Ατομική Διπλωματική Εργασία

Μελέτη Προβλήματος Εύρεσης Μέγιστης Ροής

Κυριακή Χριστοδούλου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μελέτη Προβλήματος

Εύρεσης Μέγιστης Ροής

Κυριακή Χριστοδούλου

Επιβλέπων Καθηγητής

Χρύσης Γεωργίου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Χρύση Γεωργίου για την άριστη καθοδήγηση του, όπως επίσης και για την πολύτιμη βοήθεια που μου προσέφερε στις δυσκολίες που αντιμετώπισα καθ' όλη την διάρκεια της συνεργασίας μας. Ακόμα θα ήθελα να ευχαριστήσω τον κ. Γιώργο Πάλλη και την ομάδα του για την προθυμία που επέδειξαν, παραχωρώντας μου χώρο στο Nephelae Cloud Computing.

Περίληψη

Σε αυτή την Διπλωματική Εργασία αυτό που έκανα ήταν να μελετήσω σε βάθος πέντε διαφορετικούς αλγόριθμους τους Ford Fulkerson, Edmonds Karp[3], Scaling Max Flow[1], Preflow Push[1] και Scaling Preflow Push[2] οι οποίοι ο καθένας με το δικό του ξεχωριστό τρόπο λύνει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα γράφο. Αρχικά μελέτησα τον κάθε ένα ξεχωριστά και τον υλοποίησα προσπαθώντας να κρατήσω το ίδιο προγραμματιστικό επίπεδο για όλους, όσο βέβαια αυτό ήταν εφικτό. Στην συνέχεια αφού έγινε η υλοποίηση δημιούργησα συνολικά εννέα διαφορετικές στοχευόμενες τοπολογίες, όπου η κάθε μια από αυτές είχε ως στόχο την αποδυνάμωση κάποιου ή κάποιων από τους αλγορίθμους. Όταν σχεδίασα όλες τις στοχευόμενες τοπολογίες έτρεξα την κάθε μια από αυτές με όλους τους αλγορίθμους, με διαφορετικούς αριθμούς κόμβων. Αυτό έγινε επίσης και για εντελώς τυχαίες τοπολογίες. Μέσω αυτής της διαδικασίας κατάφερα να αξιολογήσω τους αλγορίθμους με βάση τον χρόνο που χρειάζονταν για να επιλύσουν το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής και να προσδιορίσω για κάθε τοπολογία ποιος είναι ο πιο αποδοτικός από τους πέντε. Τελειώνοντας την όλη διαδικασία έφτασα στο συμπέρασμα ότι κανένας από τους πέντε αλγορίθμους που μελετήθηκαν, δεν υπερτερεί πάντοτε των υπολοίπων. Για κάθε ένα βρέθηκε μια τουλάχιστον τοπολογία που να τον αποδυναμώνει.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο - Στόχος:

Το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής, είναι ένα αρκετά σημαντικό πρόβλημα που για την επίλυση του δημιουργήθηκαν πολλοί διαφορετικοί ή ακόμα και όμοιοι μεταξύ τους αλγόριθμοι, όσον αφορά την τεχνική που χρησιμοποιούν. Ο καθένας από τους αλγόριθμους που έχουμε στην διάθεση μας παρουσιάζει κάποια πλεονεκτήματα όπως επίσης και κάποια μειονεκτήματα. Αυτό ήταν που αποτέλεσε το κίνητρο για την μελέτη και την δημιουργία αυτής της Διπλωματικής Εργασίας πάνω στο πρόβλημα Εύρεσης Μέγιστης Ροής. Η σύγκριση μεταξύ κάποιων από τους αλγορίθμους που ήδη γνωρίζουμε θα μας οδηγήσει σε κάποια συμπεράσματα σχετικά με την απόδοση των αλγορίθμων και ίσως ακόμα και στον καλύτερο από αυτούς που περιλαμβάνει η Διπλωματική Μελέτη.

1.2 Σκοπός:

Σκοπός είναι να μελετηθούν κάποιοι αλγόριθμοι που επιλύουν το πρόβλημα Εύρεσης Μέγιστης Ροής, να συγκριθούν μεταξύ τους όσο αφορά την αποδοτικότητα τους και να μπορέσουμε έτσι να εξαγάγουμε κάποια συμπεράσματα. Στην Διπλωματική αυτή Εργασία μελετήθηκαν πέντε συνολικά αλγόριθμοι οι οποίοι αξιολογήθηκαν και σε στοχευόμενες τοπολογίες, σε γράφους δηλαδή με συγκεκριμένα χαρακτηριστικά αλλά και σε εντελώς τυχαίες τοπολογίες. Μελετώντας λοιπόν τα αποτελέσματα που προέκυψαν από την αξιολόγηση των πέντε αυτών αλγορίθμων προσδιορίστηκαν κάποια από τα πλεονεκτήματα και τα μειονεκτήματα του καθενός και μπορέσαμε να βρούμε τον "καλύτερο" με βάση την μελέτη μας αλγόριθμο από τους πέντε που μελετήσαμε.

1.3 Μεθοδολογία:

Οι αλγόριθμοι που μου ανατέθηκαν για μελέτη είναι οι εξής: Ford Fulkerson, Edmonds Karp[3], Scaling Max Flow[1], Preflow Push[1] και Scaling Preflow Push[2].

Αρχικά μελέτησα τον κάθε αλγόριθμο ξεχωριστά ώστε να μπορέσω να εντοπίσω τα κύρια χαρακτηριστικά του αλλά και την τεχνική που χρησιμοποιεί ο κάθε ένας από αυτούς για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα κατευθυνόμενο γράφο. Στην συνέχεια υλοποίησα και τους πέντε αλγορίθμους προσπαθώντας να μην αδικήσω κανένα από αυτούς, όσον αφορά το προγραμματιστικό μέρος της μελέτης. Επειδή το προγραμματιστικό κομμάτι είναι πολύ σημαντικό για την μετέπειτα συμπεριφορά των αλγορίθμων στα πειράματα, έπρεπε να γίνει πολύ προσεκτικά ούτως ώστε να μην επηρεαστεί ο χρόνος κάποιου αλγορίθμου λόγω της δικής μου προγραμματιστικής προσέγγισης. Αφού λοιπόν υλοποίησα και τους πέντε αλγορίθμους δημιούργησα κάποια στοχευόμενα και κάποια μη στοχευόμενα σενάρια πάνω στα οποία εργάστηκα για να πραγματοποιήσω τα πειράματα μου και να εξαγάγω τα συμπεράσματα μου. Τα στοχευόμενα σενάρια είναι τοπολογίες με συγκεκριμένα χαρακτηριστικά, που έχουν ως στόχο την αποδυνάμωση κάποιου αλγορίθμου ή ακόμα και την μελέτη συμπεριφοράς των αλγορίθμων βάση της τοπολογίας. Τα μη στοχευόμενα σενάρια είναι τυχαίες τοπολογίες με ορισμένα κάποια χαρακτηριστικά εκ των προτέρων με τυχαίους όμως γράφους για διάφορους αριθμούς κόμβων. Στην συνέχεια αφού κατασκευάστηκαν οι απαιτούμενες τοπολογίες πραγματοποιήθηκαν τα πειράματα όπου μετρήθηκε ο χρόνος που χρειαζόταν ο κάθε αλγόριθμος για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής για διαφορετικό αριθμό κόμβων. Έτρεξα τις τοπολογίες και για μικρό αριθμό κόμβων αλλά και για μεγάλο για να διαφανούν πιο καλά οι αδυναμίες του κάθε αλγορίθμου. Έτσι μπόρεσα να εξαγάγω συμπεράσματα και να διατυπώσω τις παρατηρήσεις μου για τον κάθε αλγόριθμο ξεχωριστά αλλά και συγκριτικά όλους τους αλγορίθμους μεταξύ τους.

