Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική σχολή

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων

Διπλωματική εργασία

"Συγκριτική μελέτη αλγορίθμων αραιοποίησης μεγάλης κλίμακας σύνθετων δικτύων"

"A comparative study of algorithms sparsifying large scale complex networks"

Υψηλάντης Α. Κωνσταντίνος

Επιβλέποντες Καθηγητές : Κατσαρός Δημήτριος
Λέκτορας Π.Θ

Μποζάνης Παναγιώτης
Αναπληρωτής Καθηγητής Π.Θ

Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

Στην οικογένεια μου

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας κ. Κατσαρό Δημήτριο και κ. Μποζάνη Παναγιώτη για την εμπιστοσύνη που μου έδειξαν, την άριστη συνεργασία και την συνεχή καθοδήγηση που μου έδωσαν. Τέλος ένα μεγάλο ευχαριστώ στην οικογένεια μου και τους φίλους μου για την στήριξη τους καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Λίστα Εικόνων

Κεφάλαιο 1

Εισαγωγή

1.1 Μια γενική περιγραφή

Είναι γεγονός πλέον πως στην σημερινή κοινωνία η χρήση των δικτύων είναι απαραίτητη. Η μελέτη των δικτύων, συμπεριλαμβανομένων των δικτύων υπολογιστών, τα κοινωνικά δίκτυα, και τα βιολογικά δίκτυα, έχει προσελκύσει τεράστιο ενδιαφέρον στην κοινωνία των υπολογιστών.[1]

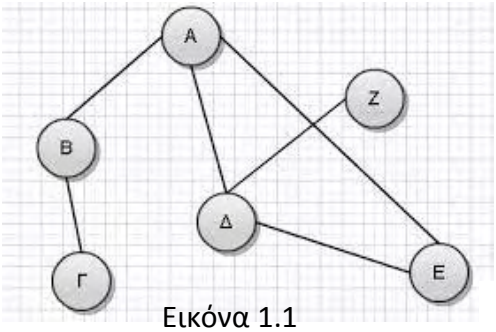
Τα σημερινά πραγματικού κόσμου δίκτυα τείνουν ολοένα και περισσότερο να είναι πυκνά, και μάλιστα τις περισσότερες φορές σε τέτοιο βαθμό που κάνει την μελέτη τους αρκετά δύσκολη και φυσικά πολύ πιο αργή. Όμως χρειαζόμαστε πραγματικά όλους αυτούς τους κόμβους σε ένα δίκτυο ; Μήπως τελικά είναι πολύ πιο εύκολο να μελετήσουμε ένα δίκτυο ή στην δική μας περίπτωση έναν γράφο, αν ξεχωρίζαμε τους κόμβους και κρατούσαμε αυτούς που πραγματικά χρειαζόμαστε;
Η απάντηση είναι προφανής. Φυσικά και είναι, και όχι απλώς πιο εύκολο αλλά και πιο γρήγορο, με λιγότερες απαιτήσεις σε μνήμη, λιγότερες απαιτήσεις σε υπολογιστική ισχύ, και φυσικά πολύ πιο εύκολο στην κατανόηση και εξακρίβωση των αποτελεσμάτων που παρήχθηκαν.

Όμως με πιο τρόπο θα μπορέσουμε να ξεχωρίσουμε ποιοι είναι οι "χρήσιμοι " κόμβοι ; Πιο είναι αυτό το κριτήριο;

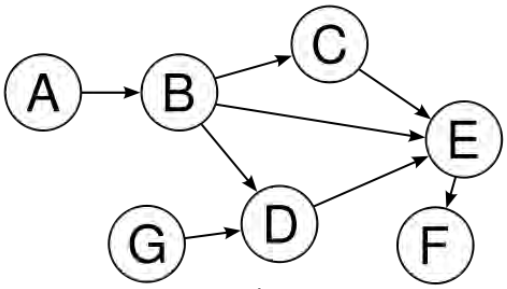
Πριν προχωρήσουμε στην απάντηση των παραπάνω ερωτημάτων θα κάνουμε μια αναφορά στον ορισμό του γράφου και θα δούμε δυο από τα πιο γνωστά αλλά και σημαντικά είδη γράφων, τους μη-κατευθηνόμενους και τους κατευθυνόμενους γράφους.

Με τον όρο γράφο εννοούμε μια αφηρημένη αναπαράσταση ενός συνόλου στοιχείων, όπου μερικά ή και όλα τα ζευγάρια στοιχείων συνδέονται μεταξύ τους με δεσμούς. Τα διασυνδεδεμένα στοιχεία ονομάζονται κορυφές ενώ οι δεσμοί που συνδέουν τα ζευγάρια των κορυφών ονομάζονται ακμές. Συνήθως, ένας γράφος απεικονίζεται σε διαγραμματική μορφή ως ένα σύνολο κουκκίδων για τις κορυφές ενωμένες μεταξύ τους με γραμμές για τις ακμές.
Οι ακμές μπορούν να είναι κατευθυνόμενες ή μη-κατευθυνόμενες . Για παράδειγμα, εάν οι κορυφές αντιπροσωπεύουν τα άτομα σε ένα πάρτι και υπάρχει ακμή μεταξύ δύο ανθρώπων, αν δίνουν τα χέρια, τότε αυτό είναι ένα μη-κατευθυνόμενο γράφημα γιατί αν το πρόσωπο Α έδωσε τα χέρια με το πρόσωπο Β, τότε και το πρόσωπο Β έδωσε τα χέρια και με πρόσωπο Α. Από την άλλη πλευρά, εάν οι κορυφές αντιπροσωπεύουν τα άτομα σε ένα πάρτι και υπάρχει μια ακμή από το πρόσωπο Α στο πρόσωπο Β, όταν το πρόσωπο Α γνωρίζει το πρόσωπο Β, τότε η γραφική αυτή παράσταση είναι κατευθυνόμενη γιατί το να γνωρίζεις κάποιον δεν

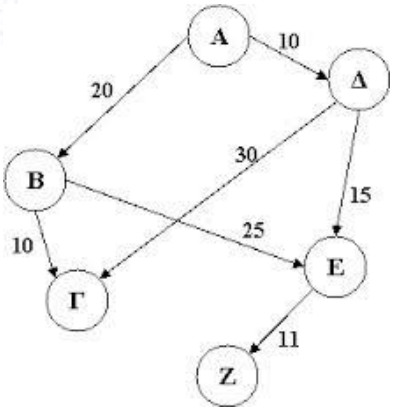
είναι απαραίτητα μια συμμετρική σχέση (δηλαδή, ένα άτομο που γνωρίζει ένα άλλο πρόσωπο δεν συνεπάγεται κατ 'ανάγκη το αντίθετο) .Το τελευταίο αυτό είδος γραφήματος ονομάζεται κατευθυνόμενος γράφος και οι ακμές του ονομάζονται κατευθυνόμενες ακμές ή τόξα.[2] Πολλές φορές είναι δυνατό οι ακμές να έχουν κάποιον αριθμό πάνω τους . Τότε λέμε ότι πρόκειται για έναν βεβαρημένο γράφο ή γράφο με βάρη.



Εικόνα 1.1



Εικόνα 1.2



Εικόνα 1.3

Την απάντηση στην πιο πάνω ερώτηση , πως δηλαδή θα επιλέξουμε τους " καλούς " κόμβους έρχονται να μας την δώσουν δυο αλγόριθμοι, όπου και θα μελετήσουμε σε αυτήν την εργασία, ο LANS και ο SPINE , ο καθένας με τις δίκες του παραδοχές για τον γράφο.

1.2 Σκοπός Της Εργασίας

Σκοπός αυτής της εργασίας είναι η μελέτη και η κατανόηση των δυο αλγόριθμων. Γίνετε αναλυτική περιγραφή κάθε ενός , ποιες παραδοχές κάνει ο κάθε ένας, σε ποια είδη γράφων μπορούν να χρησιμοποιηθούν. Τέλος γίνετε σύγκριση χρόνων εκτέλεσης για διάφορα γραφήματα και μελέτη των αποτελεσμάτων τους. Σαφώς κάθε αλγόριθμος έχει υλοποιηθεί με τα ίδια κριτήρια. Όλες οι υλοποιήσεις είναι σε γλώσσα C.

Κεφάλαιο 2

Αλγόριθμος LANS (Locally Adaptive Network Sparsification)

2.1 Σύντομη Περιγραφή Του Προβλήματος

Η ανάλυση του δικτύου έχει γίνει ένα σημαντικό επιστημονικό εργαλείο για τη μελέτη των πυκνών, πολύπλοκων δόμων μεταξύ κάποιων σχέσεων . Μερικές φορές η δομή του δικτύου είναι εγγενής στις σχέσεις (π.χ., το WWW[3] , αεροπορικά δίκτυα[4], ή πραγματικά νευρωνικά δίκτυα[5]). Σε άλλες περιπτώσεις, είναι ένα χρήσιμος τρόπος για την κωδικοποίηση των σχέσεων αλληλεπίδρασης που μπορεί να υπάρχουν μεταξύ των οργανισμών , των κοινωνικών ομάδων ή τεχνολογικών δομών, όπως το Διαδίκτυο.

Σε πολλά σημαντικά παραδείγματα η δομή του δικτύου επιβάλλεται ως τυπολογική για τη διευκόλυνση της οπτικοποίησης και της μετέπειτα ανάλυσης ενός συνόλου δεδομένων, τα οποία δεν έρχονται εκ των προτέρων σε μορφή δικτύου όμως παρατηρούνται να έχουν κάποιες ομοιότητες (ή ανομοιότητες) μεταξύ τους. Σε αυτές τις περιπτώσεις, ένα βεβαρημένο δίκτυο μπορεί να χρησιμοποιηθεί για να συνοψίσει και να οπτικοποιήσει αυτά τα δεδομένα, καθώς και να διευκολύνει στην μετέπειτα μελέτη τους. [6]

Φυσικά ένα τέτοιο δίκτυο πολλές φορές μπορεί να είναι υπερβολικά πυκνό , που να καθιστά δύσκολη την ανάλυση του. Γι αυτό το λόγο είναι χρήσιμο σε τέτοιες περιπτώσεις να βρίσκουμε τη " ραχοκοκαλιά " (backbone) του δικτύου.

2.2 Μέθοδοι Εύρεσης Ραχοκοκαλιάς Δικτύου

Έστω κατευθυνόμενο γράφημα με μη αρνητικά βάρη ακμών. Μια κλασσική μέθοδος αραιοποίησης ενός τέτοιου γραφήματος είναι να κρατήσουμε όλες τις ακμές που έχουν βάρος μεγαλύτερο από κάποια τιμή κατωφλιού T . Πρόκειται για μια γρήγορη και εύκολη στην υλοποίηση μέθοδος που όμως αποτυγχάνει να διατηρήσει την σημαντική πληροφορία όταν η χρήσιμη γεωμετρία του δικτύου κατανέμετε σε πολλαπλά βάρη ακμών.

