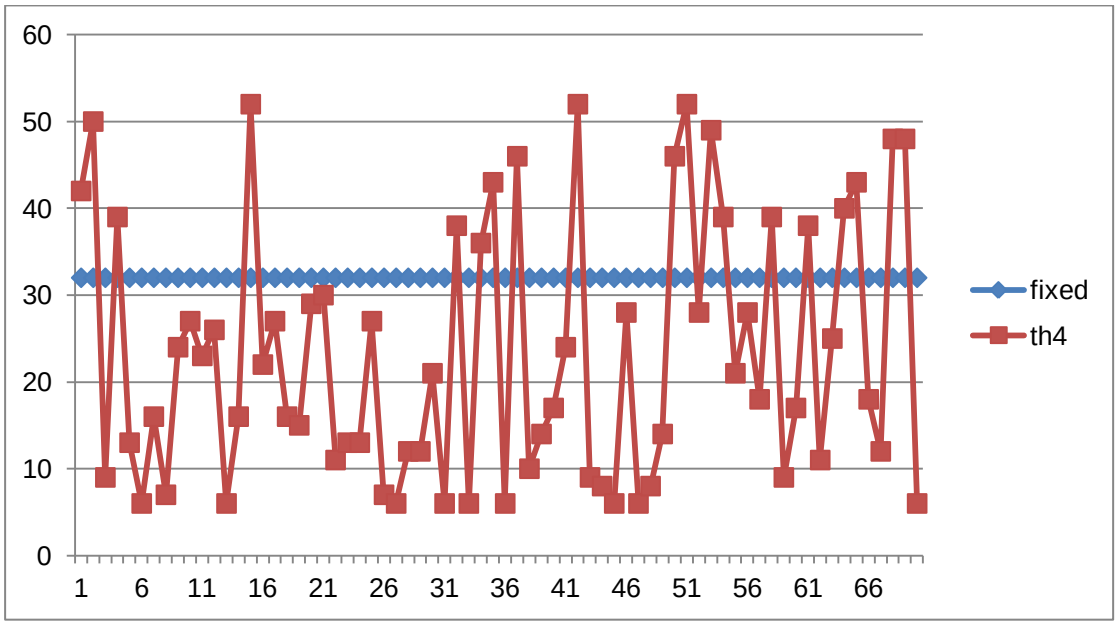
Όπως αναμένεται δεν υπάρχουν αλλαγές.

-Phase 2

Γράφημα 12 : GreenTime της SchIC και του Fixed_time της φάσης 2 με **thc = 2**



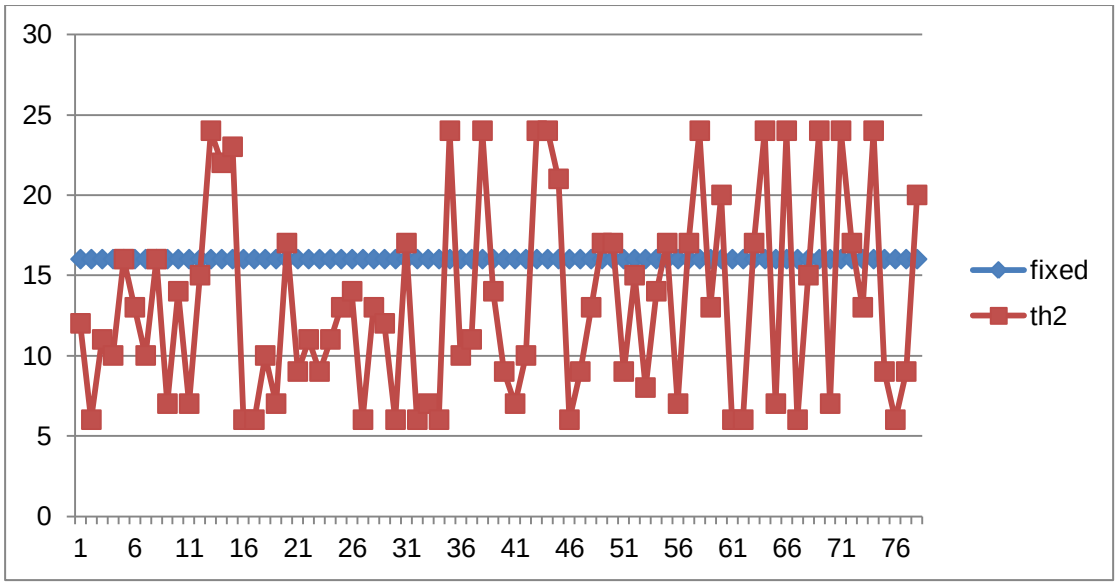
Γράφημα 13 : GreenTime της SchIC και του Fixed_time της φάσης 2 με **thc = 4**



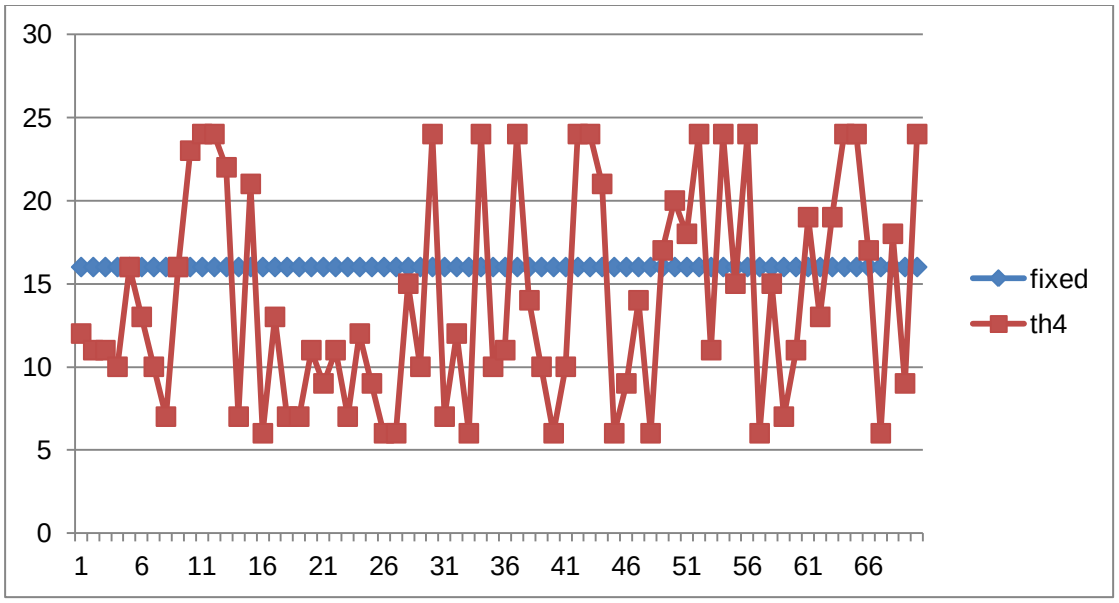
Εδώ πραγματικά αρχίζει να φαίνεται η διαφορά της μεθόδου SchIC και του fixed time, καθώς εύκολα παρατηρείται η οικονομία σε GreenTime και στις 2 περιπτώσεις. Με λίγα λόγια η ζήτηση εξυπηρετείται σε μικρότερους κύκλους.