1.4Τελικό Συμπέρασμα:

Συμπερασματικά κατέληξα στο ότι κανένας από τους αλγορίθμους δεν υπερτερεί πάντα των υπολοίπων. Αυτό σημαίνει πως κανένας δεν ήταν ο καλύτερος από όλους σε όσες τοπολογίες μελετήσαμε. Για τον κάθε αλγόριθμο υπήρξε μια τοπολογία που τον αποδυνάμωσε. Επίσης για τον κάθε αλγόριθμο υπήρξε μια τοπολογία στην οποία ήταν ο καλύτερος από όλους τους άλλους. Καταλήγουμε λοιπόν στο συμπέρασμα ότι ο κάθε αλγόριθμος από τους πέντε που μελετήσαμε έχει ένα τρωτό σημείο. Αν θέλαμε να ορίσουμε εντελώς χαλαρά ποιος από τους πέντε αλγορίθμους είναι ο καλύτερος θα επιλέγαμε αυτόν που ήταν ο καλύτερος στις περισσότερες τοπολογίες που μελετήσαμε. Στις περισσότερες λοιπόν τοπολογίες που μελετήθηκαν συμπεριλαμβανομένων και των αποτελεσμάτων των τυχαίων σεναρίων, ο καλύτερος αλγόριθμος ήταν ο Scaling Preflow Push[2]. Δεν υπερτερούσε βέβαια πάντα των υπολοίπων, διότι σε κάποιες τοπολογίες παρουσίασε πολύ μεγάλο χρόνο σε σχέση με τους υπόλοιπους αλγορίθμους αλλά κρίνοντας με βάση τις τοπολογίες που κατασκεύασα ήταν καλύτερος από τους υπόλοιπους σχεδόν σε αρκετές από αυτές.

Κεφάλαιο 2

Πρόβλημα Μέγιστης Ροής

2.1 Πρόβλημα Μέγιστης Ροής[1]:

Δεδομένου ενός Δικτύου Ροής, στόχος είναι η διευθέτηση της κίνησης ή αλλιώς Ροής, έτσι ώστε να γίνει όσο αποτελεσματική χρήση των χωρητικοτήτων των ακμών που αποτελούν το δίκτυο. Οπότε αυτό που καλούνται να κάνουν οι αλγόριθμοι που έχουν υλοποιηθεί είναι δεδομένου ενός Δικτύου Ροής να βρουν ένα αποτελεσματικό τρόπο, ούτως ώστε να αποστείλουν την μέγιστη ροή που μπορούν από την πηγή στον προορισμό.

2.2 Εφαρμογές του προβλήματος Μέγιστης Ροής:

Το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα γράφο δεν είναι μόνο ένα θεωρητικό πρόβλημα αλλά έχει πάμπολλες εφαρμογές στην πραγματική ζωή. Πολλοί επιστήμονες έχουν ασχοληθεί με την κατασκευή αλγορίθμων επίλυσης αυτού του προβλήματος για αυτό ακριβώς τον λόγο. Η επίλυση αυτού του προβλήματος σε μικρό χρόνο είναι δυνατό να αποτελέσει την λύση για μεγάλου μήκους προβλήματα. Ένα τέτοιο παράδειγμα που αξίζει να σημειώσουμε είναι ο χρονοπρογραμματισμός αεροπορικών δρομολογίων. Όπως όλοι μπορούμε να αντιληφτούμε αυτό είναι ένα πρόβλημα που όχι μόνο απαιτεί μια σωστή λύση αλλά απαιτεί και μια γρήγορη απάντηση, έχοντας ως γνώση ένα μεγάλο όγκο δεδομένων. Μια άλλη σημαντική εφαρμογή είναι η διοχέτευση νερού από κάποια πηγή σε ολόκληρη την χώρα με τον καλύτερο εφικτό τρόπο ώστε να μην γίνεται σπατάλη του νερού αλλά και να φτάνει το νερό στον προορισμό του που είναι στην συγκεκριμένη περίπτωση οι διάφορες για παράδειγμα πόλεις. Ακόμα η επίλυση του προβλήματος Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής μπορεί να δώσει λύση στα προβλήματα των Διμερών Ταιριασμάτων, Δίκτυα Μεταφοράς, Συνδεσιμότητα Δικτύων κ.α. Βλέπουμε λοιπόν ότι το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα δίκτυο είναι ένα αρκετά διαδεδομένο πρόβλημα σε πολλούς τομείς της ζωής μας χωρίς εμείς να το αντιλαμβανόμαστε. Έχουν λοιπόν κατασκευαστεί πολλοί αλγόριθμοι για την επίλυση αυτού του προβλήματος πέντε από τους οποίους επιλέγηκαν για μελέτη.

2.3Αυστηρή μορφή προβλήματος Μέγιστης Ροής[1]:

Ορίζουμε ότι ένας γράφος αποτελεί ένα δίκτυο μεταφοράς, στο οποίο οι ακμές μεταφέρουν κάποιο είδος κίνησης και οι κόμβοι λειτουργούν σαν μεταγωγείς για την μεταφορά κίνησης στις ακμές αυτές. Στο δίκτυο αυτό υπάρχουν πολλές παράμετροι που πρέπει να οριστούν:

- οι χωρητικότητες των ακμών (capacities - c_e) οι οποίες δείχνουν τις ποσότητες ροής που είναι εφικτό να περάσουν από την κάθε ακμή, μη αρνητικός ακέραιος αριθμός
- ο κόμβος πηγή (source) ο οποίος παράγει κίνηση
- ο κόμβος προορισμός (sink) ο οποίος θα απορροφήσει την κίνηση

Η κίνηση αυτή αναφέρεται ως ροή σε αυτήν τη Διπλωματική Εργασία η οποία θα παράγεται από τον κόμβο πηγή(s) και θα απορροφάται από τον κόμβο προορισμός(t).