Μια άλλη μέθοδος είναι η εύρεση του γεννητικού δέντρου (Spanning Tree) του δικτύου.[6] Ένα γεννητικό δέντρο είναι ένα δέντρο (άκυκλος γράφος) στο οποίο περιέχονται όλοι οι κόμβοι και ορισμένες ή όλες οι ακμές του αρχικού γραφήματος. Το συνολικό βάρος του αποτελείται από το άθροισμα των βαρών των ακμών που το αποτελούν. [7]

Ακόμη όμως και με την δεύτερη μέθοδο πάλι δεν είμαστε 100% καλυμμένοι αφού θέλουμε να διατηρούμε τις κυκλικές γεωμετρίες που μπορεί να περιέχονται στο δίκτυο μας και καμία από τις παραπάνω μεθόδους δεν μας το παρέχει αυτό.

Μια τρίτη μέθοδος είναι αυτή του ανομοιόμορφου φίλτρου (disparity filter). Έστω

$$pij = \frac{wij}{\sum_{k=1}^{Ni} wik}$$

το κλασματικό βάρος ακμής από τον κόμβο i στον j και Ni ο αριθμός των γειτόνων του κόμβου i.
Αυτή η κατανομή ακμών στο i συγκρίνετε με ένα "τυχαίο" μοντέλο (null model) στο οποίο Ni-1 σημεία δημιουργούν μια τυχαία κατανομή Wi βαρών που αθροίζουν στο 1. Μια ακμή θεωρείτε σημαντική αν η πιθανότητα p μιας ακμής μεγαλύτερης του pij του τυχαίου μοντέλου, είναι μικρότερη μιας τιμής α.

Τέλος μια παρόμοια μέθοδος (bistochastic filter) μέσα από μια επαναληπτική διαδικασία μετασχηματίζει τα αρχικά βάρη ,δημιουργώντας έναν διστοχαστικό πίνακα όπου όλες οι στήλες και οι γραμμές του αθροίζουν στο 1. Όταν όλα τα βάρη μετασχηματιστούν , προστίθενται ακμές στο γράφημα με φθίνουσα σειρά βάρους έως ότου κάποιο κριτήριο τερματισμού εκπληρωθεί.[8]

2.3 Μέθοδος LANS

Ο LANS προσπαθεί να εκμεταλλευτεί τα πλεονεκτήματα των μεθόδων που αναφέραμε στο 2.2 ώστε να δημιουργήσει μια μέθοδο που να κρατάει τις σημαντικές ,βάση κάποιου κριτηρίου, ακμές χωρίς όμως να αγνοεί την γεωμετρία του δικτύου. Πιο αναλυτικά :
Έστω μια ακμή θετικού βάρους wij και pij το κλασματικό βάρος που περιγράψαμε στην προηγούμενη μέθοδο. Για κάθε ακμή θεωρούμε ως βαθμό ενός κόμβου τον αριθμό των ακμών με θετικό βάρος που προσάπτονται σε αυτόν. Τέλος για κάθε κόμβο i και για κάθε γείτονα του j θεωρούμε τη κλασματική μη-μηδενικη ακμή με βάρος μικρότερο ή ίσο με pij, και έχουμε

$$F(pij) = \frac{1}{Ni} \sum_{k=1}^{Ni} 1\{pik \leq pij\}$$

όπου 1{} είναι η χαρακτηριστική συνάρτηση , Ni είναι ο αριθμός των γειτόνων του κόμβου i. Αυτός ο υπολογισμός είναι ισοδύναμος με τον υπολογισμό του εμπειρικού cdf για τα κλασματικά βάρη ακμής για κάθε κόμβο.
Αν η τιμή 1- F(pij) είναι μεγαλύτερη από μια προκαθορισμένη τιμή α, τότε θεωρούμε πως η ακμή είναι σημαντική και τη προσθέτουμε στο backbone του δικτύου.[9]

Στην εικόνα που ακολουθεί δίνετε μια τυπική περιγραφή του αλγορίθμου σε μορφή ψευδοκώδικα

1: **function** LANS(S)
2: $P_{ij} = S_{ij} / \sum_k S_{ik}, \forall\, i, j$
3: $A_{ij} = 0, \forall\, i, j$
4: **for** $i = 1,\ldots,$num_nodes **do**
5: Compute empirical cdf of nonzero entries in i'th row of P
6: p_α = 1-α quantile of empirical cdf
7: **for** $j = 1,\ldots,$num_nodes **do**
8: if $P_{ij} > p_\alpha$, set $A_{ij} = Sij$ # Also set $A_{ji} = S_{ij}$ if symmetric backbone desired
9: **end for**
10: **end for**
11: **return** A
12: **end function**

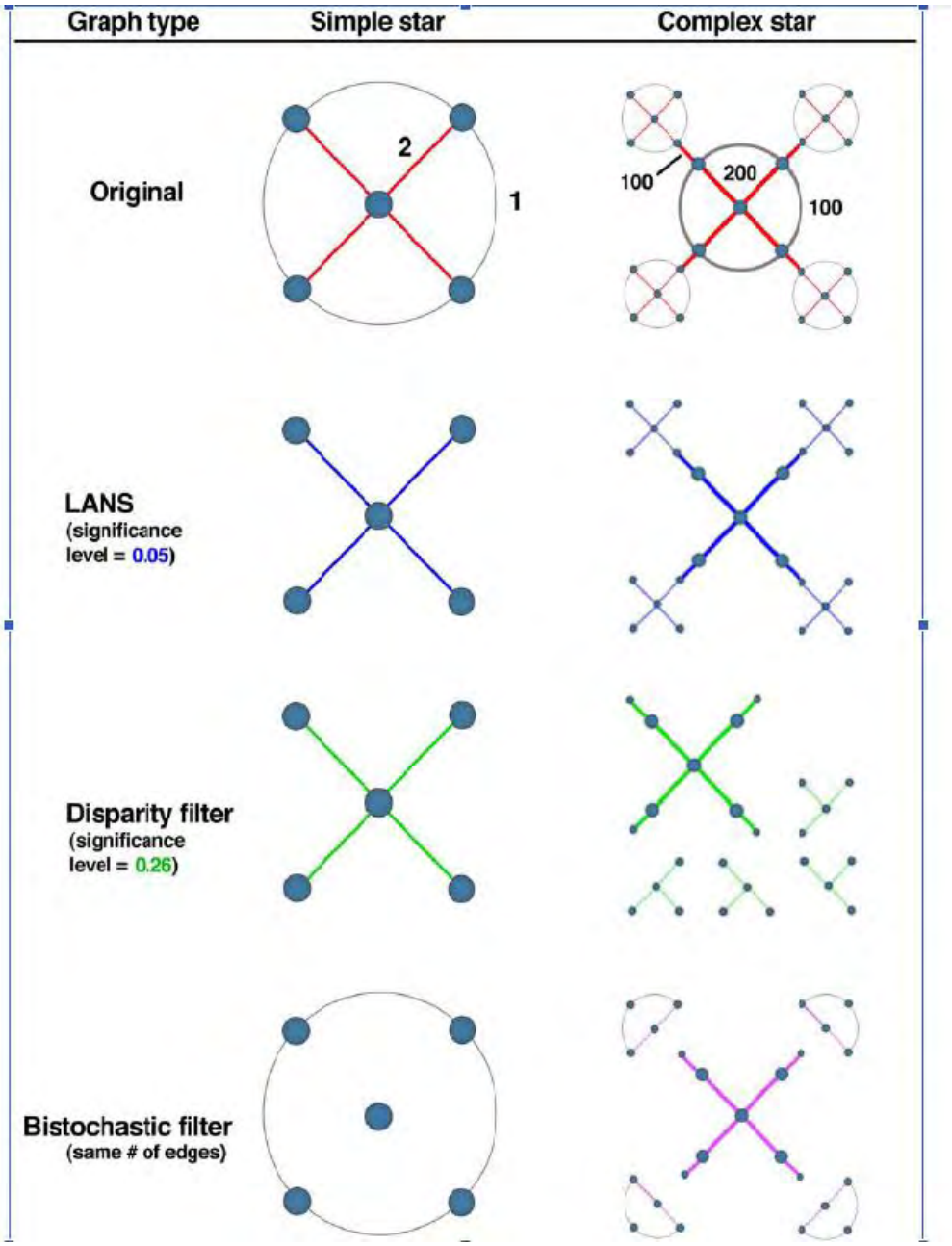
Εικόνα 2

2.4 Πολυπλοκότητα Αλγορίθμου

Λόγω του υπολογισμού του cdf ο αλγόριθμος χρειάζεται O($n^2 log$N) χρόνο, όπου n ο αριθμός των κόμβων του δικτύου και N ο μεγαλύτερος βαθμός κόμβου.
Σε δίκτυα με μη μηδενικά βάρη μεταξύ δυο άκμων έχουμε N=n-1, όμως σε αραιά γραφήματα πραγματικών προβλημάτων το N είναι αρκετά μικρότερο του n. Ο χρόνος για την κατασκευή του διστοχαστικού πίνακα (γραμμή 6-8 εικόνα 2), θέλει O(n^2). Έτσι καταλήγουμε πως ο χρόνος του LANS είναι της τάξης του O(logN).[9]

2.5 Ο Αλγόριθμος Εν Δράση

Ας συγκρίνουμε τώρα σε δυο, σχετικά μικρά γραφήματα, τις μεθόδους του 2.2 με τον LANS.



Εικόνα 3

Παρατηρούμε πως για το πρώτο γράφημα (simple star) και ο LANS και η μέθοδος disparity filter για τιμές α= 0.05 και 0.26 αντίστοιχα, κατάφεραν να βρουν την ιδία γεωμετρία ενώ η τρίτη μέθοδος (distochastic filter) απέτυχε.
Όσο αναφορά το δεύτερο γράφημα (Complex Star) είναι φανερό πως μόνο ο LANS κατάφερε να βρει την σωστή γεωμετρία του δικτύου.[10]

Κεφάλαιο 3

Αλγόριθμος SPINE(Sparcification Of Influence Networks)

3.1 Σύντομη Περιγραφή Του Προβλήματος

Σε πολλά σενάρια ανάλυσης ενός δικτύου, η αραιοποίηση είναι κύριας σημασίας προκειμένου να μπορέσουμε να προχωρήσουμε στην ανάλυση και την μελέτη του. Ακόμη βοήθα στη μείωση του θορύβου των δεδομένων μας , επιτυγχάνοντας πιο ακριβοί αποτελέσματα.[11] Σκοπός σε αυτόν τον αλγόριθμο είναι να εξαλείψουμε τις περιττές ακμές και να κρατήσουμε τις κ πιο σημαντικές ακμές του δικτύου για την καλύτερη διάδοση μιας πληροφορίας μέσα στο δίκτυο μας.

3.2 Η Θεωρία Πίσω Από Τον Αλγόριθμο

Ας θεωρήσουμε ένα κατευθυνόμενο δίκτυο G=(V,D) όπου V ο αριθμός των κόμβων του και D οι ακμές του. Για δυο κόμβους u , v υπάρχει μια ακμή arc(u,v) που δηλώνει ότι ο v συνδέεται με τον u ή αλλιώς ο v ακολουθεί τον u.

Ας θεωρήσουμε πως έχουμε ένα σύνολο A από ενέργειες α και την διάδοση αυτών στο δίκτυο. Σύμφωνα με το ανεξάρτητο κασκοδικό μοντέλο, εάν ο κόμβος u εκτελέσει μια ενέργεια σε χρόνο t και ο κόμβος v ακολουθεί τον u , τότε ο u θα προσπαθήσει να τον επηρεάσει και θα το πετύχει με πιθανότητα p(u,v). Αυτή η πιθανότητα είναι ανεξάρτητη από τις προσπάθειες άλλων κόμβων που πιθανώς να κάνουν το ίδιο. Εάν ο u τα καταφέρει και ο v δεν έχει εκτελέσει ξανά την ενέργεια α, τότε θα το κάνει σε χρόνο t+1.