-Phase 3

Γράφημα 14 : GreenTime της SchIC και του Fixed_time φάσης 3 με **thc = 2**

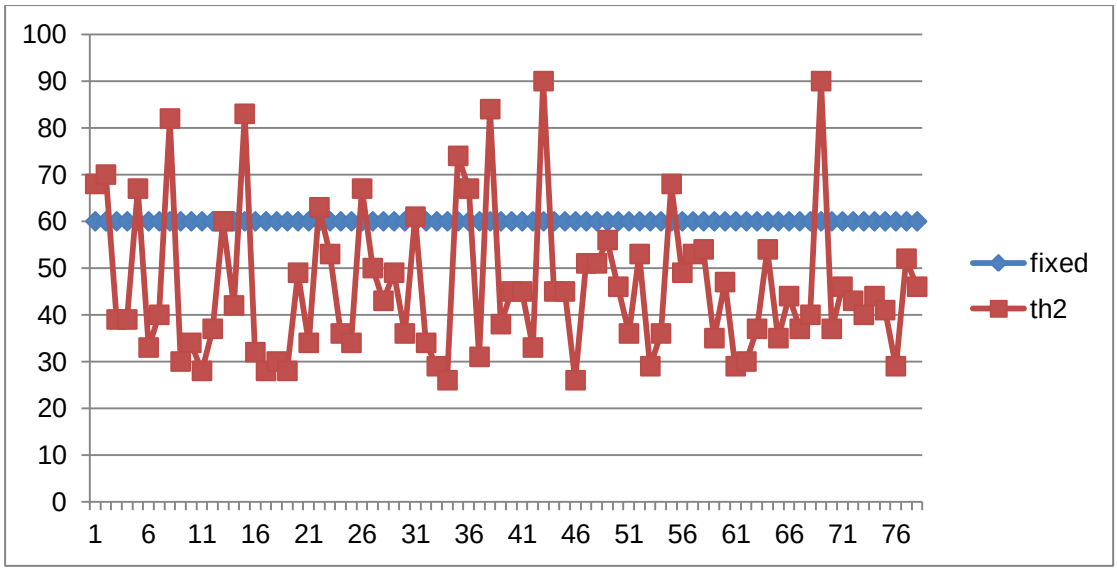


Γράφημα 15 : GreenTime της SchIC και του Fixed_time φάσης 3 με **thc = 4**



-Cycle

Γράφημα 16 : Κύκλος της SchIC και του Fixed_time με **thc = 2**



Στο γράφημα αυτό η γενικότερα μεγάλη διαφορά SchIC με fixed_time ως προς το GreenTime που παρέχουν σε όλες τις φάσεις του κόμβου.

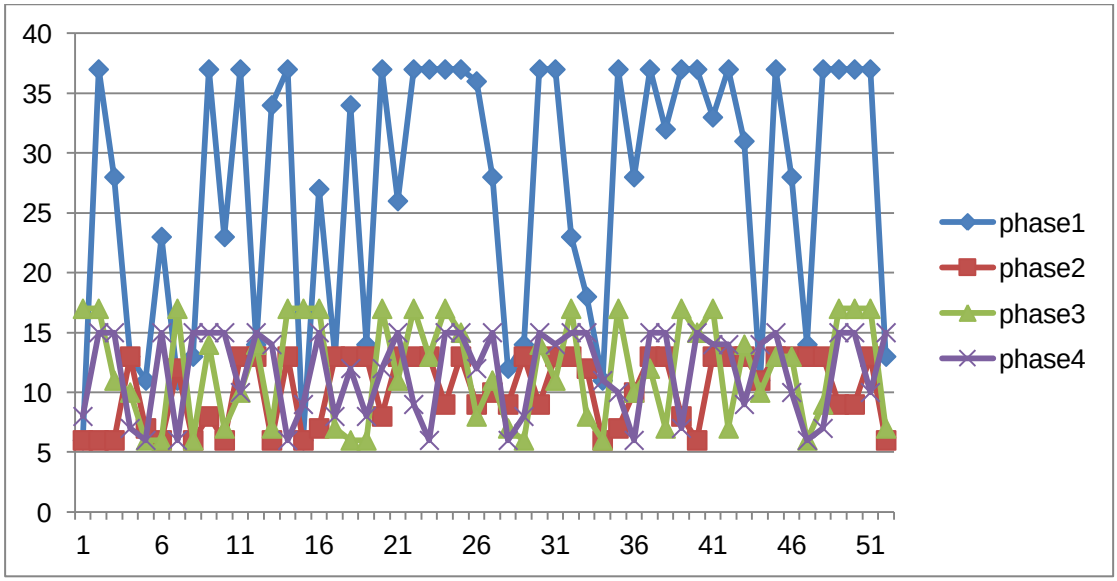
Κόμβος 2

Αντίστοιχα στον κόμβο 2 :

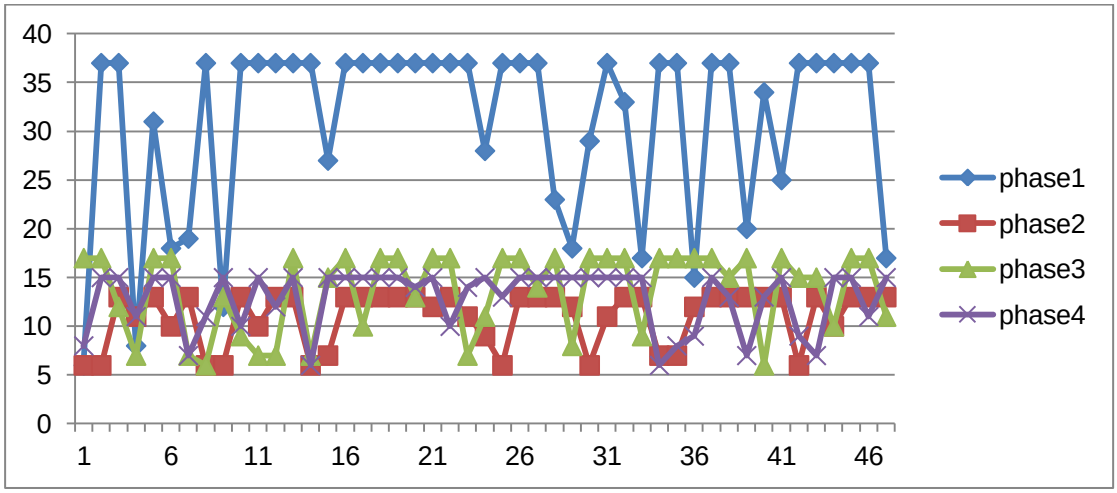
Πίνακας 2 : Gmax των φάσεων του κόμβου 2