Οπότε γενικά το Δίκτυο Ροής[1] είναι ένας κατευθυνόμενος γράφος G(V,E) όπου:

- |V| =: αριθμός των κόμβων που υπάρχουν στον γράφο
- |E| =: αριθμός των ακμών που υπάρχουν στον γράφο
- κάθε ακμή είναι συσχετιζόμενη με μία μόνο χωρητικότητα
- υπάρχει μόνο ένας κόμβος πηγή/προέλευσης(source/s)
- υπάρχει μόνο ένας κόμβος προορισμός/απόληξης(destination/sink/t)

Για την διευκόλυνση της μελέτης του προβλήματος Εύρεσης Μέγιστης Ροής έχουν γίνει κάποιες παραδοχές όπου:

- δεν υπάρχει εισερχόμενη ακμή στον κόμβο πηγή
- δεν υπάρχει εξερχόμενη ακμή από τον κόμβο προορισμός
- υπάρχει τουλάχιστον μια ακμή προσκείμενη σε κάθε κόμβο
- όλες οι χωρητικότητες είναι ακέραιοι αριθμοί

Ορισμός της Ροής:

Ροή είναι μια συνάρτηση f: $E \rightarrow R^+$ που αντιστοιχίζει κάθε ακμή σε έναν μη αρνητικό, πραγματικό αριθμό έτσι να διατηρούνται οι εξής συνθήκες:

1. Συνθήκη Χωρητικότητας:
   για κάθε ακμή $e \in E$ ισχύει ότι $0 \leq f(e) \leq c_e$. Δηλαδή ισχύει ότι δεν μπορώ να περάσω μέσα από μια ακμή περισσότερη ροή από την χωρητικότητα της ακμής
2. Συνθήκη Διατήρησης:
   για κάθε κόμβο $v \in V$ εκτός από τους κόμβους {s,t}, ισχύει ότι

$$\sum_{e \text{ προς τον } v} f(e) = \sum_{e \text{ από τον } v} f(e)$$

Δηλαδή ισχύει ότι η συνολική ροή που φτάνει σε ένα κόμβο ισούται με την συνολική ροή που εξέρχεται από τον κόμβο αυτό

Η τιμή της συνολικής Ροής $|f|$ μπορεί να οριστεί ως:

$$|f| = \sum_{\text{e από τον s}} f(e) \quad \text{ή} \quad \sum_{\text{e' προς τον t}} f(e')$$

Σε ένα Δίκτυο Ροής είναι σημαντικό να ορίσουμε ένα άνω όριο για την τιμή της Ροής που μπορεί να αποσταλεί από την πηγή και να απορροφηθεί από τον προορισμό. Βασικό τροχοπέδη ως προς την ύπαρξη μεγάλων ροών είναι το εξής: Αν χωρίσουμε όλους τους κόμβους του Δικτύου Ροής σε δύο σύνολα A και B, έτσι ώστε ο κόμβος πηγή(s) να ανήκει στο σύνολο A και ο κόμβος προορισμός(t) να ανήκει στο σύνολο B, $A \cap B = \emptyset$ και $A \cup B = V$, οποιαδήποτε ροή από τον s προς τον t θα πρέπει σε κάποιο σημείο να περνάει από το σύνολο A στο σύνολο B, άρα θα χρησιμοποιεί κάποια από τις χωρητικότητες ακμής από το A στο B. Αυτό μας δείχνει ότι οποιοσδήποτε τέτοιος διαχωρισμός θέτει ένα όριο ως προς την μέγιστη δυνατή τιμή της ροής. Ο διαχωρισμός στα σύνολα A και B ονομάζεται *αποκοπή(A,B)* [1].

Οπότε έχουμε ότι η χωρητικότητα της αποκοπής c(A,B) ισούται με το άθροισμα των χωρητικοτήτων των ακμών από το A στο B. Φτάνουμε τελικά στο συμπέρασμα ότι η τιμή της Μέγιστης Ροής είναι μικρότερη ή ίση από ελάχιστη αποκοπή, δηλαδή από το $\min_{A,B}\{c(A,B)\}$.

Ακόμα υπάρχουν και κάποιες επιπρόσθετες έννοιες που είναι απαραίτητο να οριστούν, η *ευθύδρομη* και η *ανάδρομη* λειτουργία. Ευθύδρομη είναι η λειτουργία όπου γίνεται προώθηση ροής προς τα εμπρός σε ακμές που έχουν περίσσευμα χωρητικότητας ενώ Ανάδρομη είναι η λειτουργία όπου γίνεται προώθηση ροής προς τα πίσω σε ακμές που μεταφέρουν ήδη ροή ώστε να την εκτρέψουμε σε διαφορετική κατεύθυνση.

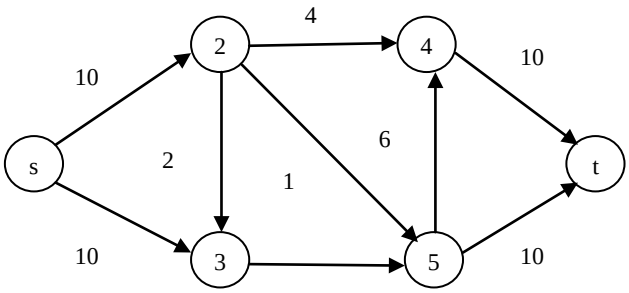
Επίσης μια άλλη έννοια που πρέπει να ορίσουμε είναι η έννοια του *υπολειπόμενου δικτύου.* Δεδομένου λοιπόν ενός Δικτύου Ροής G και μια ροής f πάνω στο G, το υπολειπόμενο Δίκτυο $Gf(V_f,E_f)$ ορίζεται ως εξής:

- οι κόμβοι του υπολειπόμενου Δικτύου είναι ίδιοι με τους κόμβους του Δικτύου Ροής G ανά πάσα στιγμή, δηλαδή $Vf = V$
- για κάθε ακμή e=(u,v) με ροή $f(e)<c_e$, συμπεριλαμβάνουμε στο Ef, την ακμή e=(u,v) με χωρητικότητα $c_e - f(e)$: ευθύδρομη ακμή
- για κάθε ακμή e=(u,v) με ροή $f(e)>0$, συμπεριλαμβάνουμε στο Ef, την ακμή $e^{\alpha}=(v,u)$ με χωρητικότητα $f(e)$: ανάδρομη ακμή

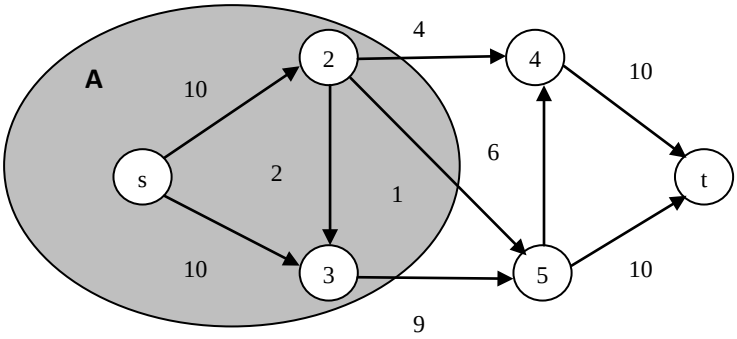
Διαισθητικά εφόσον ξέρουμε ότι από κάθε ακμή του Δικτύου Ροής G μπορεί να προκύψουν μία η δύο ακμές στο γράφο Gf, φτάνουμε στο συμπέρασμα ότι $|E| \leq |Ef| \leq 2|E|$

2.4 Παράδειγμα Δικτύου Ροής:

Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα Δικτύου Ροής



Η Μέγιστη Ροή σε αυτό το Δίκτυο Ροής είναι ίση με 14 και ισούται με την ελάχιστη αποκοπή η οποία φαίνεται στο πιο κάτω σχήμα.



c(A,B) = 4+1+9 = 14

Κεφάλαιο 3

Αλγόριθμοι Ford Fulkerson

Στο κεφάλαιο αυτό θα παρουσιάσουμε τον αλγόριθμο Ford Fulkerson[1]αλλά και τους αλγορίθμους που αποτελούν παραλλαγές του.