Τέλος θεωρούμε πως υπάρχει ένας κόμβος ο $\Omega \in V$,που αρχικοποιεί τις διαδόσεις πληροφορίας του δικτύου μας και όλοι οι κόμβοι του δικτύου τον ακολουθούν. Υπάρχει δηλαδή arc(Ω,u) , για κάθε u.

Ορίζουμε ένα σύνολο Fa^{+}(u) για κάθε u, ως το σύνολο των κόμβων που πιθανώς επηρέασαν τον u. Αντίστοιχα το Fa^{-}(u) είναι το σύνολο των κόμβων που σίγουρα απέτυχαν να επηρεάσουν τον u. Έτσι η πιθανότητα μιας ενεργείας δίνεται ως

$$La(G) = \prod_{u \in V} Pa^{+}(u) * Pa^{-}(u)$$

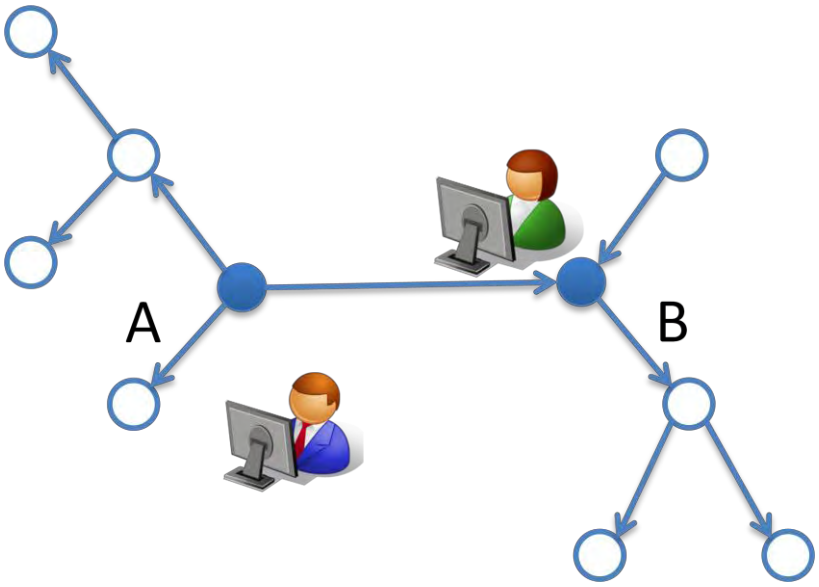
όπου

$$Pa^{+}(u) = \begin{cases} 1 & , \quad \alpha\nu\ Fa^{+}(u) = 0 \\ 1 - \prod_{u \in Fa^{+}(u)} 1 - p(u,v), & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

και

$$Pa^{-}(u) = \prod_{u \in Fa^{-}(u)} 1 - p(u,v)$$

Το τελευταίο που μας μείνε να βρούμε είναι πως θα ξέρουμε ποιοι κόμβοι μπορεί να επηρέασαν και ποιοι όχι ένα κόμβο. Σύμφωνα με το ανεξάρτητο κασκοδικό μοντέλο μια ενέργεια διαδίδετε σε ακέραια βήματα. Φυσικά στα πραγματικά προβλήματα διαλέγουμε μια τιμή κατωφλιού ΔΤ. Οι κόμβοι που πιθανώς επηρέασαν τον u και ανήκουν στο $Fa^{+}(u)$ ειναι οσοι εκτελεσαν το α σε χρόνο μικρότερο του χρόνου του u και εντός διαστήματος ΔΤ. Αντίστοιχα στο $Fa^{-}(u)$ ανήκουν όσοι κόμβοι εκτέλεσαν α είτε μετά το u είτε εκτός διαστήματος ΔΤ.[12]



Εικόνα 4:
Κατευθυνόμενο γράφημα όπου ο χρήστης Α ακολουθεί τον χρηστή Β

3.3 Ο Αλγόριθμος SPINE

Όπως αναφέραμε και πιο πάνω στόχος μας είναι δοθέντος ενός γραφήματος G , κάποιων πιθανοτήτων των ακμών του, ενός συνόλου A από ενέργειες και των διαδόσεων τους, να βρούμε τις κ ακμές που είναι πιο σημαντικές για την διάδοση των ενεργειών στο γράφημα.

Ο αλγόριθμος χωρίζεται σε δυο φάσεις. Κατά την πρώτη προσπαθούμε να βρούμε μια μη-μηδενικης πιθανότητας λύση, επιλέγοντας άπληστα ακμές με βάση τον αριθμό των συμμετοχών τους στις περισσότερες διαδόσεις. Η φάση 1 εκτελείται έως ότου είτε να έχουν επιλέγει κ ακμές είτε όσο το $\log\left(L\big(G0(V,D0)\big)\right) = -\infty$, όπου

$$L\big(G0(V,D0)\big) = \sum_{a\in A} logLa(G) = \sum_{a\in A}\sum_{u\in V}(logPa^{+}(u) + logPa^{-}(u) = \sum_{u\in V}\sum_{a\in A}(logPa^{+}(u) + logPa^{-}(u)))$$

Αν από την φάση 1 υπολείπονται ακμές , τότε εκτελείτε η φάση 2 . Σε αυτήν ,σε κάθε βήμα επιλέγετε και προστίθεται η ακμή που δίνει το μεγαλύτερο κέρδος στο γράφημα.

Πιο συγκριμένα έστω Du το σύνολο των κόμβων που δεν έχουν επιλεγεί από την φάση 1 και επηρεάζουν τον u και

$$\lambda(Xu) = \sum_{\alpha\in A} \log\left(Pa^{+}(u|Xu) + \log(Pa^{-}(u|Xu)\right)$$

τότε θεωρούμε έναν πινάκα X όπου Xu(i) είναι το υποσύνολο του Du που μεγιστοποιεί το λ και ο αλγόριθμος θα πρόσθετε στο βήμα i.

Πιο αναλυτικά έχουμε πως

$$Xu(i) = \begin{cases} 0 & ,\alpha\nu\ i = 0 \\ Xu(i-1)\cup eu(i) & ,\alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

όπου $eu(i) = argmaxe \in Uu\ \{\lambda\ (Xu(i-1)\cup e)\}$ και $Uu = Du\backslash Xu(i-1)$.

Ακόμη θεωρούμε το

$$Hu(i) = \begin{cases} 0 & ,\alpha\nu\ i = 0 \\ \lambda\big(Xu(i)\big) - \lambda\big(Xu(i-1)\big), & \alpha\lambda\lambda\iota\omega\varsigma \end{cases}$$

όπου συμβολίζει το κέρδος του λ όταν η i ακμή προστεθεί στο γράφημα.

Σε κάθε επανάληψη ο SPINE υπολογίζει τα Xu και Hu για κάθε u και επιλεγεί την ακμή eu(i^*) του u που δίνει το μεγαλύτερο κέρδος. Στην συνεχεία υπολογίζονται τα $Xu^*(i^*+1)$ και $Hu^*(i^*)$ και επαναλαμβάνετε η διαδικασία έως ότου επιλεχθούν οι υπόλοιπες ακμές.[13]

Ακολουθεί μια τυπική περιγραφή του αλγορίθμου σε μορφή ψευδοκώδικα.

1: {First Phase}
2: for $i = 1$ to $|V|$ do
3: $D_0 := \emptyset$
4: $C(v) := \{D_\alpha^+(v) \neq \emptyset\ ;\ \alpha \in \mathcal{A}\}$
5: while $|D_0| < k$ and $\log L(G_0(V, D_0) = -\infty$ do
6: find u : $(u, v) \in D$ with max $n(u, v)$
7: $D_0 := D_0 \cup \{(u, v)\}$
8: $C(v) := C(v) -$ {all $D_\alpha^+(v)$ that contain (u, v)}
9: {Second Phase}
10: $D_{\text{sp}} := D_0$, $Q :=$ empty priority queue
11: $\forall v \in V$: compute $X_v(1)$, insert $H_v(1)$ to Q
12: while $|D_{\text{sp}}| \leq k$ do
13: extract $H_{v^*}(i^*)$ from Q
14: compute $X_{v^*}(i^* + 1)$, insert $H_{v^*}(i^* + 1)$ to Q
15: $D_{\text{sp}} := D_{\text{sp}} \cup \{X_{v^*}(i^*) \setminus X_{v^*}(i^* - 1)\}$
16: return $G_{\text{sp}} = G_{\text{sp}}(V, D_{\text{sp}})$