	Phase 1	Phase 2	Phase 3	Phase 4
Gmax	40	24	24	24

Γράφημα 17 : GreenTime της προσομοίωσης στον κόμβο 2 ανά φάση με **thc = 2**



Γράφημα 18: GreenTime της προσομοίωσης στον κόμβο 2 ανά φάση με **thc = 4**

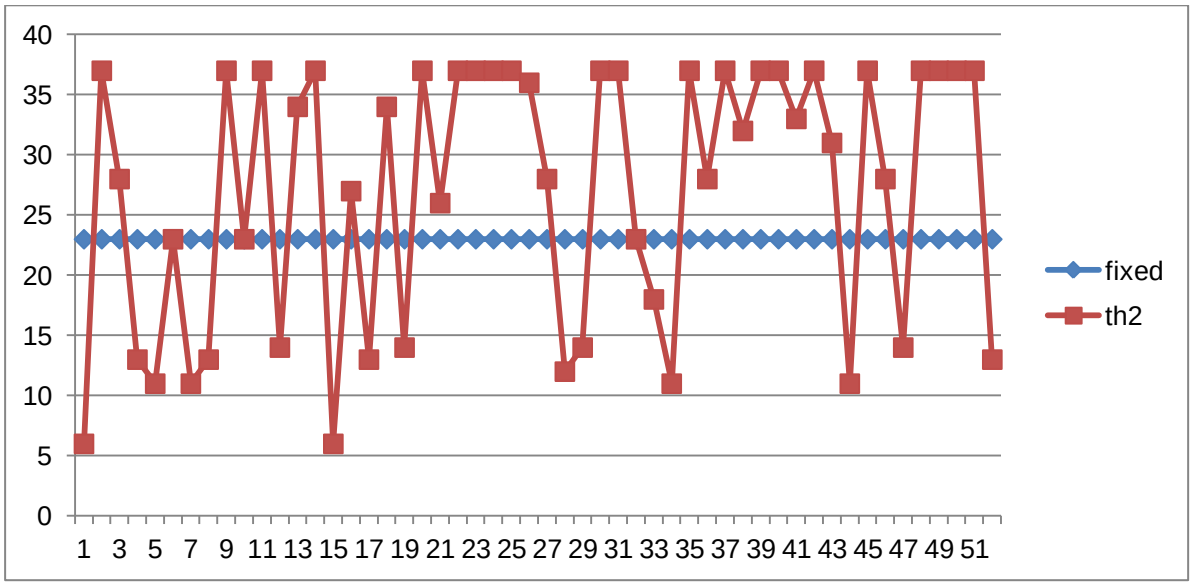


Παρατηρείται σε αυτό το γράφημα η διαφορά της χρήσης της πρώτης οδού σε σχέση με τις άλλες, καθώς πολύ συχνά δίνει χρόνο GreenTime κοντά στο Gmax, σε αντίθεση με τις άλλες οδούς οι οποίες σχεδόν ποτέ δε φτάνουν το δικό τους Gmax.

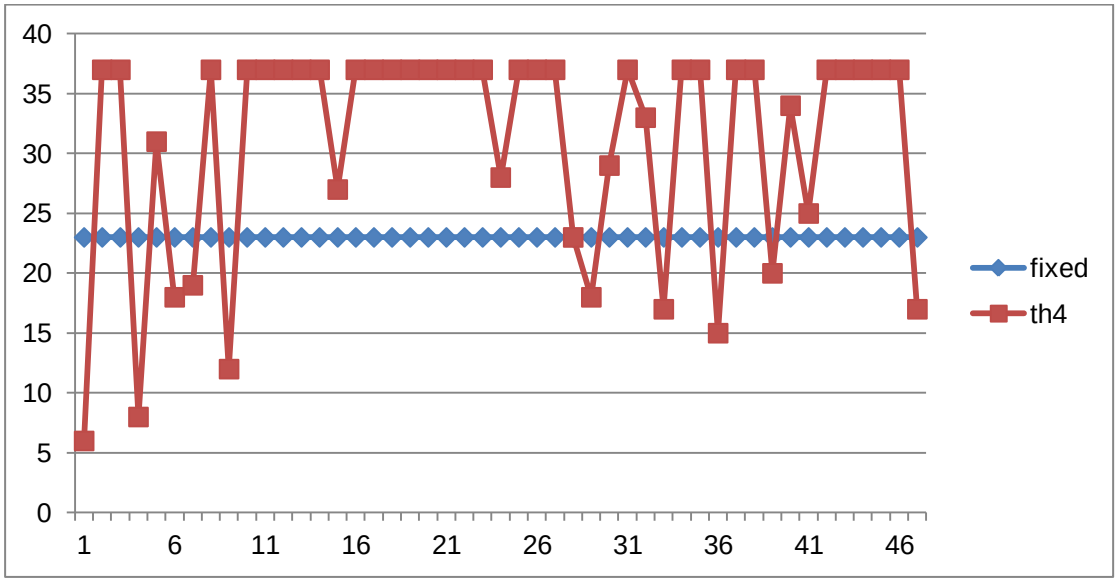
Στη συνέχεια γίνεται ξανά αντίστοιχη ανάλυση ανά φάση, του GreenTime της SchIC σε σχέση με της fixed_time πολιτικής.