3.1 Ford Fulkerson

Περιγραφή Αλγορίθμου:

Ο Αλγόριθμος Ford Fulkerson[1] χρησιμοποιεί μια συγκεκριμένη τεχνική, όπου κατασκευάζει διαδρομές επαύξησης σε ένα υπολειπόμενο γράφημα προσπαθώντας να αποστείλει όση πιο πολλή ροή είναι δυνατόν από την πηγή στον προορισμό. Ακολουθώντας λοιπόν ουσιαστικά ένα άπληστο αλγόριθμο επιλέγει κάθε φορά την νέα διαδρομή επαύξησης. Ξεκινώντας η τιμή της ροής για το γράφημα είναι μηδέν. Στην συνέχεια προσπαθεί να βρει ένα μονοπάτι διαμέσου του οποίου θα προσπαθήσει να αποστείλει όση πιο πολλή ροή μπορεί από την πηγή στον προορισμό. Ανάλογα με τις στενώσεις που υπάρχουν στις ακμές που εμπεριέχονται στην διαδρομή που επιλέγηκε, στέλνεται και η ανάλογη ροή. Με άλλα λόγια η μέγιστη ροή που μπορεί να σταλεί μέσα από μια διαδρομή είναι όσο η πιο στενή ακμή που υπάρχει στο μονοπάτι. Ο αλγόριθμος φροντίζει ώστε οι συνθήκες Διατήρησης και Χωρητικότητας που περιγράφηκαν πιο πάνω να διατηρούνται σε κάθε βήμα που κάνει. Ο αλγόριθμος αυτός τερματίζει είτε όταν δεν υπάρχει μονοπάτι για να σταλεί ροή από την πηγή στον προορισμό, είτε όταν δεν υπάρχει άλλη ροή που να προέρχεται από την πηγή. Η συνολική ροή που αποστέλλεται μέσα από το Δίκτυο Ροής αυξάνεται σταδιακά με βάση τις διαδρομές επαύξησης. Κάθε διαδρομή που επιλέγεται για να σταλεί μέσω αυτής ροή θα αυξήσει την συνολική ροή τουλάχιστον κατά ένα λόγω της παραδοχής που κάναμε που λέει ότι οι χωρητικότητες είναι ακέραιοι αριθμοί.

Ψευδοκώδικας:

Ford Fulkerson

Αρχικά f(e)=0 για όλα τα e του G

While υπάρχει μια διαδρομή s-t στο υπολειπόμενο γράφημα Gf

Έστω ότι η P είναι μια απλή διαδρομή s-t στο Gf

f ' = augment(f, P)

Ενημέρωσε το f σε f '

Ενημέρωσε το υπολειπόμενο γράφημα Gf σε Gf '

End While

Return f

όπου augment(f,P) τρέχει η εξής συνάρτηση

augment (f,P)

Έστω ότι b=bottleneck (P, f)

For όλες τις ακμές (u,v) που ανήκουν στην P

If e=(u,v) είναι ευθύδρομη ακμή **then**

αύξησε το f(e) στο G κατά b

Else (το (u,v) είναι ανάδρομη ακμή, και έστω e=(v,u))

μείωσε το f(e) στο G κατά b

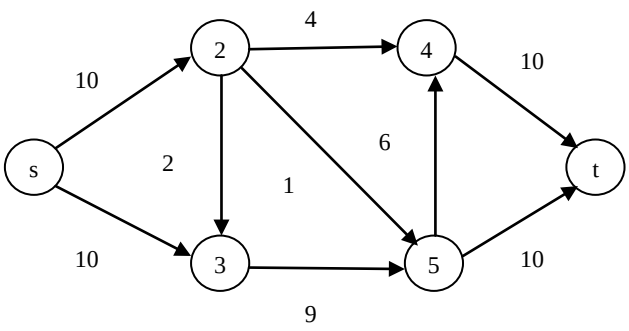
End If

End For

Return(f)

Παράδειγμα:

Δίκτυο Ροής G

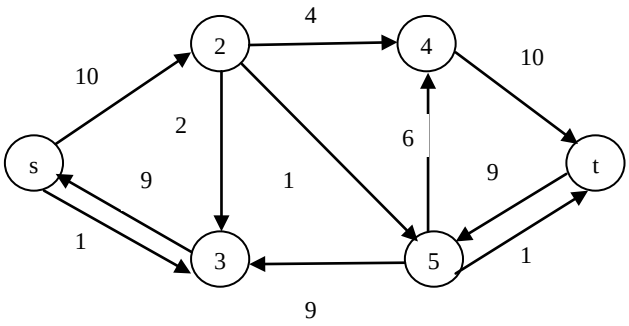


Βήμα 1:

Διαδρομή: s→3→5→t

Στένωση=9

Υπολειπόμενο Δίκτυο Ροής Gf



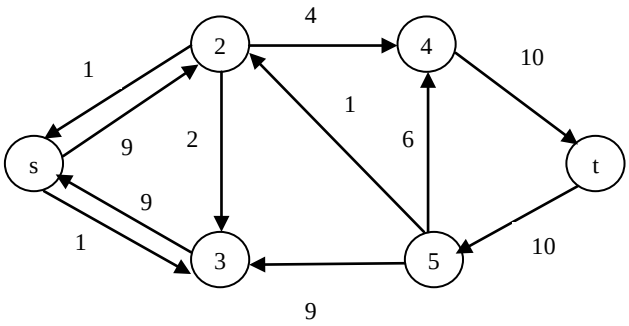
Ροή= 9

Βήμα 2:

Διαδρομή: s→2→5→t

Στένωση=1

Υπολειπόμενο Δίκτυο Ροής Gf



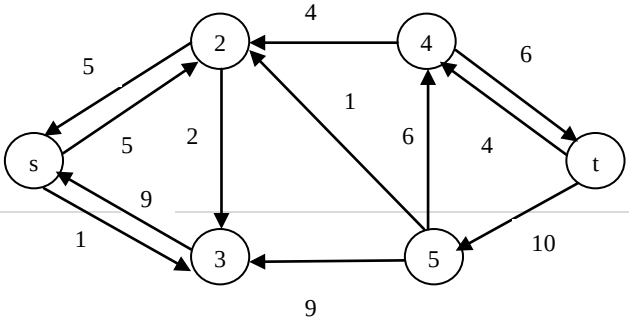
Ροή= 10

Βήμα 3:

Διαδρομή: s→2→4→t

Στένωση=4

Υπολειπόμενο Δίκτυο Ροής Gf



Ο αλγόριθμος Ford Fulkerson τερματίζει αφού δεν υπάρχουν άλλα μονοπάτια από την πηγή στον προορισμό, οπότε δεν υπάρχει κάποια διαθέσιμη διαδρομή επαύξησης και η Μέγιστη Ροή ισούται με 14.