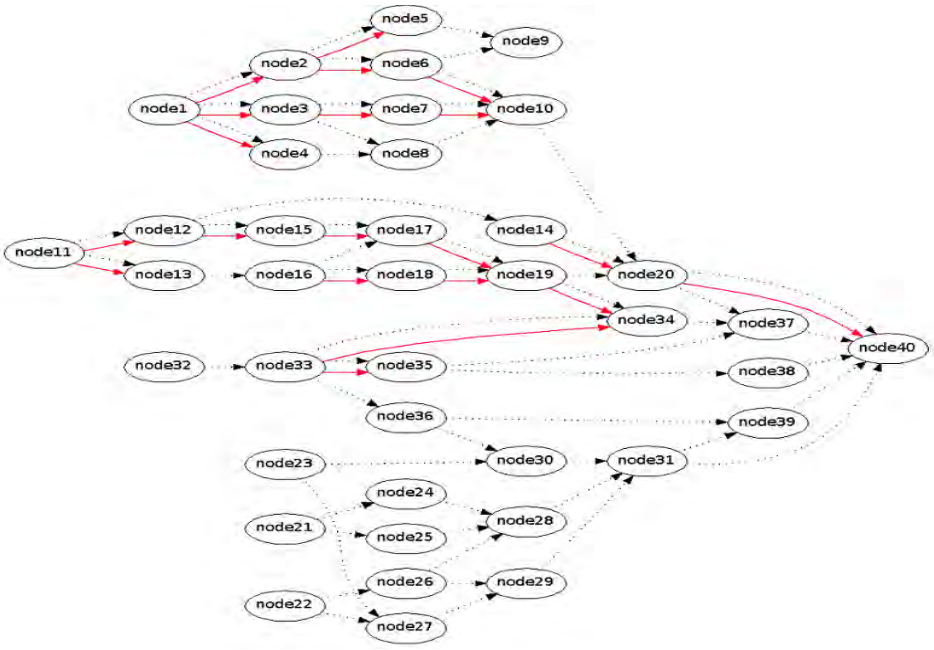
Εικόνα 5

3.4 Πολυπλοκότητα Αλγορίθμου

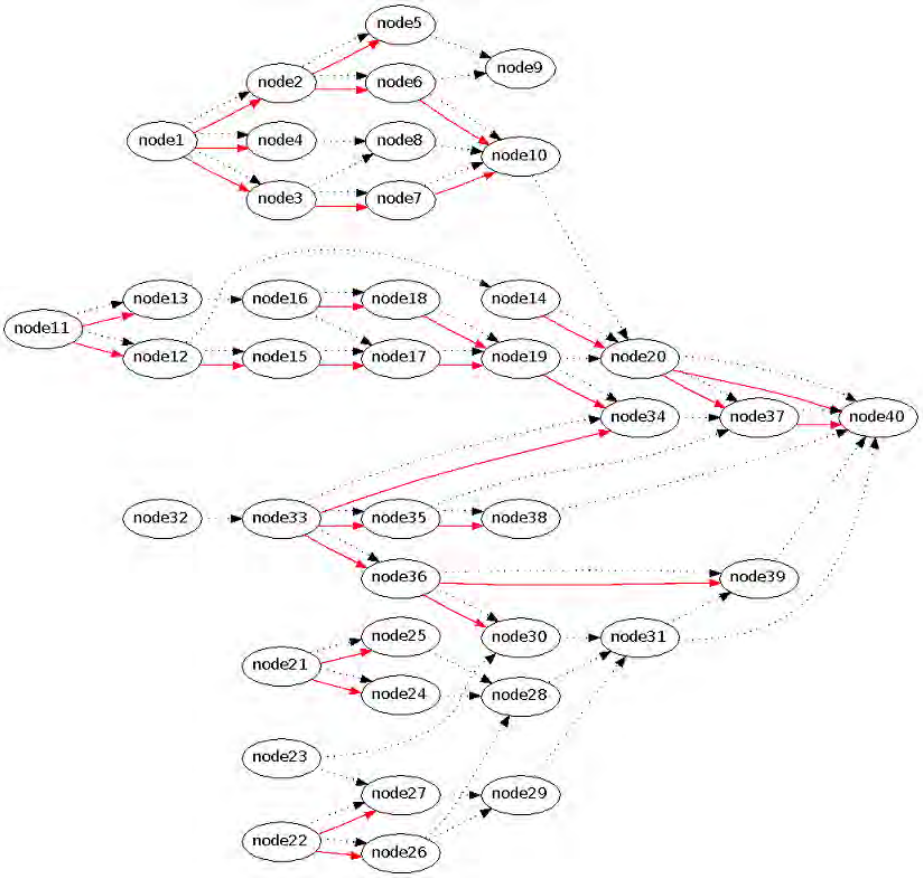
Ο υπολογισμός του $Xu(i+1)$ δοθέντος του Xu(i) είναι της τάξης του O(|Du|) , ενώ η πράξη εισαγωγής και εξαγωγής από την ουρά (γραμμές 11 ,13 εικόνας 5) είναι O(log|V|). Έτσι στην χειρότερη περίπτωση η φάση 2 θα χρειαστεί O(k(max_u(|Du|) + log|V|) χρόνο.[13]

3.5 Παραδείγματα Του Αλγόριθμου

Ακολουθούν τα αποτελέσματα 2 εκτελέσεων του αλγόριθμου για διαφορετική τιμή κ. Και στις δυο εικόνες οι κόκκινες γραμμές συμβολίζουν τις ακμές που επιλέχθηκαν , ενώ οι διακεκομμένες μαύρες είναι οι αρχικές ακμές του γραφήματος.



Εικόνα 6 : Για κ =20



Εικόνα 7 : Για κ =30

Κεφάλαιο 4

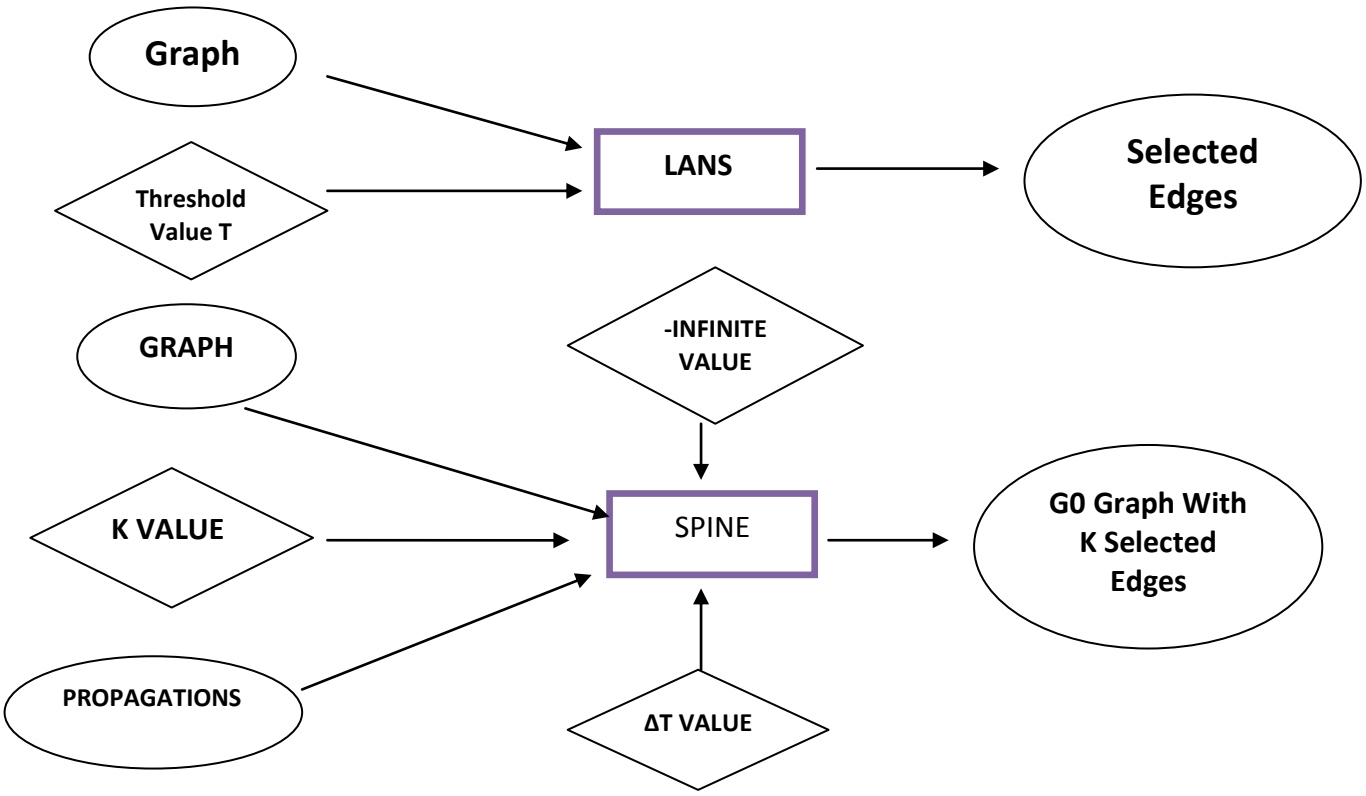
Υλοποίηση Αλγορίθμων

4.1 Μια Αναφορική Περιγραφή Στον Τρόπο Και Την Υλοποίηση

Στόχος αυτής της εργασίας, πέραν από την θεωρητική ματιά στους δυο αλγορίθμους και στην κατανόηση του προβλήματος που επιλύουν, είναι και η πρακτική μελέτη τους. Βέβαια για να μπορέσουμε να βγάλουμε ασφαλή συμπεράσματα ως προς τις συγκρίσεις των δυο, χρειάστηκε να γίνουν οι υλοποιήσεις τους, από την αρχή , επικεντρώνοντας όσο το δυνατό στην ταχύτητα εκτέλεσης.

Και οι δυο αλγόριθμοι γράφτηκαν σε γλώσσα C, με πανομοιότυπη λογική υλοποίησης.

Τα διαγράμματα που ακολουθούν περιγράφουν τις παραμέτρους και τα αρχεία εισόδου που χρειάζεται κάθε αλγόριθμος, καθώς και τις εξόδους τους.



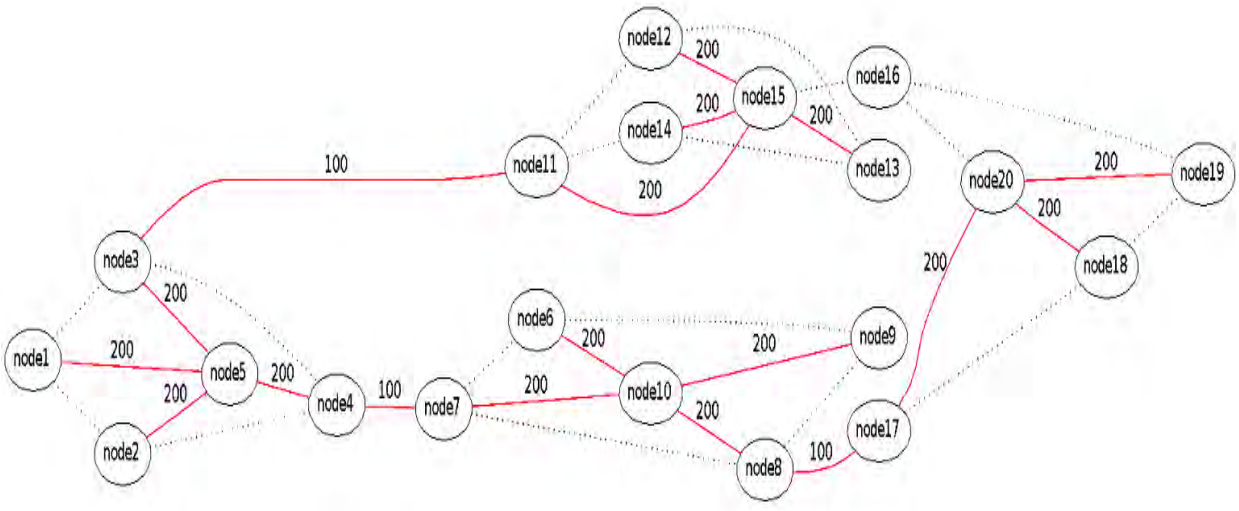
Κεφάλαιο 5

Πειραματική Αξιολόγηση

5.1.1 Μελέτη LANS

Ξεκινάμε την μελέτη μας από τον πρώτο αλγόριθμο που είδαμε, τον LANS. Ακολουθεί οπτική αναπαράσταση των γραφημάτων και των χρόνων εκτέλεσης.