-Phase 1

Γράφημα 19: GreenTime της SchIC και του Fixed_time της φάσης 1 με **thc = 2**



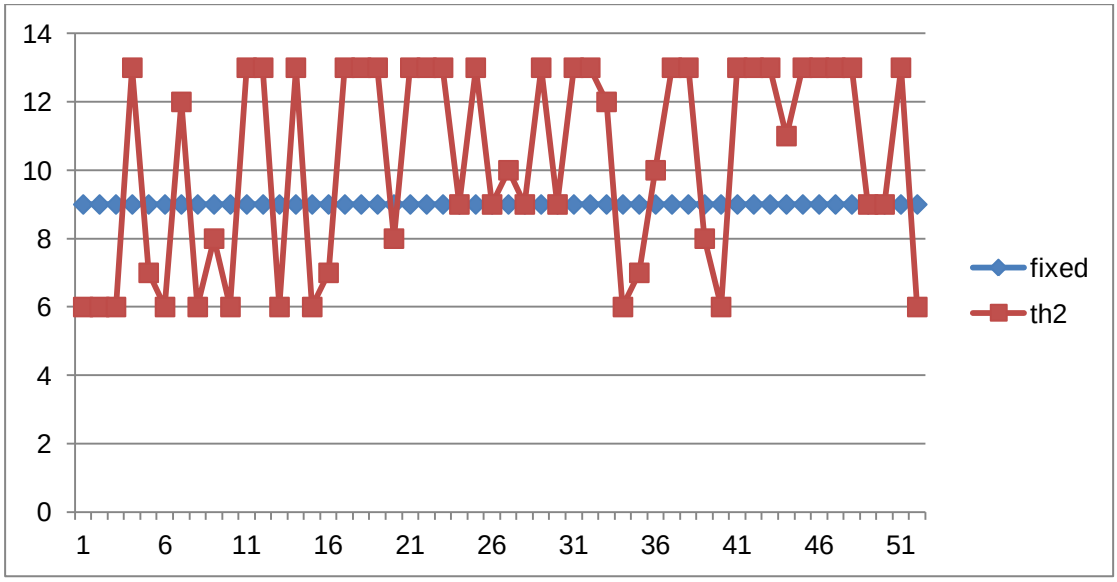
Γράφημα 20: GreenTime της SchIC και του Fixed_time της φάσης 1 με **thc = 4**



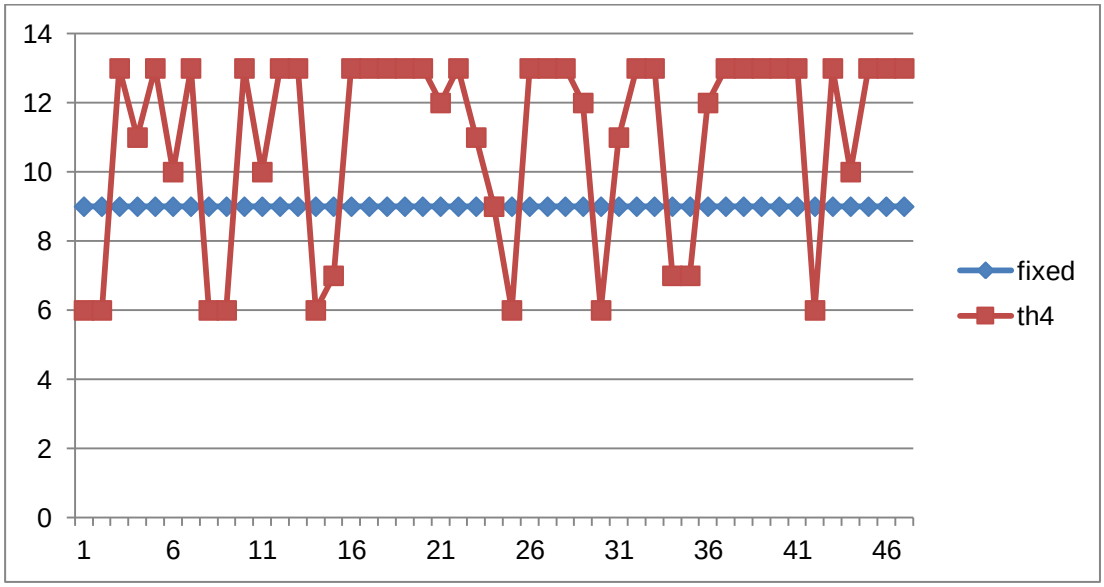
Όπως προαναφέρθηκε η SchIC δίνει στην πρώτη φάση αρκετά συχνά τιμές GreenTime κοντά στο Gmax και έτσι για αυτό παρατηρείται και μεγάλη διαφορά με το fixed_time.