Θεωρητική Ανάλυση Τερματισμός[1]:

Ο θεωρητικός χρόνος που χρειάζεται ο αλγόριθμος Ford Fulkerson[1]για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι

O (m*C)

όπου m = αριθμός ακμών του Δικτύου Ροής(|E|)

και C = άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή

Λεπτομέρειες Υλοποίησης:

Στην υλοποίηση αυτού του αλγορίθμου, λόγω του ότι ήταν ο πρώτος ο οποίος υλοποιήθηκε έπρεπε να εξευρεθεί ένας τρόπος ούτως ώστε η υλοποίηση του να είναι αντικειμενική αλλά και να με βοηθήσει ώστε να υλοποιήσω με παρόμοιο προγραμματιστικά τρόπο τουλάχιστον τους άλλους δύο αλγορίθμους της ομάδας του. Οπότε αυτό που έκανα ήταν να σκιαγραφήσω ένα κοινό άξονα και για τους τρείς αλγορίθμους της ομάδας Ford Fulkerson και άρχισα την υλοποίηση του αλγόριθμου. Στην λύση μου οι δομές δεδομένων που χρησιμοποίησα ήταν κυρίως πίνακες και μια λίστα για την υλοποίηση επιλογής των μονοπατιών. Ακόμα έσπασα τον κώδικα μου σε συναρτήσεις προσπαθώντας να είμαι όσο πιο κοντά γινόταν στον ψευδοκώδικα για να μπορέσω έτσι να κρατήσω όλους τους αλγορίθμους στο ίδιο προγραμματιστικό επίπεδο.

Δεν αντιμετώπισα γενικά πολλές δυσκολίες, όμως ήταν εξαιρετικά δύσκολο το γεγονός ότι έπρεπε η υλοποίηση μου να είναι πολύ προσεγμένη ούτως ώστε σε κανένα σημείο να μην αδικεί τον αλγόριθμο και να μην επηρεάσει τον μετέπειτα χρόνο του. Ακόμα μια γενική δυσκολία ήταν το ότι έπρεπε τουλάχιστον για τους αλγορίθμους της ίδιας ομάδας να μην απέχει κατά πολύ η μια λύση από την άλλη προγραμματιστικά ούτως ώστε να μην αποδυναμωθεί κάποιος από τους αλγορίθμους λόγω της δικής μου προγραμματιστικής προσέγγισης.

3.2 Edmonds Karp[3]

Περιγραφή Αλγορίθμου:

Ο αλγόριθμος Edmonds Karp[3]χρησιμοποιεί παρόμοια τεχνική με τον αλγόριθμο Ford Fulkerson[1] όπου κατασκευάζει διαδρομές επαύξησης σε ένα υπολειπόμενο γράφημα προσπαθώντας να αποστείλει όση πιο πολλή ροή είναι δυνατόν από την πηγή στον προορισμό με ένα επιπλέον βήμα,

όπου επιλέγει πρώτα τα πιο σύντομα μονοπάτια που υπάρχουν στο Δίκτυο Ροής. Αυτό το επιτυγχάνει χρησιμοποιώντας κατά βάθος ή κατά πλάτος διερεύνηση και επιλέγοντας τα πιο σύντομα μονοπάτια σαν διαδρομή επαύξησης. Αυτή είναι η βασική διαφορά ανάμεσα στους δύο αλγορίθμους. Αρχικά η τιμή της ροής για το γράφημα είναι μηδέν. Στην συνέχεια ο αλγόριθμος προσπαθεί να βρει το πιο γρήγορο μονοπάτι από τα εναπομείναντα διαμέσου του οποίου θα προσπαθήσει να αποστείλει όση πιο πολλή ροή μπορεί από την πηγή στον προορισμό. Ανάλογα με τις στενώσεις που υπάρχουν στις ακμές που εμπεριέχονται στην διαδρομή που επιλέγηκε, στέλνεται και η ανάλογη ροή. Με άλλα λόγια η μέγιστη ροή που μπορεί να σταλεί μέσα από μια διαδρομή είναι όσο η πιο στενή ακμή που υπάρχει στο μονοπάτι. Ο αλγόριθμος φροντίζει ώστε οι συνθήκες Διατήρησης και Χωρητικότητας που περιγράφηκαν πιο πάνω να διατηρούνται σε κάθε βήμα που κάνει. Ο αλγόριθμος αυτός τερματίζει είτε όταν δεν υπάρχει μονοπάτι για να σταλεί ροή από την πηγή στον προορισμό, είτε όταν δεν υπάρχει άλλη ροή που να προέρχεται από την πηγή. Η συνολική ροή που αποστέλλεται μέσα από το Δίκτυο Ροής αυξάνεται σταδιακά με βάση τις διαδρομές επαύξησης. Κάθε διαδρομή που επιλέγεται για να σταλεί μέσω αυτής ροή θα αυξήσει την συνολική ροή τουλάχιστον κατά ένα λόγω της παραδοχής που κάναμε που λέει ότι οι χωρητικότητες είναι ακέραιοι αριθμοί.

Ψευδοκώδικας:

Edmonds Karp

Αρχικά f(e)=0 για όλα τα e του G

While υπάρχει μια διαδρομή s-t στο υπολειπόμενο γράφημα Gf, επέλεξε την πιο σύντομη

Έστω ότι η P είναι μια απλή διαδρομή s-t στο Gf, η πιο σύντομη που υπήρχε διαθέσιμη

f ' = augment(f, P)

Ενημέρωσε το f σε f '

Ενημέρωσε το υπολειπόμενο γράφημα Gf σε Gf '

End While

Return f

όπου augment(f,P) τρέχει η εξής συνάρτηση επαύξησης

augment (f,P)

Έστω ότι b=bottleneck (P, f)

For όλες τις ακμές (u,v) που ανήκουν στην P

If e=(u,v) είναι ευθύδρομη ακμή **then**

αύξησε το f(e) στο G κατά b

Else (το (u,v) είναι ανάδρομη ακμή, και έστω e=(v,u))