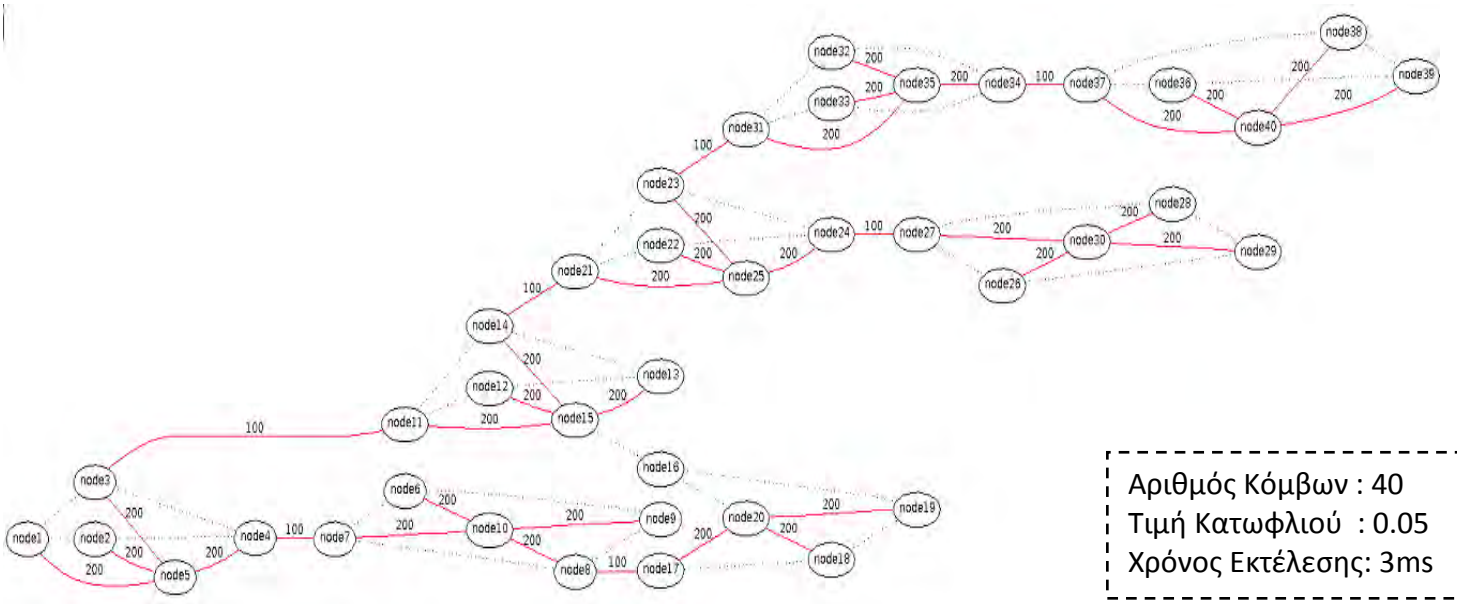
Εκτέλεση Πρώτη:



Εικόνα 8.1

Αριθμός Κόμβων : 20
Τιμή Κατωφλιού : 0.05
Χρόνος Εκτέλεσης: 1ms

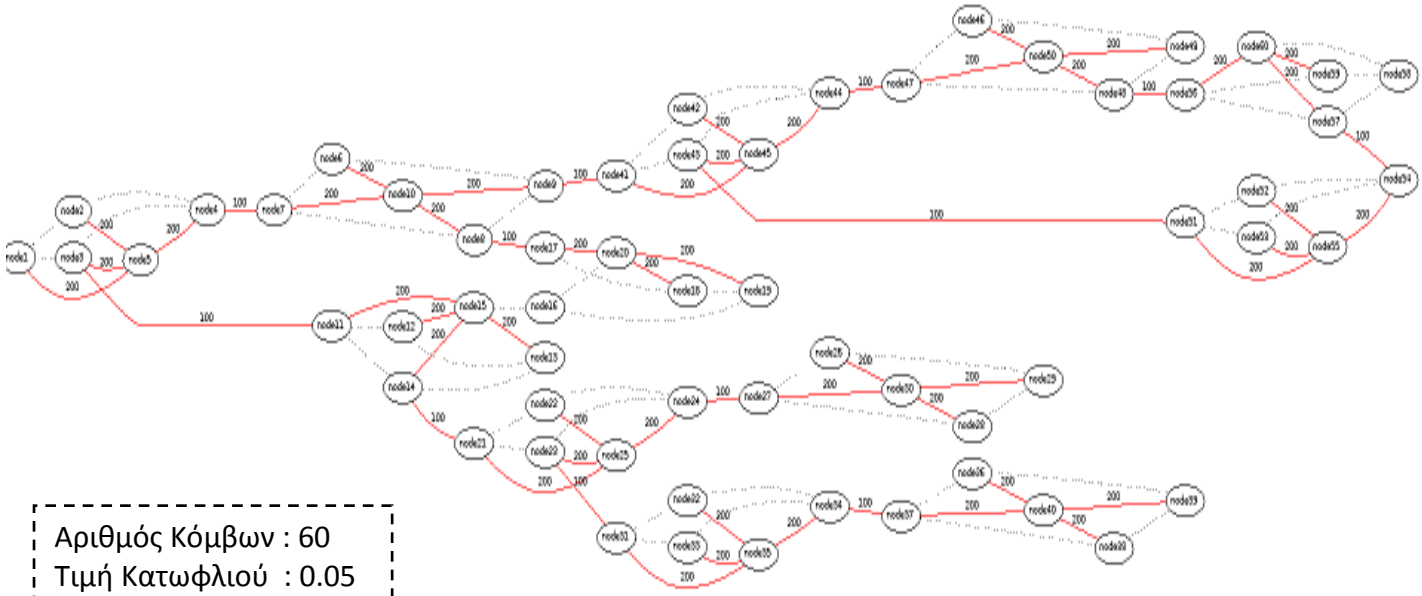
Εκτέλεση Δεύτερη:



Αριθμός Κόμβων : 40
Τιμή Κατωφλιού : 0.05
Χρόνος Εκτέλεσης: 3ms

Εικόνα 8.2

Εκτέλεση Τρίτη:

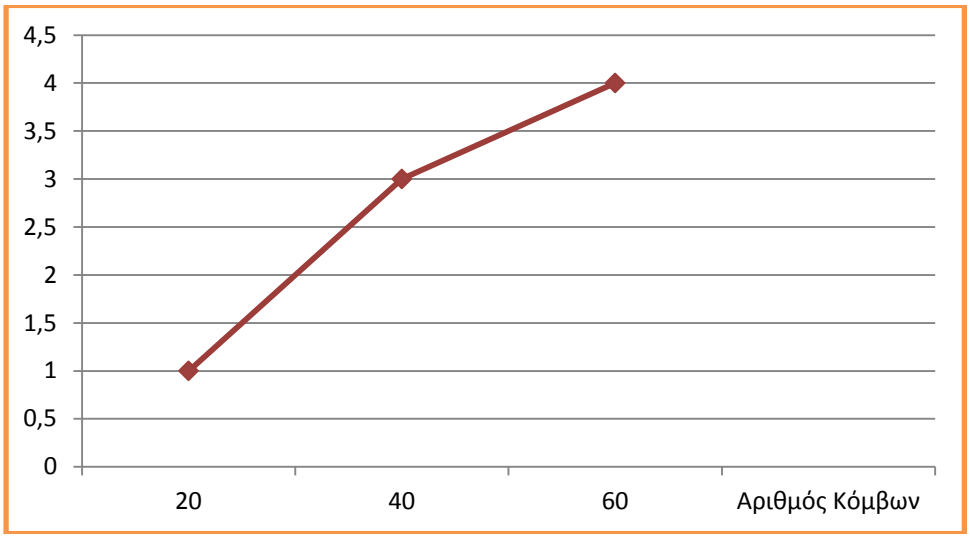


Αριθμός Κόμβων : 60
Τιμή Κατωφλιού : 0.05
Χρόνος Εκτέλεσης: 4ms

Εικόνα 8.3

5.1.2 Σχολιασμός Αποτελεσμάτων

Όπως φαίνεται από τα σχήματα (εικόνες 8.1,8.2,8.3) ο αλγόριθμος καταφέρνει κάθε φορά να μας δώσει ένα συνεκτικό αραιοποιημένο γράφημα. Σε κάθε εκτέλεση ο χρόνος αυξάνετε ανάλογα με τον αριθμό των κόμβων. Παρόλο που τα γραφήματα που χρησιμοποιηθήκαν είναι σχετικά μικρά, είναι διακριτή η αύξηση του χρόνου εκτέλεσης, όπως φαίνετε και στο διάγραμμα που ακολουθεί.

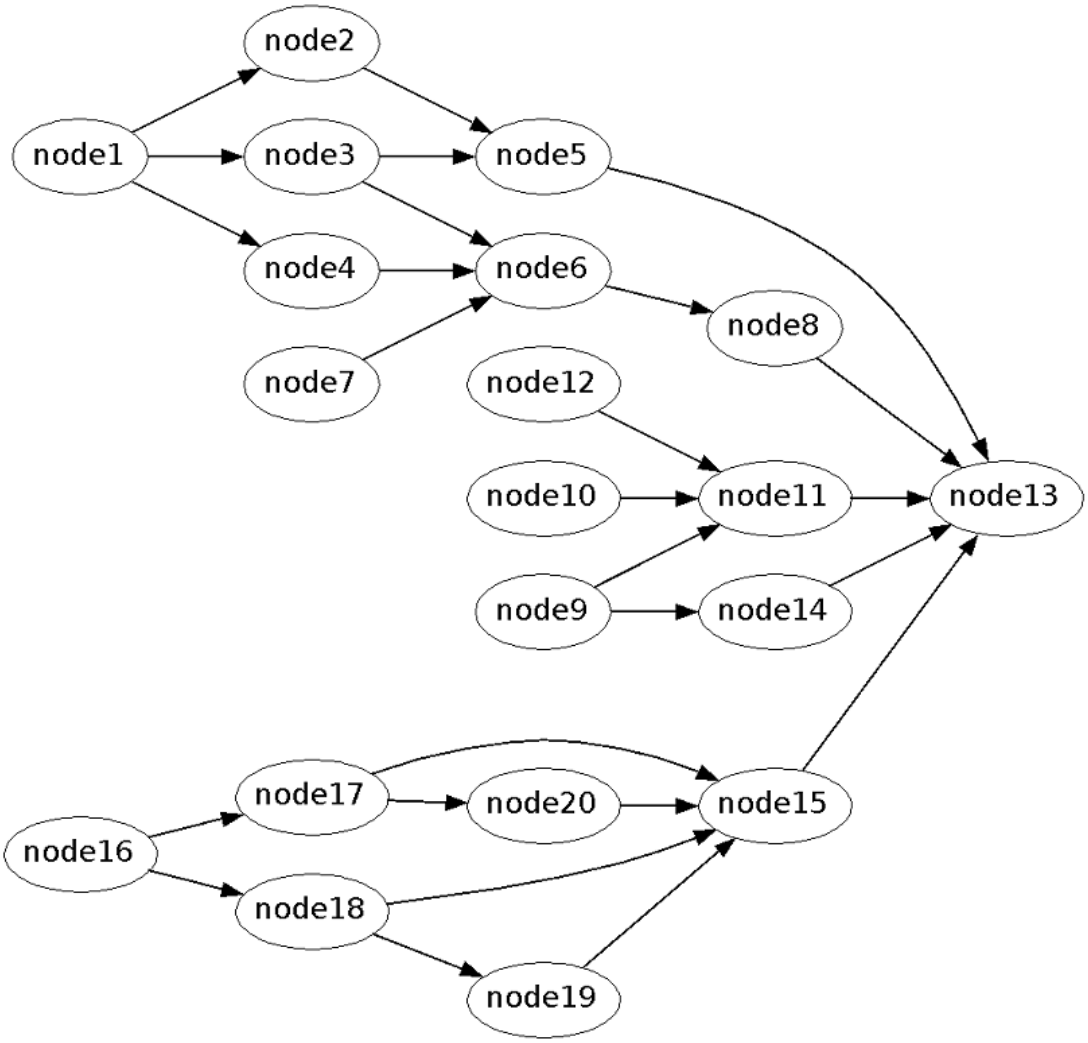


Εικόνα 9 : Διάγραμμα Χρόνου Εκτέλεσης

5.2.1 Μελέτη SPINE

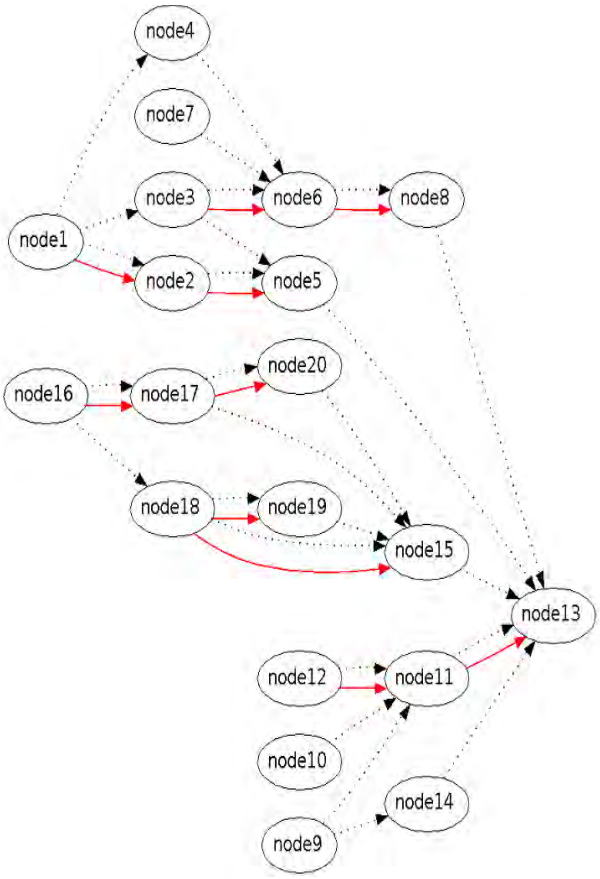
Παράδειγμα 1:

Στο πρώτο παράδειγμα θα εξετάσουμε πως επηρεάζει η τιμή ΔΤ τον αλγόριθμο και πως αλλάζει η έξοδος του.

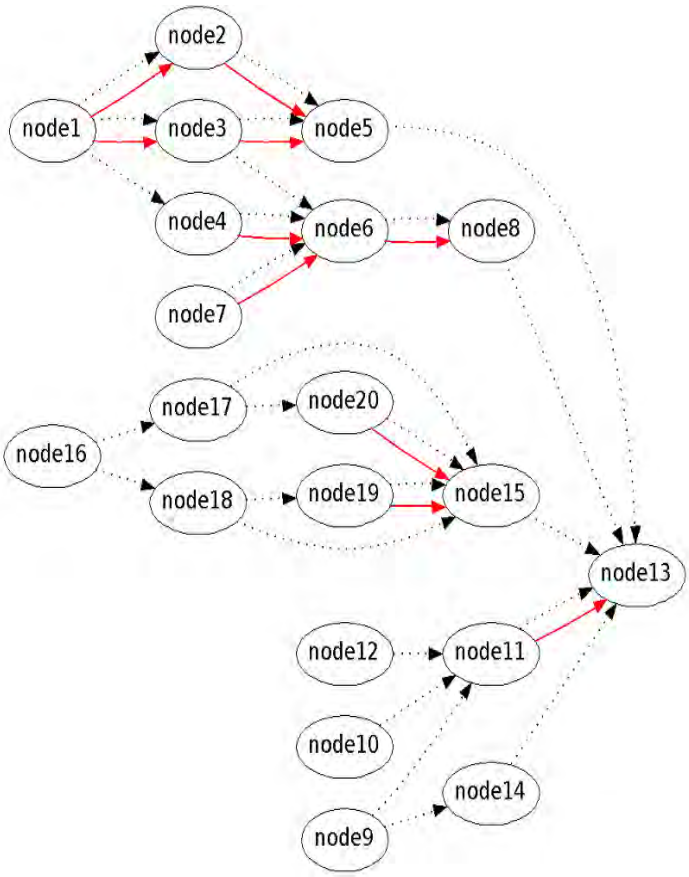


Εικόνα 10: Αρχικό Γράφημα

Στην εικόνα 11.1 φαίνονται οι ακμές που επέλεξε ο αλγόριθμος για k=10 , -inf= -1 και ΔΤ=2 ενώ στην εικόνα 11.2 είναι τα αποτελέσματα για ΔΤ=1.



Εικόνα 11.1



Εικόνα 11.2

Όπως είναι φανερό για διαφορετική τιμή του ΔΤ ο αλγόριθμος έκανε εντελώς διαφορετικές επιλογές ακμών. Αυτό βέβαια είναι αναμενόμενο και ας εξηγήσουμε το γιατί.

Όπως περιγράψαμε πιο πάνω ο αλγόριθμος σχηματίζει ένα σύνολο $Fa^{+}(i)$,για κάθε i, επιλέγοντας να βάλει σε αυτό έναν κόμβο j, αν ο i ακολουθεί τον j, και ο j έχει εκτελέσει την ενέργεια α πριν από αυτόν και μέσα σε διάστημα ΔΤ. Στην συνέχεια από αυτά τα σύνολα επιλέγονται οι τελικές ακμές που θα κρατήσουμε.

Δηλαδή για διαφορετικό ΔΤ έχουμε διαφορετικά σύνολα Fa^{+} και κατ' επέκταση διαφορετικές πιθανές επιλογές ως τελικές ακμές.

Αυτό φαίνετε και στα σύνολα Fa^+ και Fa^- για το παράδειγμα μας, στους πινάκες που ακολουθούν:

Κόμβος	Fa^+	Fa^-
1	-	-
2	1	-
3	1	-
4	1	-
5	2,3	-
6	3,4,7	-
7	-	-
8	6	-
9	-	-
10	-	-
11	9,10,12	-
12	-	-
13	8,11,14,15	5
14	-	9
15	18,19,20	17
16	-	-
17	16	-
18	-	16
19	18	-
20	17	-

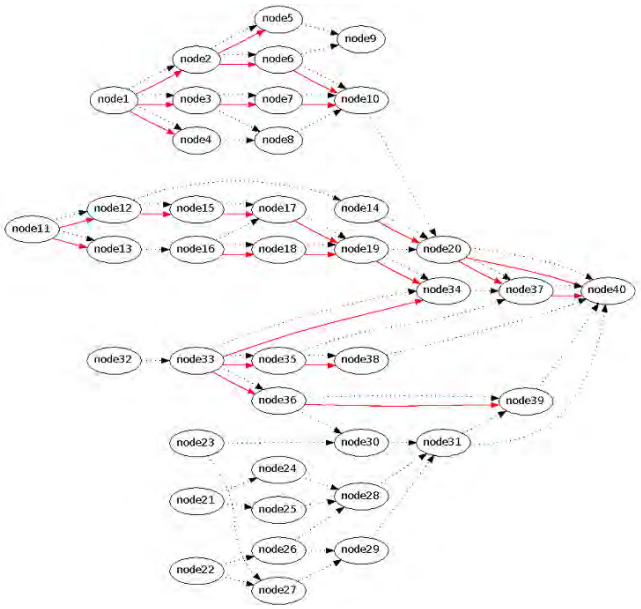
Κόμβος	Fa^+	Fa^-
1	-	-
2	1	-
3	1	-
4	-	1
5	2,3	-
6	4,7	3
7	-	-
8	6	-
9	-	-
10	-	-
11	-	9,10,12
12	-	-
13	11,14,15	5,8
14	-	9
15	19,20	17,18
16	-	-
17	-	16
18	-	16
19	-	18
20	-	17

Πινάκες Συνόλων Fa^+ και Fa^- για σχηματα εικόνας 11.1 και 11.2
Παρατηρούμε πως για διάφορους κόμβους (π.χ. κόμβος 4, κόμβος 6) τα σύνολα Fa^+ και Fa^- είναι διαφορετικά .

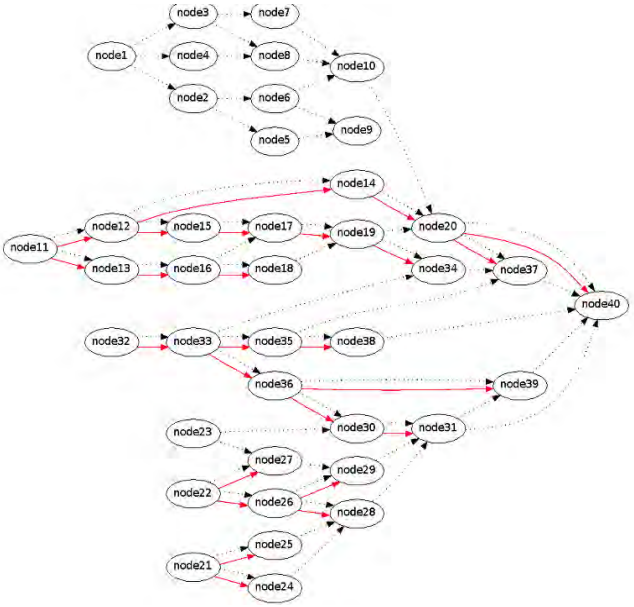
Παράδειγμα 2:

Σε αυτό το παράδειγμα θα δούμε πως μπορεί να επηρεάσει η τιμή -inf στην εκτέλεση του αλγορίθμου.
Όπως αναφέραμε στο 3.3 , η φάση 1 εκτελείται έως ότου είτε έχουμε επιλέξει κ ακμές είτε όσο το $\log\left(L\left(G0(V,D0)\right)\right) = -\infty$. Πως όμως μας επηρεάζει η τιμή -∞ ;

Στις δυο εικόνες φαίνονται τα αποτελέσματα για -inf= -1 (εικόνα 12.1) και -inf = -4 (εικόνα 12.2)