-Phase 2

Γράφημα 21: GreenTime της SchIC και του Fixed_time της φάσης 2 με **thc = 2**

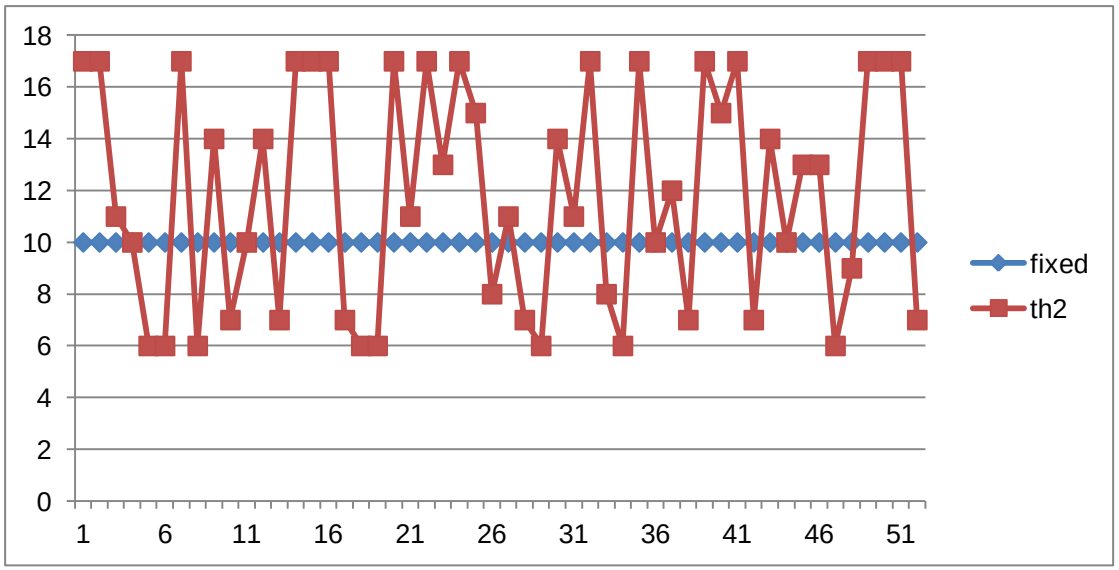


Γράφημα 22: GreenTime της SchIC και του Fixed_time της φάσης 2 με **thc = 4**

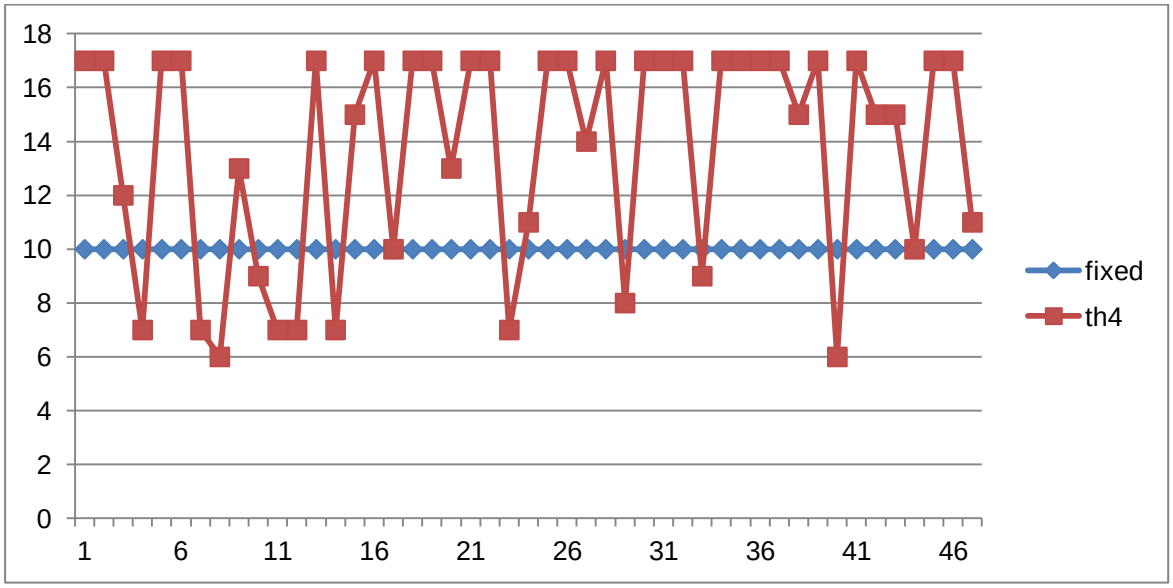


-Phase 3

Γράφημα 23: GreenTime της SchIC και του Fixed_time της φάσης 3 με **thc = 2**

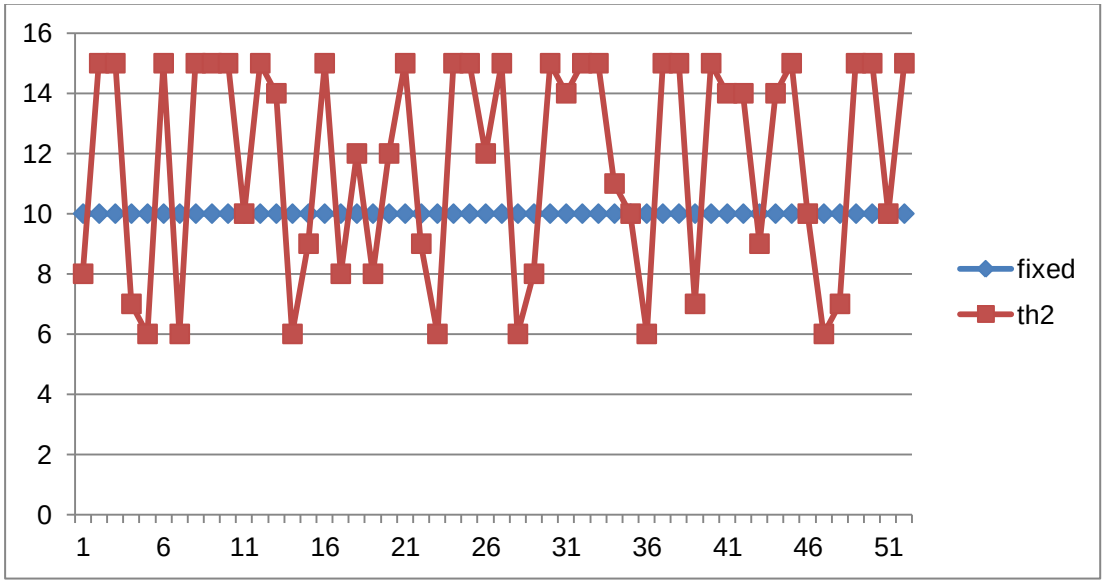


Γράφημα 24: GreenTime της SchIC και του Fixed_time της φάσης 3 με **thc = 4**

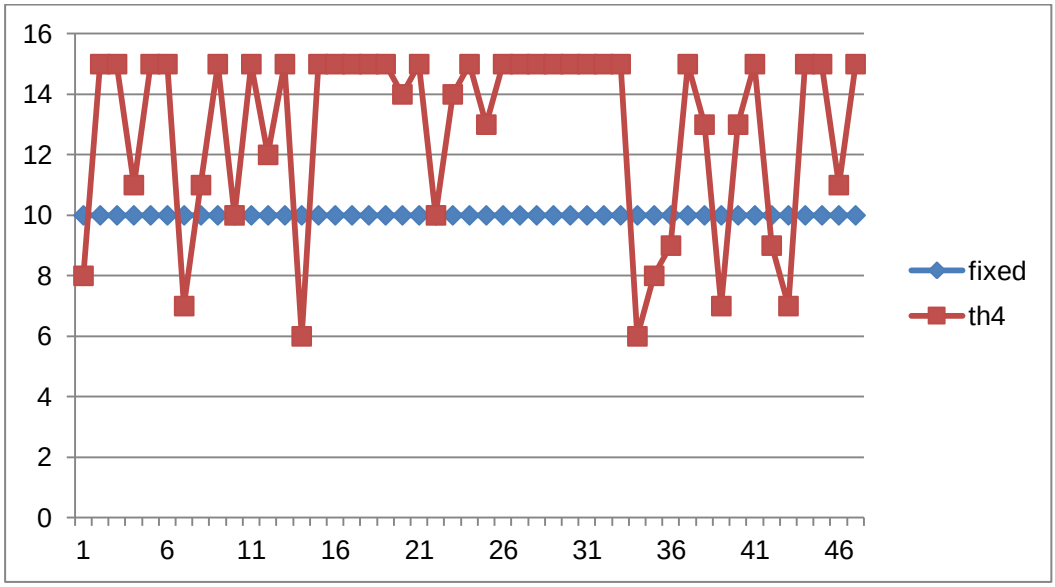


-Phase 4

Γράφημα 25: GreenTime της SchIC και του Fixed_time της φάσης 4 με **thc = 2**

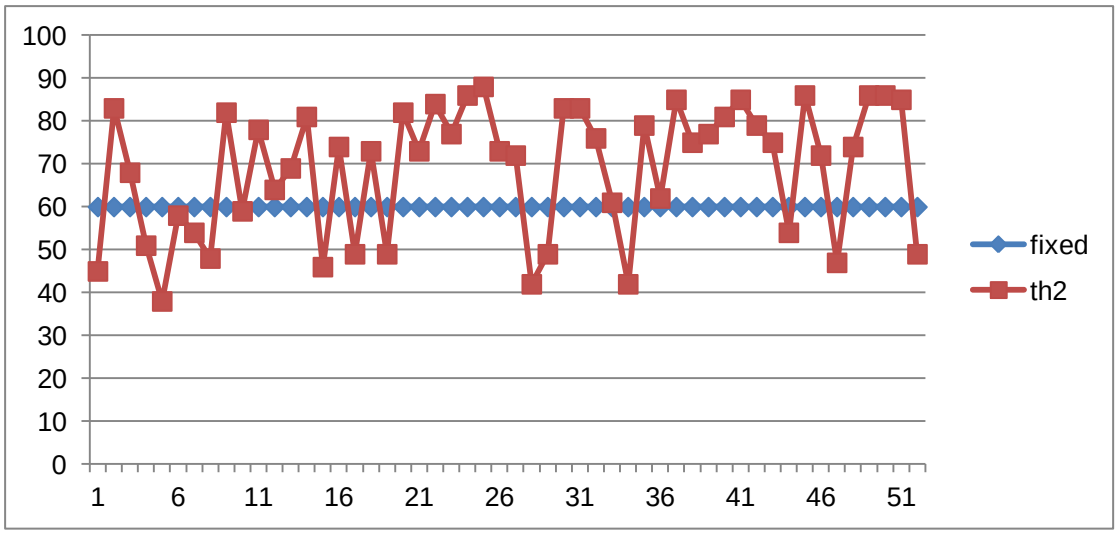


Γράφημα 26: GreenTime της SchIC και του Fixed_time της φάσης 4 με **thc = 4**

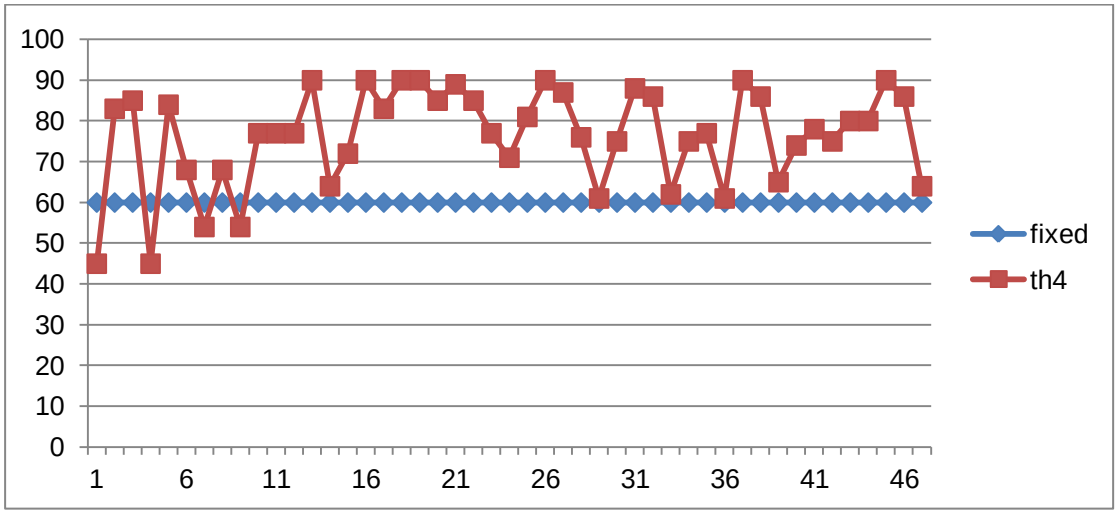


-Cycle

Γράφημα 27: Κύκλος της SchIC και του Fixed_time με **thc = 2**



Γράφημα 28:Κύκλος της SchIC και του Fixed time με **thc = 4**



Στην ανάλυση του cycle του δευτέρου κόμβου φαίνεται καθαρά η υπέρβαση που έγινε στους χρόνους του πρασίνου σε σύγκριση με το fixed_time και για thc = 2 αλλά και ειδικότερα για thc = 4.

8.Αποτελέσματα

Πίνακας 3: Σύγκριση αποτελεσμάτων

Συγκριτικός πίνακας	**Delay T. (sec/km)**	**Density (veh/km)**	**H. Speed (km/h)**	**Stops(#/veh/km)**
suptrac_th2	65,16	9,73	26,09	2,54
suptrac_th4	65,57.	9.739	26,01	2,45
fixed_time	91.35	11,45	21,93	3,18.
Διαφορά	26,19	1,72	4,16	0.63