μείωσε το f(e) στο G κατά b

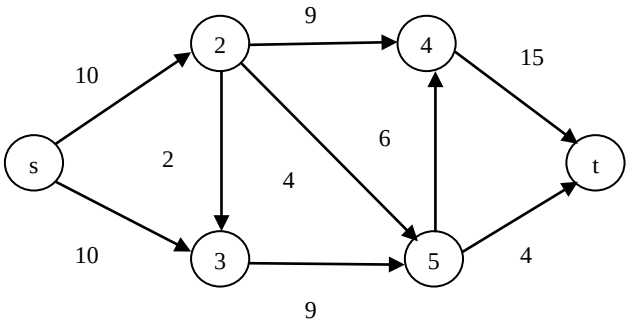
End If

End For

Return(f)

Παράδειγμα:

Δίκτυο Ροής G

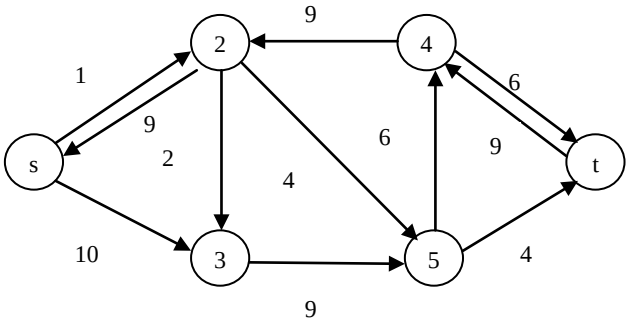


Βήμα 1:

Διαδρομή: s→2→4→t

Στένωση=9

Υπολειπόμενο Δίκτυο Ροής Gf



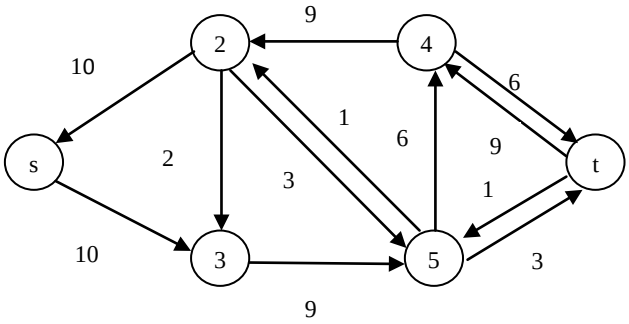
Ροή =9

Βήμα 2:

Διαδρομή: s→2→5→t

Στένωση=1

Υπολειπόμενο Δίκτυο Ροής Gf



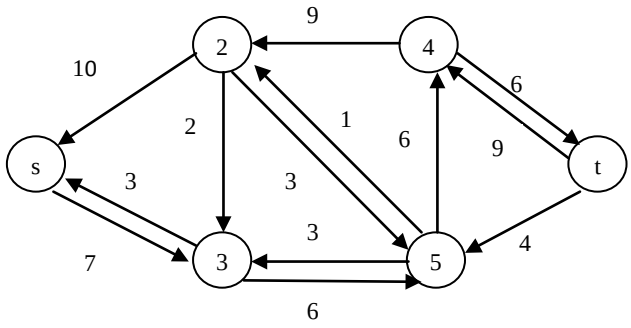
Ροή=10

Βήμα 3:

Διαδρομή: s→3→5→t

Στένωση=3

Υπολειπόμενο Δίκτυο Ροής Gf



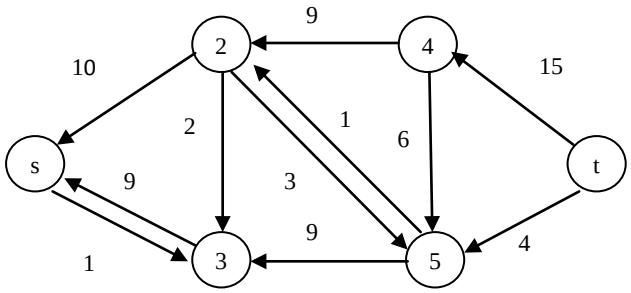
Ροή=13

Βήμα 4:

Διαδρομή: s→3→5→4→t

Στένωση=6

Υπολειπόμενο Δίκτυο Ροής Gf



Ο αλγόριθμος Edmonds Karp[3] τερματίζει αφού δεν υπάρχουν άλλα μονοπάτια από την πηγή στον προορισμό, οπότε δεν υπάρχει κάποια διαθέσιμη διαδρομή επαύξησης και η Μέγιστη Ροή ισούται με 19.

Θεωρητική Ανάλυση Τερματισμός[3]:

Ο θεωρητικός χρόνος που χρειάζεται ο αλγόριθμος Edmonds Karp[3]για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι

$O(m^2 * n)$

όπου m = αριθμός ακμών του Δικτύου Ροής(|E|)

και n= αριθμός κόμβων του Δικτύου Ροής(|V|)

Λεπτομέρειες Υλοποίησης:

Η υλοποίηση αυτού του αλγορίθμου βασίστηκε στην υλοποίηση του αλγορίθμου Ford Fulkerson[1]και με λίγες μετατροπές στον ήδη υπάρχοντα κώδικα πετύχαμε το αποτέλεσμα που θέλαμε. Όπως είδαμε και πιο πάνω η βασική διαφορά ανάμεσα στους αλγορίθμους Ford Fulkerson[1]και Edmonds Karp[3]είναι ο τρόπος επιλογής των μονοπατιών οπότε η αλλαγή που κάναμε στον κώδικα του αλγορίθμου Ford Fulkerson[1]ήταν ακριβώς για να τροποποιήσουμε την τεχνική. Στην λύση μου οι δομές δεδομένων που χρησιμοποίησα ήταν κυρίως πίνακες και μια λίστα για την υλοποίηση επιλογής των μονοπατιών. Ακόμα έσπασα τον κώδικα μου σε συναρτήσεις προσπαθώντας να είμαι όσο πιο κοντά γινόταν στον ψευδοκώδικα για να μπορέσω έτσι να κρατήσω όλους τους αλγορίθμους στο ίδιο προγραμματιστικό επίπεδο.

Δεν αντιμετώπισα γενικά πολλές δυσκολίες, εκτός από το γεγονός ότι έπρεπε να βρω ένα τρόπο ώστε με πολύ λίγες μετατροπές στον ήδη υπάρχοντα κώδικα να καταφέρω το επιθυμητό αποτέλεσμα χωρίς να αδικήσω προγραμματιστικά τον αλγόριθμο σε σχέση με τους υπόλοιπους αλγόριθμους.