Εικόνα 12.1



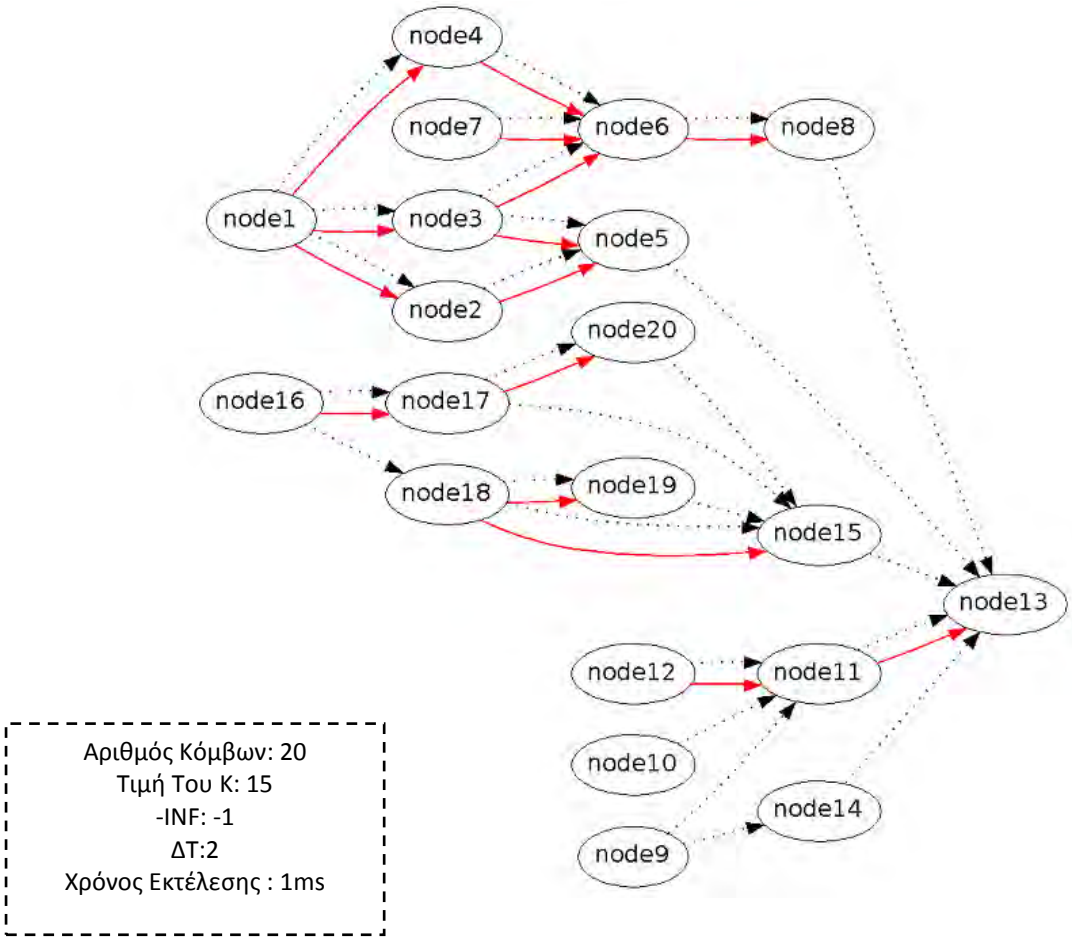
Εικόνα 12.2

Η διαφορά αποτελεσμάτων οφείλεται στο γεγονός ότι όσο πιο μικρή η τιμή του -inf τόσο λιγότερες ακμές επιλέγονται από την "άπληστη" πρώτη φάση του αλγορίθμου. Αντίστοιχα όσο μεγαλύτερη αυτή η τιμή τόσο περισσότερο αργεί η συνθήκη, $\log\left(L\left(G0(V,D0)\right)\right)$ = - ∞ να εκπληρωθεί, με αποτέλεσμα να διαλέγουμε περισσότερες ακμές από την φάση 1.
Όπως είναι λογικό οι ακμές στις δυο φάσεις επιλέγονται με διαφορετικό κριτήριο δίνοντας έτσι διαφορετικά αποτελέσματα.

Παράδειγμα 3:

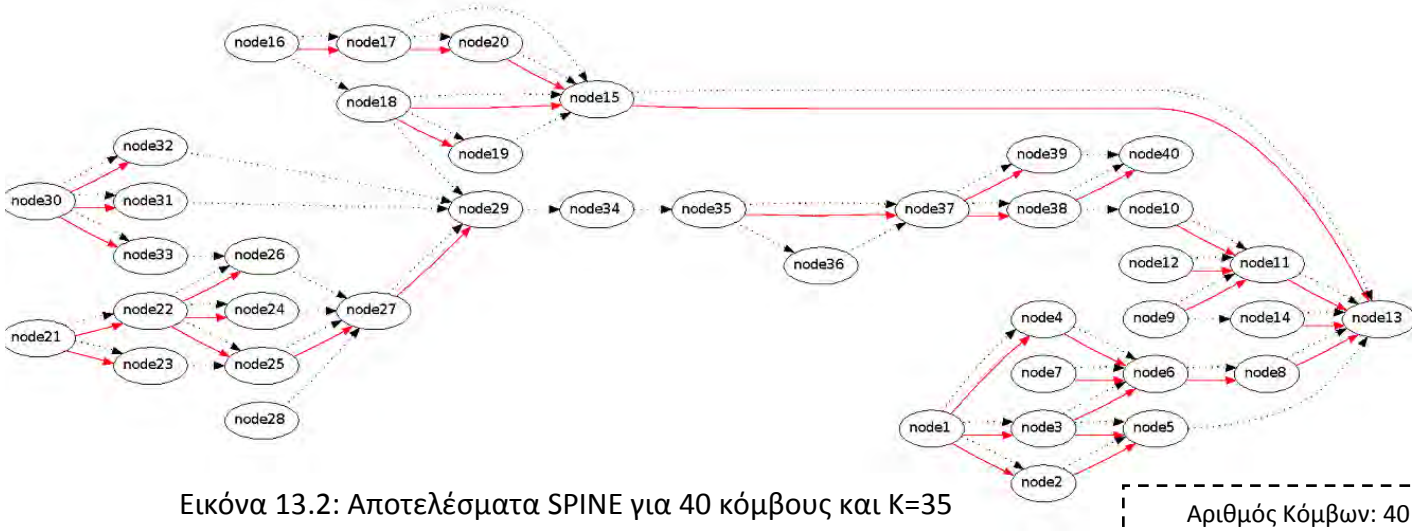
Σε αυτό το σημείο θα εξετάσουμε τους χρόνους εκτέλεσης του αλγορίθμου για διαφορά μεγέθη γραφημάτων.

Εκτέλεση Πρώτη:



Εικόνα 13.1: Αποτελέσματα SPINE για 20 κόμβους και K=15

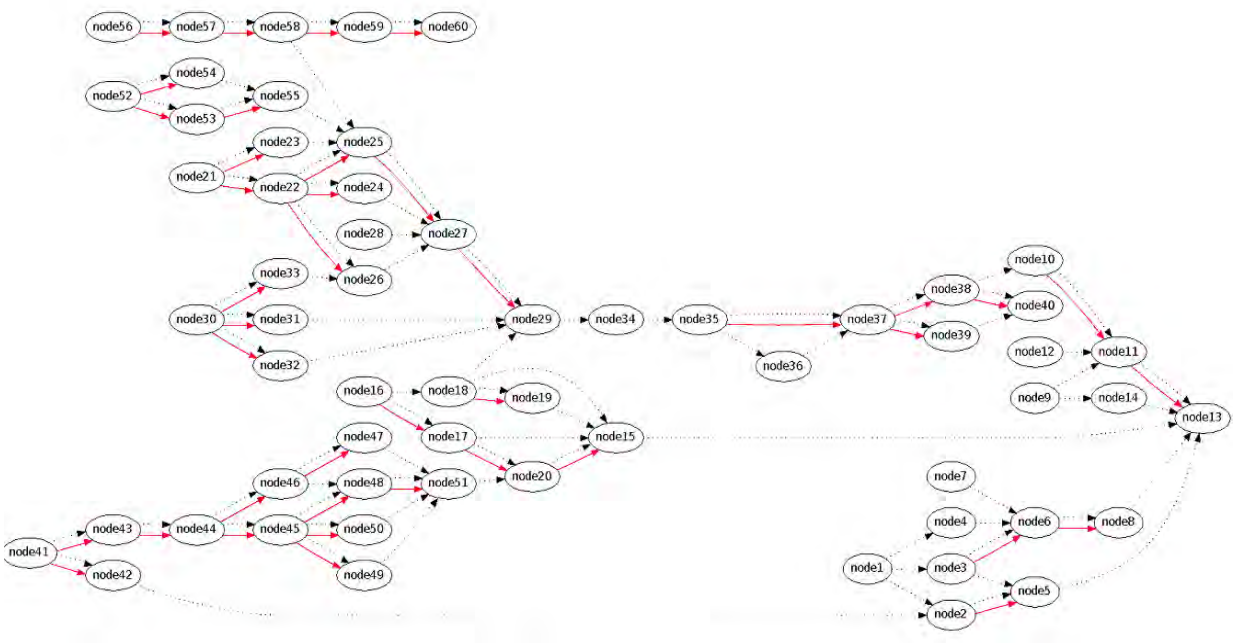
Εκτέλεση Δεύτερη:



Εικόνα 13.2: Αποτελέσματα SPINE για 40 κόμβους και K=35

Αριθμός Κόμβων: 40
Τιμή Του K: 35
-INF: -1
ΔT:2
Χρόνος Εκτέλεσης : 2ms

Εκτέλεση Τρίτη:

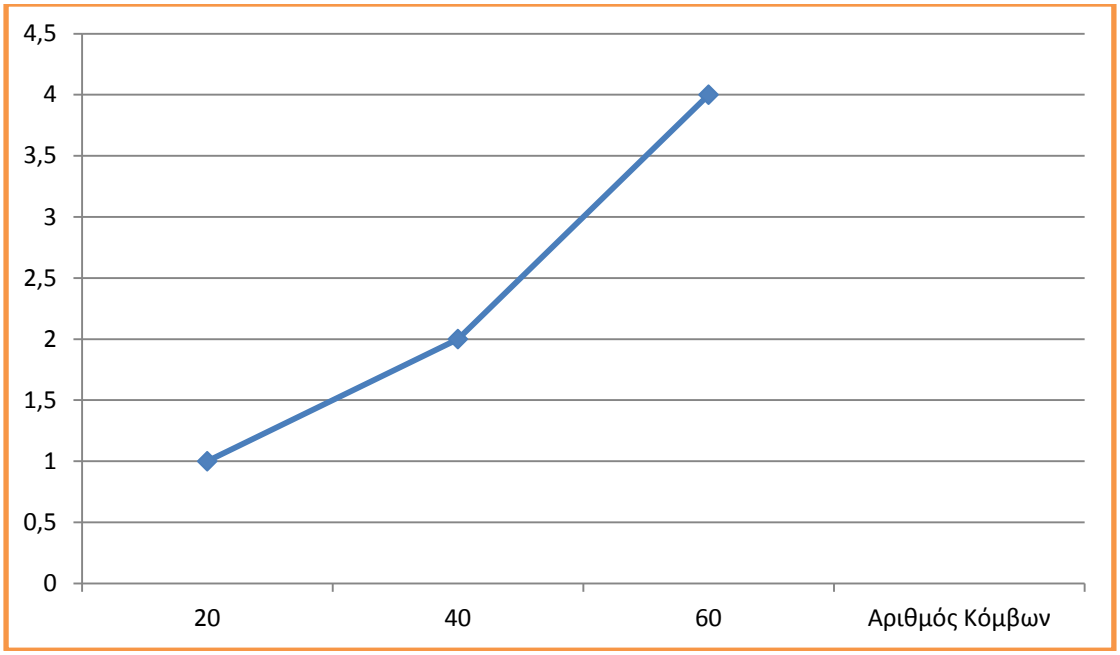


Εικόνα 13.3: Αποτελέσματα SPINE για 60 κόμβους και K=40

Αριθμός Κόμβων: 60
Τιμή Του K: 40
-INF: -1
ΔT:2
Χρόνος Εκτέλεσης : 4ms

5.2.2 Σχολιασμός Αποτελεσμάτων

Παρόλη την πολύπλοκη δομή του αλγόριθμου παρατηρούμε πως οι χρόνοι εκτέλεσης είναι πολύ γρήγοροι ,σαφώς για τα μικρής κλίμακας γραφήματα που εξετάσαμε.
Το αξιοσημείωτο είναι πως πρέπει η επιλογή των τιμών -inf και ΔΤ να γίνουν προσεκτικά, μιας που ο αλγόριθμος αλλάζει τις επιλογές ακμών, βάση αυτών των παραμέτρων.