***%** διαφορά με th2*	28,670%	15,017%	18,980%	19,971%
***%** διαφορά με th4*	28,223%	15,038%	18,629%	22,806%

Όπως εύκολα παρατηρείται η διάφορα της μεθόδου SchIC με της κλασικής μεθόδου σταθερού χρόνου δίνει ένα ξεκάθαρο προβάδισμα. Η πρώτη στήλη μας δείχνει την χρονική καθυστέρηση που θα δεχτεί ένα αμάξι, από την στιγμή που αυτό εισέρθει στο δίκτυο μας μέχρι τη στιγμή που αυτό θα εξέλθει από αυτό. Η δεύτερη στήλη μας δίνει την πυκνότητα σε αυτοκίνητα που έχει η ροή μας, ένα μέγεθος ποιότητας των μεθόδων, καθώς βλέπουμε ότι η fixed time είναι κατά 15% περίπου ποιο πυκνή και άρα και ότι εξάγει τα αμάξια που εισέρχονται στο δίκτυο της 15% περίπου πιο αργά. Η τρίτη στήλη είναι ο δείκτης της ταχύτητας στο δίκτυο. Έτσι έχουμε ότι η χρήση της SchIC επιφέρει περίπου 19% μεγαλύτερη ταχύτητα κίνησης. Η τελευταία στήλη

αφορά των αριθμό των στάσεων ενός οχήματος ανά χιλιόμετρο. Άλλο ένα ποιοτικό στοιχείο της ροής, το οποίο δείχνει ίσως πιο ‘’γραφικά’’ την καθυστέρηση των οδηγών εντός του συστήματος και αν θέλουμε να επεκταθούμε και το ψυχολογικό stress που προσδίδεται στους οδηγούς όταν είναι αναγκασμένοι να σταματούν συχνότερα.

Πιο σημαντικό μέγεθος στη σύγκριση των 2 μεθόδων είναι το Delay καθώς δείχνει την συνολική καθυστέρηση που προκαλεί το σύστημα στα οχήματα του, είτε αυτό είναι κάποιες επιπλέον στάσεις, είτε γενικά χαμηλή ταχύτητα της ροής εξαιτίας άλλων αιτιών. Έτσι μια διαφορά στο κριτήριο Delay της τάξης του 28% είναι πραγματικά πολύ σημαντική και δίνει μια συνολική εικόνα του συστήματος και της αποτελεσματικότητας της λειτουργίας του.