3.3 Scaling Max Flow[1]

Περιγραφή Αλγορίθμου:

Ο αλγόριθμος Scaling Max Flow[1] είναι ουσιαστικά μια παραλλαγή του αλγορίθμου Ford Fulkerson[1]ως προς την τακτική επιλογής των μονοπατιών. Με μια μετατροπή στον ήδη υπάρχοντα κώδικα η οποία εισάγει την έννοια της κλιμάκωσης έχουμε τον αλγόριθμο Scaling Max Flow[1]. Θέλοντας να βελτιωθούν οι επιλογές των μονοπατιών και να επιλέγονται τα πιο φαρδιά μονοπάτια πρώτα για αποστολή ροής από την πηγή στον προορισμό εντάχθηκε η παράμετρος Δ. Το Δ είναι η μεγαλύτερη δύναμη του δύο που είναι μικρότερη από την μέγιστη χωρητικότητα που έχουν οι ακμές που προέρχονται από την πηγή. Εντάσσοντας την παράμετρο αυτή στην τεχνική επιλογής μονοπατιών αυτό που κερδίζουμε είναι ότι επιλέγονται διαδρομές που έχουν υπολειπόμενη χωρητικότητα τουλάχιστον ίση με Δ. Δουλεύουμε λοιπόν με δυνάμεις του για αυτό και όταν δεν υπάρχουν διαδρομές με υπολειπόμενη χωρητικότητα τουλάχιστον ίση με Δ, διαιρούμε το Δ δια δύο. Έτσι ο αλγόριθμος αυτός τερματίζει όταν η παράμετρος Δ φτάσει να είναι μικρότερη του ενός και αυτό συμβαίνει επειδή όπως ξέρουμε με βάση τις παραδοχές που έχουν γίνει, δεν υπάρχει χωρητικότητα που να μην θετικός, ακέραιος αριθμός.

Ψευδοκώδικας:

Scaling Max Flow[1]

Αρχικά f(e)=0 για όλα τα e του G

Αρχικά θέσε το Δ ίσο με την μεγαλύτερη δύναμη του 2 που δεν είναι μεγαλύτερη από την μέγιστη χωρητικότητα των ακμών που εξέρχονται από το s: $\Delta \leq \max_{e \text{ από το } s} c_e$

While $\Delta \geq 1$

While υπάρχει μια διαδρομή s-t στο υπολειπόμενο γράφημα $G_f(\Delta)$

Έστω ότι η P είναι μια απλή διαδρομή s-t στο $G_f(\Delta)$

f ' = augment(f, P)

Ενημέρωσε το f σε f '

Ενημέρωσε το υπολειπόμενο γράφημα Gf(Δ)

End While

Δ=Δ/2

End While

Return f

όπου augment(f,P) τρέχει η εξής συνάρτηση

augment (f,P)

Έστω ότι b=bottleneck (P, f)

For όλες τις ακμές (u,v) που ανήκουν στην P

If e=(u,v) είναι ευθύδρομη ακμή **then**

αύξησε το f(e) στο G κατά b

Else (το (u,v) είναι ανάδρομη ακμή, και έστω e=(v,u))

μείωσε το f(e) στο G κατά b

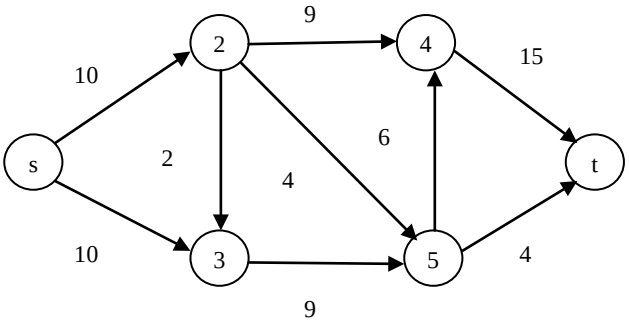
End If

End For

Return(f)

Παράδειγμα:

Δίκτυο Ροής G



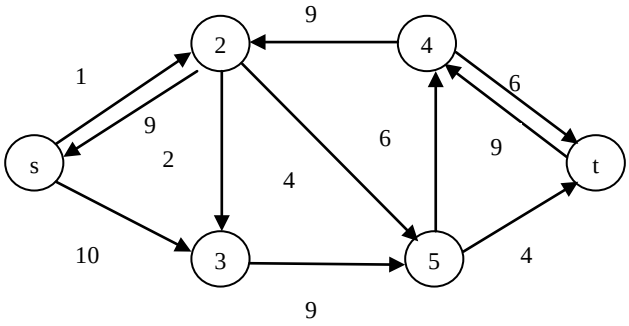
Βήμα 1:

Δ=8

Διαδρομή: s→2→4→t

Στένωση=9

Υπολειπόμενο Δίκτυο Ροής Gf



Ροή= 9

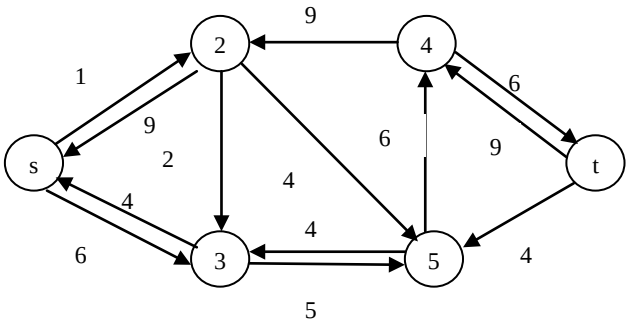
Βήμα 2:

Δ=4

Διαδρομή: s→3→5→t

Στένωση=4

Υπολειπόμενο Δίκτυο Ροής Gf



Ροή=13

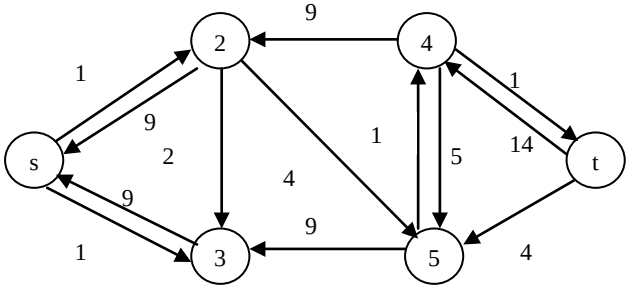
Βήμα 3:

Δ=4

Διαδρομή: s→3→5→4→t

Στένωση=5

Υπολειπόμενο Δίκτυο Ροής Gf



Ροή=18

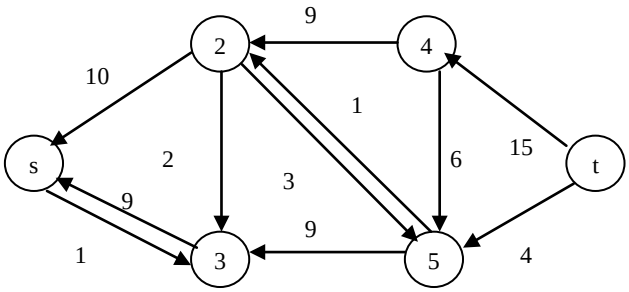
Βήμα 4:

Δ=2

Διαδρομή: s→2→5→4→t

Στένωση=1

Υπολειπόμενο Δίκτυο Ροής Gf



Ο αλγόριθμος Scaling Max Flow τερματίζει όταν το Δ γίνει μικρότερο του ενός και η μέγιστη ροή ισούται με 19.