Εικόνα 14: Διάγραμμα Χρόνου Εκτέλεσης

Κεφάλαιο 6

Σύγκριση Αλγορίθμων

Είναι η στιγμή να γίνει σύγκριση των δυο αλγορίθμων και να δούμε τα πλεονεκτήματα και μειονεκτήματα κάθε ενός.

6.1 Σύγκριση Ως Προς Τις Παραμέτρους

Όπως αναφέρθηκε και μελετήθηκε στα προηγούμενα κεφάλαια κάθε αλγόριθμος δέχεται διαφορετικές παραμέτρους και αντίστοιχα ανταποκρίνεται διαφορετικά ως προς σε αυτές.

Ο LANS δέχεται σαν μοναδική παράμετρο μια τιμή κατωφλίου T η οποία επηρεάζει στο αν μια ακμή θα θεωρηθεί σημαντική ή όχι και αντίστοιχα αν θα επιλεχτεί.

Ο SPINE όμως δέχεται 3 διαφορετικές παραμέτρους. Μια τιμή ΔT , την τιμή K (που δεν επηρεάζει το ποιες ακμές θα επιλεχτούν αλλά το πόσες) και την σταθερά για το αρνητικό άπειρο. (κεφάλαιο 5)

Συμπεριφορά ως προς ΔT: Ο αλγόριθμος επιλέγει τις ακμές από ένα σύνολο Fa^{+}. Αυτό το σύνολο δημιουργείτε με βάση την τιμή ΔT. Έτσι για διαφορετικές τιμές δημιουργούνται διαφορετικά σύνολα Fa^{+} και κατ επέκταση επιλέγονται διαφορετικές τελικές ακμές.

Συμπεριφορά ως προς την τιμή -Inf: Ο αλγόριθμος χωρίζετε σε δυο φάσεις. Στην πρώτη επιλέγονται άπληστα κάποιες ακμές έως ότου είτε να έχουμε επιλέξει K ακμές είτε το $\log\left(L\left(G0(V,D0)\right)\right.$ να γίνει -∞. Όσο μικρότερη η τιμή του -inf τόσο περισσότερες ακμές επιλέγονται από την φάση 1, παίρνοντας διαφορετικά αποτελέσματα ανάλογα με το πόσο έτρεξε η φάση αυτή.

Έχοντας αυτά υπόψη μπορούμε να πούμε πως ο LANS είναι πιο εύκολος και πιο απλός ως προς τις παραμέτρους και συνεπώς πιο σταθερός στα αποτελέσματα που δίνει αφού επηρεάζεται μόνο από την τιμή T.

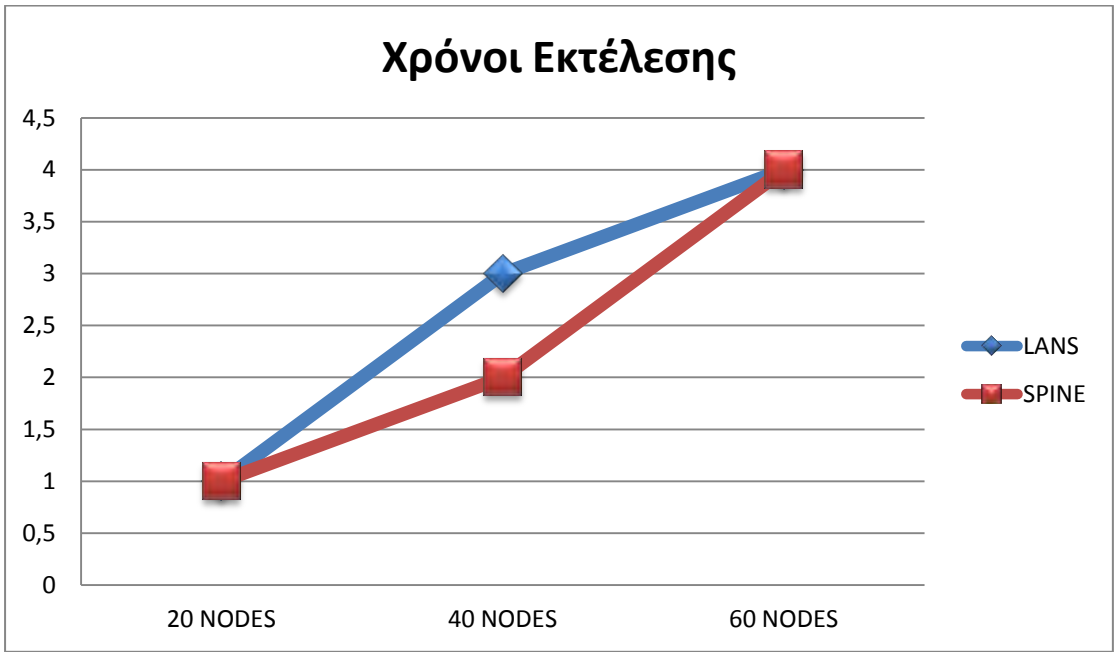
6.2 Σύγκριση Ως Προς Την Πολυπλοκότητα

Άλλος ένας βασικός παράγοντας στην σύγκριση των δυο αλγορίθμων είναι η πολυπλοκότητα τους, είτε ως προς την υλοποίηση είτε ως προς τον τρόπο παραγωγής αποτελεσμάτων.

Ο LANS εκτελείται σε μια φάση. Αρχικά υπολογίζει το βάρος Pij για κάθε κόμβο και στην συνέχεια υπολογίζει το cdf και ελέγχει για κάθε ακμή αν Pij > pa, όπου pa=1-T quantile του cdf. Η διαδικασία αυτή είναι της τάξης του O(logN). (παράγραφος 2.4)

Ο SPINE από την άλλη μεριά είναι κάπως πιο πολύπλοκος. Εκτελείται σε δυο φάσεις. Η πρώτη επιλέγει άπληστα τις |D| ακμές που συμμετέχουν στις περισσότερες διαδόσεις μιας ενέργειας α, ενώ στην δεύτερη επιλέγονται οι υπόλοιπες K-|D| ακμές υπολογίζοντας για κάθε ακμή το κέρδος Hu(i) και επιλέγει την ακμή που δίνει optimal Hu(i). Ο χρόνος εκτέλεσης είναι της τάξης O(k(max_u(|Du|) + log|V|) στην χειρότερη περίπτωση. (παράγραφος 3.4).

Μπορούμε να συμπεράνουμε πως ο LANS είναι γενικά πιο γρήγορος στην εκτέλεση αφού κάνει ελαφρύτερους και πιο απλούς υπολογισμούς. Αυτό φαίνετε και από τα αποτελέσματα χρόνων εκτέλεσης των παραγράφων 5.1.2 και 5.2.2.



Εικόνα 15: Χρόνοι Αλγόριθμων

6.3 Σύγκριση Ως Προς Τα Αποτελέσματα

Σαν τελευταίο μέτρο σύγκρισης θα εξετάσουμε τα αποτελέσματα των δυο αλγορίθμων.

Σε κάθε εκτέλεση (παράγραφος 5.1.1) ο LANS κατάφερνε να δώσει κάθε φορά ένα συνεκτικό γράφημα έχοντας επιλέξει ως κύριο κορμό τις ίδιες ακμές. Από την άλλη ο SPINE επιλέγει επηρεαζόμενος από τις 3 παραμέτρους που του δίνονται, αλλάζοντας έτσι το ποιες ακμές θα κρατήσει, κάνοντας τον λιγότερο σταθερό στα αποτελέσματα του.

6.4 Συμπεράσματα

Έχοντας υπ' όψιν όλα τα παραπάνω γίνετε φανερό πως ο LANS υπερτερεί έναντι του SPINE. Είναι αρκετά πιο απλός τόσο στην υλοποίηση όσο και στον τρόπο υπολογισμού αποτελεσμάτων. Είναι πιο εύκολος ως προς τον χρήστη αφού δέχεται μόνο μια παράμετρο και ως εκ τούτου είναι πιο σταθερός στα αποτελέσματα του.
Ο SPINE παρόλα αυτά δίνει την δυνατότητα της ευελιξίας. Δίνει την επιλογή να κρατήσουμε όσες ακμές θέλουμε μέσω της τιμής K, επιτρέπει να ελέγξουμε το διάστημα ΔT μεταξύ των γεγονότων αλλά και το πόσες ακμές θα πάρουμε από κάθε φάση της εκτέλεσης. (έμμεσα μέσω της τιμής για το αρνητικό άπειρο). Και οι δυο αλγόριθμοι βρίσκουν ένα συνεκτικό υπογράφημα του δικτύου. Είναι στην επιλογή του χρήστη και των δεδομένων που έχει, ποιον από τους δυο θα χρησιμοποιήσει.

	LANS	**SPINE**
Χρόνος	1ms -> 20 nodes 3ms -> 40 nodes 4ms -> 60 nodes	1ms -> 20 nodes 2ms -> 40 nodes 4ms -> 60 nodes
Πολυπλοκότητα	O(logN)	O(k(max_u(|Du|) + log|V|)
Παράμετροι	Threshold Value T	- INF value Number of Nodes K ΔT value
Σταθερότητα Αποτελεσμάτων	Εξαρτώνται μόνο από την τιμή T.	Εξαρτώνται και επηρεάζονται από το -inf και το ΔT.
Χρήση	Για γρήγορα αποτελέσματα, αν δεν μας ενδιαφέρει η ευελιξία.	Για γραφήματα που θέλουμε την δυνατότητα ευελιξίας αποτελεσμάτων και ακριβές αριθμό κόμβων.

Συνοπτικός Συγκριτικός Πινάκας Αλγορίθμων

Αναφορές:

[1] Networks: An Introduction Oxford University Press, March 2010 By M. E. J. Newman

[2] http://el.wikipedia.org/wiki/%CE%93%CF%81%CE%AC%CF%86%CE%BF%CF%82

[3] Leskovec J, Faloutsos C (2007) Scalable modeling of real graphs using Kronecker multiplication. In: 24th ICML. pp 497–504

[4] Bureau of Transportation Statistics (2010) http://transtats.bts.gov

[5] Bullmore E, Sporns O (2009) Complex brain networks: graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience 10: 186–198

[6] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks. pp 1.

[7] http://en.wikipedia.org/wiki/Spanning_tree

[8] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks. pp 2.

[9] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks. pp 3.

[10] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks. pp 10.

[11] Michael Mathioudakis, Francesco Bohni, Carlos Castillo, Aristides Gionis, Antti Ukkonen. Sparsification of Influence Networks. pp 1

[12] Michael Mathioudakis, Francesco Bohni, Carlos Castillo, Aristides Gionis, Antti Ukkonen. Sparsification of Influence Networks. pp 2-3

Αναφορές Εικόνων:

[1] http://lexiconcs.wikispaces.com

[2] http://ai.uom.gr/Courses/ArtificialIntelligence/Exams/June_2003.htm

[3]http://commons.wikimedia.org/wiki/File:Directed_acyclic_graph.svg

[4] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks.

[5] Nicholas J. Foti, James M. Hughes, Daniel N. Rockmore. Nonparametric Sparsification of Complex Multiscale Networks. pp 5.

[6] Michael Mathioudakis, Francesco Bohni, Carlos Castillo, Aristides Gionis, Antti Ukkonen. Sparsification of Influence Networks.

[7] Michael Mathioudakis, Francesco Bohni, Carlos Castillo, Aristides Gionis, Antti Ukkonen. Sparsification of Influence Networks. pp 6.