9.Συμπεράσματα

Από ότι μελετήθηκε παραπάνω συμπεραίνεται με ασφάλεια ότι η μέθοδος SchIC παράγει καλύτερα αποτελέσματα σε σχέση με την καθιερωμένη μέθοδο. Παρόλη την αύξηση του thc και την ενδεχόμενη πτώση της ευαισθησίας του συστήματος, η διαφορά που παρατηρήθηκε κρίνεται μάλλον αμελητέα. Ειδικότερα στον πρώτο κόμβο δε φάνηκαν σχεδόν καθόλου διαφορές μεταξύ thc = 2 ή thc = 4 και στον δεύτερο κόμβο που παρατηρήθηκαν διαφορές, αυτές δεν ήταν μεγάλες. Όπως δείχνουν τα τελικά αποτελέσματα χάνοντας ευαισθησία με thc = 4 είχαμε μια μείωση στο Delay και στην ταχύτητα του συστήματος, αλλά είναι μια ‘’θυσία’’ της τάξης του 0,3% με 0,4 % η οποία ήταν ικανή για να επιτρέψει στον αλγόριθμο να παράγει λύσεις με ευκολία για αρκετά μεγάλη ζήτηση. Επιπλέον παρατηρείται μια διαφορά της πυκνότητας των οχημάτων αλλά αρκετά μικρής κλίμακας 0,02% υπέρ του thc =2, αλλά και μια διαφορά ως προς τις συνολικές στάσεις, περίπου 3% υπέρ του thc = 4 που είναι αναμενόμενο εφόσον μειώνονται τα clusters στο σύστημα.

10.Μελλοντικές διερευνήσεις

Ένας τομέας που θα μπορούσε να διερευνηθεί στο μέλλον θα μπορούσε να είναι η προσπάθεια επικοινωνίας των κόμβων ενός δικτύου μεταξύ τους έτσι ώστε να υπάρχει γνώση της ροής στον επόμενο κύκλο, αλλά και της αποθηκευτικής ικανότητας του κατάντη κόμβου. Επίσης ναι μεν παρατηρήθηκε ότι η μέθοδος SchIC δημιουργεί έμμεσα καλό offset μεταξύ των 2 γειτονικών κόμβων, αλλά θα πρέπει να γίνει περαιτέρω διερεύνηση με πιο πολύπλοκα δίκτυα.

Παράρτημα

Δημιουργία πίνακα C

```
for (i=0;i<I;i++)
{
    j=0;
    if(a[i][0] >= Q)
    {
        q[i]=a[i][0];

        if(q[i]>qmax[i])
        {
            qmax[i]=q[i];
        }

        a[i][0]=0;
    }

    if(q[i]>0)
    {


        C[j][0][i]=q[i];
        C[j][1][i]=0;
        C[j][2][i]=(float)q[i]/(float)sfr;
        C[j][3][i]=(float)q[i]/(float)sfr;
        C[j][4][i]=sfr;



        j=j+1;
    }

    for (h=0;h<Hp/samp;h++)
    {
        if(a[i][h] > 0)
        {
            C[j][0][i]=a[i][h];
            C[j][1][i]=h*samp;
            C[j][2][i]=samp+h*samp;
            C[j][3][i]=samp;
            C[j][4][i]=(float)a[i][h]/(float)samp;

              if(h>0)
              {
                  if(j>0)
                  {
                      if((C[j][1][i]-C[j-1][1][i])<=thc)
                      {
                          col=C[j-1][0][i]+C[j][0][i];

                          arr=min1(C[j][1][i],C[j-1][1][i]);
                          dep=max1(C[j][2][i],C[j-1][2][i]);


                          C[j-1][0][i]=col;
                          C[j-1][1][i]=arr;
                          C[j-1][2][i]=dep;
                          C[j-1][3][i]=dep-arr;
                          C[j-1][4][i] = (float)col/(float)(dep-arr);



                          C[j][0][i]=0;
                          C[j][1][i]=0;
                          C[j][2][i]=0;
                          C[j][3][i]=0;
                          C[j][4][i]=0;

                          j=j-1;
                      }
```

```
if(q[i]>0)
{
    if(j==1)
    {
        if(C[j][1][i] <= C[j-1][2][i])
        {

            if((C[j][2][i] <= C[j-1][2][i])||(C[j][4][i] >= C[j-1][4][i]))
            {

                C[j-1][0][i]=C[j-1][0][i]+C[j][0][i];
                C[j-1][3][i]=((float)C[j-1][0][i])/(float)C[j-1][4][i];
                C[j-1][2][i]= C[j-1][3][i]+C[j-1][1][i];

                C[j][0][i]=0;
                C[j][1][i]=0;
                C[j][2][i]=0;
                C[j][3][i]=0;
                C[j][4][i]=0;

                j=j-1;
            }
            else
            {
                ddur=(float)(C[j-1][2][i]-C[j][1][i])/(float)((float)(1-C[j][4][i])/(float)sfr);

                if(ddur >= C[j][3][i])
                {
                    C[j-1][0][i]=(C[j-1][0][i]+C[j][0][i]);
                    C[j-1][3][i]=((float)C[j-1][0][i])/(float)C[j-1][4][i];
                    C[j-1][2][i]= C[j-1][3][i]+C[j-1][1][i];

                    C[j][0][i]=0;
                    C[j][1][i]=0;
                    C[j][2][i]=0;
                    C[j][3][i]=0;
                    C[j][4][i]=0;

                    j=j-1;
                }


            else
            {
                ddur=(float)(C[j-1][2][i]-C[j][1][i])/(float)((float)(1-C[j][4][i])/(float)sfr);

                if(ddur >= C[j][3][i])
                {
                    C[j-1][0][i]=(C[j-1][0][i]+C[j][0][i]);
                    C[j-1][3][i]=((float)C[j-1][0][i])/(float)C[j-1][4][i];
                    C[j-1][2][i]= C[j-1][3][i]+C[j-1][1][i];

                    C[j][0][i]=0;
                    C[j][1][i]=0;
                    C[j][2][i]=0;
                    C[j][3][i]=0;
                    C[j][4][i]=0;

                    j=j-1;
                }

                else
                {

                        C[j-1][0][i]=(C[j-1][0][i]+(C[j][0][i]*(ddur/C[j][3][i])));
                        C[j-1][3][i]=((float)C[j-1][0][i]/(float)sfr);
                        C[j-1][2][i]= C[j-1][3][i];

                        C[j][0][i]=(C[j][0][i]-(C[j][0][i]*(ddur/ C[j][3][i])));
                        C[j][1][i]=(C[j][1][i]+ddur);


                        if( C[j][0][i] < 0)
                        {
                            if( (C[j][0][i]-(C[j][0][i]*(ddur/ C[j][3][i]))) >= (float)(0.55) )
                            {
                                C[j][0][i]=1;
                                C[j-1][0][i]--;
                            }
                        }

                        C[j][0][i]=(int)(C[j][0][i]);
                        C[j-1][0][i]=(int)(C[j-1][0][i]);
```

```
                                        }
                                    }
                                }
                            }
                            }
                        }

                    }
                    j=j+1;
                }

            }
            j=0;
        }
```

Εύρεση Βέλτιστης Λύσης

```
Xempty=(int*)malloc(I*sizeof(int));

for(i=0;i<I;i++)
{

    Xempty[i]=0;
    cmax=0;
    for(j=0;j<Hp/samp;j++)
    {

        cmax=C[j][0][i] + cmax;
    }

}



for(n=0;n<w;n++)
{d=0;


    Sy[0][0][n]=ic;
    Sy[0][1][n]=0;
    Sy[0][2][n]=0;

  for(k=1;k<noz+1;k++)
{

    i=0;
    j=0;

    i=S[n][k-1];

    j=Sx[n][k-1];


    pst=Sy[k-1][1][n]+ MinSwitch[(int)Sy[k-1][0][n]][i];

    ast=maxfl((float)C[j][1][i], pst);
```

```
    if((pst>(float)C[j][1][i])&&(Sy[k-1][0][n] != (float)i))
    {

        ast=ast+(float)sult;

    }

    tt=ast+(float)C[j][3][i];
    dd=((float)C[j][0][i])*(maxfl(ast-C[j][1][i],0));
    d=d+dd;

    for(v=0;v<3;v++)
    {

        Sy[k][v][n]=Xempty[v];
    }

    Sy[k][0][n]=S[n][k-1];
    Sy[k][1][n]=tt;
    Sy[k][2][n]=d;

    }
    }

    sn=0;

    for(n=1;n<w;n++)
    {


        if (Sy[noz][1][n] < Sy[noz][1][sn])
        {

            sn=n;

        }

       else
       {
           if(Sy[noz][1][n] == Sy[noz][1][sn])
           {

               if(Sy[noz][2][n] < Sy[noz][2][sn])
               {

               sn=n;

               }

           }
       }

}
```

Δημιουργία πίνακα MinSwitch

```
MinSwitch[0][0]=0;
MinSwitch[0][1]=Y;

if(I>2)
{
    for(j=2;j<I;j++)
    {
        MinSwitch[0][j]=MinSwitch[0][j-1]+Y+Gmin;
    }
}

for(i=1;i<I;i++)
{
    for(j=0;j<I;j++)
    {
        if(j-i==0)
        {
            MinSwitch[i][j]=0;
        }
        else
        {
            if(j-i > 0)
            {

                MinSwitch[i][j]=MinSwitch[0][j-i];
            }
            else
            {
                MinSwitch[i][j]=MinSwitch[0][I+(j-i)];
            }
        }
    }
}
```

Δημιουργία Προσομοιωτή

```
for( i = 0 ; i < 3600 ; i++ )
    {
        if(phaseTime>80)
        {
            break;
        }


        if(fmod(phaseTime,samp)==0)
        {
            for(j=0;j<I;j++)
            {
                printf("\n MaxTime[%d]=%d \n",j,MaxTime[j]);
            }
            if(SwiftTime>0)
            {
            printf("\n MTime=%d \n", MTime/SwiftTime );
            }

            printf("\n SwiftTime=%d \n",SwiftTime);

            fscanf(fp4,"%d ",&n);

            nm=nm+n;

            Ra= rand() % I;

            in[Ra]=n;
            act++;
            printf(" epanalipsi %d \n", act);
            printf("\n phaseTime = %d & phase = %d \n \n",phaseTime,phase );


                ext=jimmy(in,phase);
```

```
            for( j = 0 ; j < I ; j++ )
                {
                    printf(" %d ", in[j]);
                    in[j]=0;
                }
            printf("\n \n");


            if(phaseTime>=greenTime[phase])
            {
                decision=greenTime[phase]+ext;
                printf("ext= %d decision = %d \n \n",ext, decision);


                if(decision < Gmin)
                {
                    decision=Gmin;
                    printf("\n \n decision < Gmin \n\n");
                }
                if(decision > Gmax[phase])
                {
                    decision=Gmax[phase];
                    printf("\n \n decision > Gmax \n \n");
                }
                if(ext!=0)
                {
                    if((phaseTime+samp) <= Gmax[phase])
                    {
                        if( (phaseTime >= Gmin) && (phaseTime <= decision) )
                        {
                            greenTime[phase]=decision;
                            printf("\n \n extention \n \n");

                        }
                    }

                     else
                     {
                         SwiftTime++;
                         MTime =  phaseTime + MTime  ;


                         if(phaseTime>MaxTime[phase])
                         {
                             MaxTime[phase]=phaseTime;
                         }

                         greenTime[phase]=Gmin;
                         phase=(phase+1) %I;
                         phaseTime=-1;
                         printf("\n \n ext==0 \n \n");
                     }
                 }
                 else
                 {
                     if(phaseTime>=Gmin)
                     {
                         SwiftTime++;
                         MTime= MTime + phaseTime ;


                         if(phaseTime>MaxTime[phase])
                         {
                             MaxTime[phase]=phaseTime;
                         }

                         printf("\n \n phaseTime>=Gmin  \n \n");
                         greenTime[phase]=Gmin;
                         phase=(phase+1) %I;
                         phaseTime=-1;

                     }
                 }
             }
        }
        phaseTime++;
}
```

Βιβλιογραφία

- [1] Xie, X.-F., S. F. Smith, L. Lu, and G. J. Barlow. Schedule-driven intersection control. Transportation Research Part C: Emerging Technologies, Vol. 24, 2012, pp. 168–189.

- [2] Papageorgiou, M., C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. Proceedings of the IEEE, Vol. 91, No. 12, 2003, pp. 2043–2067.

- [3] Robertson, D.I., Bretherton, R.D., 1974. Optimum control of an intersection for any known sequence of vehicle arrivals. In: IFAC/IFIP/IFORS Symposium on Traffic Control and Transportation Systems. Monte Carlo, Monaco, pp. 3–17.

- [4] Porche, I., Lafortune, S., 1999. Adaptive look-ahead optimization of traffic signals. ITS Journal 4 (3-4), 209–254.

- [5] Steven G. Shelby, 2004. Single-Intersection Evaluation of Real-Time Adaptive Traffic Signal Control Alogorithms. Transportation Research Record: Journal of the Transportation Research Board, No. 1867, TRB, National Research Council, Washington, D.C., pp.183-192.

- [6] Survrajeet Sen and K. Larry Head, 1997. Controlled Optimization of Phases at an Intersection. Transportation Science, Vol. 31, No. 1.

- [7] Nathan H. Gartner ,1983. Development of Demand-Responsive Strategies for Urban Traffic Control. Lecture Notes in Control and Information Sciences, System Modeling and Optimization, Proceedings of the 11th IFIP Conference Copenhagen, Denmark, July 25-29.