Θεωρητική Ανάλυση Τερματισμός:

Ο θεωρητικός χρόνος που χρειάζεται ο αλγόριθμος Scaling Max Flow[1] για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι

$O(m^{2*}\log_2 C)$

όπου m = αριθμός ακμών του Δικτύου Ροής(|E|)

και C = άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή

Λεπτομέρειες Υλοποίησης:

Η υλοποίηση του αλγορίθμου αυτού βασίστηκε καθαρά στην υλοποίηση του αλγορίθμου Ford Fulkerson[1]διότι όπως ανέφερα πιο πάνω, ουσιαστικά είναι μια παραλλαγή προσθέτοντας την επιπλέον παράμετρο Δ. Δεν άλλαξε λοιπόν τίποτε παρά μόνο το ότι προσθέσαμε κάποιους υπολογισμούς για να βρούμε το αρχικό Δ. Στην λύση μου οι δομές δεδομένων που χρησιμοποίησα ήταν κυρίως πίνακες και μια λίστα για την υλοποίηση επιλογής των μονοπατιών. Ακόμα έσπασα τον κώδικα μου σε συναρτήσεις προσπαθώντας να είμαι όσο πιο κοντά γινόταν στον ψευδοκώδικα για να μπορέσω έτσι να κρατήσω όλους τους αλγορίθμους στο ίδιο προγραμματιστικό επίπεδο και να μην τους αδικήσω προγραμματιστικά.

Κεφάλαιο 4

Αλγόριθμοι Preflow Push[1]

Στο κεφάλαιο αυτό θα παρουσιάσουμε τον αλγόριθμο Preflow Push[1] αλλά και τον αλγόριθμο Scaling Preflow Push[2] που αποτελεί παραλλαγή του.

Είναι απαραίτητο να εξηγηθούν κάποιες έννοιες ώστε να γίνει πιο εύκολη η κατανόηση των αλγορίθμων που είναι οι εξής:

- Προροή είναι μια συνάρτηση f η οποία αντιστοιχίζει την κάθε ακμή e σε ένα μη αρνητικό, πραγματικό αριθμό f:E→R^+.
- Συνθήκη Χωρητικότητας: είναι αυτή που ήδη ορίσαμε.
- Συνθήκη Διατήρησης:

  Στην θέση της συνθήκης Διατήρησης απαιτούνται μόνο ανισότητες. Κάθε κόμβος πρέπει να έχει εισερχόμενη ροή τουλάχιστον ίση με την εξερχόμενη, όχι ακριβώς ίση όπως συνέβαινε στους προηγούμενους αλγορίθμους.

  Για κάθε κόμβο ν εκτός του κόμβου πηγή s, έχουμε ότι το άθροισμα της ροής που περνά από τις ακμές προς τον ν είναι μεγαλύτερο ή ίσο με το άθροισμα της ροής που περνά από τις ακμές που φεύγουν από τον ν.

  Η διαφορά μεταξύ των δύο δηλαδή της ροής που εισέρχεται στον ν, και της ροής που εξέρχεται ονομάζεται πλεόνασμα - excess(e_f(ν)).
- Υπάρχει και σε αυτούς τους αλγορίθμους η έννοια του υπολειπόμενου γραφήματος σε μια προροή.
- Η απόδοση ετικετών είναι μια συνάρτηση h: V→$Z_{\geq 0}$ από τους κόμβους στους μη αρνητικούς ακέραιους.
- Επίσης λέμε ότι μια απόδοση ετικετών και μια προροή είναι συμβατές αν ισχύουν:

  1. Συνθήκη προσέλευσης και απόληξης όπου h(t)=0 και h(s)=n

  2. Συνθήκες ρύθμισης κλίσης όπου για όλες τις ακμές (v,w) που ανήκουν στο σύνολο των ακμών του υπολειπόμενου γραφήματος, έχουμε h(v)≤h(w)+1

Η διαφορά ύψους μεταξύ του κόμβου προσέλευσης(πηγή) και του κόμβου απόληξης(προορισμός) έχει ως στόχο να ξεκινά η ροή από μεγάλο ύψος και να πηγαίνει "κατηφορικά". Η συνθήκη ρύθμισης κλίσης έχει ως στόχο να γίνεται σταδιακά η κάθοδος ούτως ώστε να φτάσει στην απόληξη(προορισμός)

4.1 Preflow Push[1]

Περιγραφή Αλγορίθμου:

Οι αλγόριθμοι που ανήκουν στην ομάδα του αλγορίθμου Preflow Push[1] δουλεύουν με μια εντελώς διαφορετική φιλοσοφία σε σχέση με τους αλγορίθμους που ανήκουν στην ομάδα του Ford Fulkerson[1]για αυτό άλλωστε τους διαχωρίσαμε. Συγκεκριμένα ενώ οι αλγόριθμοι που περιγράψαμε πιο πάνω αυξάνουν την ροή επιλέγοντας διαδρομές επαύξησης στο υπολειπόμενο γράφημα οι αλγόριθμοι της ομάδας Preflow Push[1] αυξάνουν τη ροή ακμή προς ακμή. Η τροποποίηση της ροής σε μια ακμή θα παραβιάζει ουσιαστικά την συνθήκη Διατήρησης, έτσι ο αλγόριθμος πρέπει κατά την λειτουργία του να διατηρεί κάτι άλλο που δεν συμπεριφέρεται εξίσου καλά με μια ροή. Οπότε αυτό που κάνει ουσιαστικά ο αλγόριθμος είναι να διατηρεί μια προροή και θα δουλεύει μέχρι να την μετατρέψει σε ροή. Ο αλγόριθμος είναι βασισμένος στο ότι η ροή βρίσκει τον δρόμο της "κατηφορικά". Τα "υψώματα" για αυτόν τον αλγόριθμο είναι οι ετικέτες h(v) που διατηρούνται για κάθε κόμβο. Θα προωθείται λοιπόν ροή από κόμβους με υψηλότερες ετικέτες σε κόμβους με χαμηλότερες ετικέτες έχοντας σαν βάση το ότι η ροή προχωρά "κατηφορικά".

Ψευδοκώδικας:

```
Preflow Push[1]
  Αρχικά h(v)=0 για όλα τα v≠s και h(s)=n, και
  f(e)=c_e για όλα τα e=(s,v) και f(e)=0 για όλες τις υπόλοιπες ακμές
  While υπάρχει κόμβος που v≠t με πλεόνασμα e_f(v)>0
    Έστω ότι ν ένας κόμβος με πλεόνασμα
    If υπάρχει w όπου μπορεί να εφαρμοστεί push(f, h, v, w) then
      push(f, h, v, w)
    Else
      relabel(f, h, v)
  End While
Return(f)
push(f, h, v, w)
  Είναι εφαρμόσιμη αν e_f(v)>0, h(w)<h(v) και (v,w) ανήκουν στο σύνολο των ακμών του
υπολειπόμενου γραφήματος
  If e=(v, w) είναι ευθύδρομη ακμή then
    θέσε δ=min(e_f(v), c- f(e)) και
```

αύξησε το f(e) κατά δ

If (v, w) είναι ανάδρομη ακμή **then**

θέσε e=(v, w), δ=min(e_f(v), f(e)) και

μείωσε το f(e) κατά δ

Return(f, h)

relabel(f, h, v)

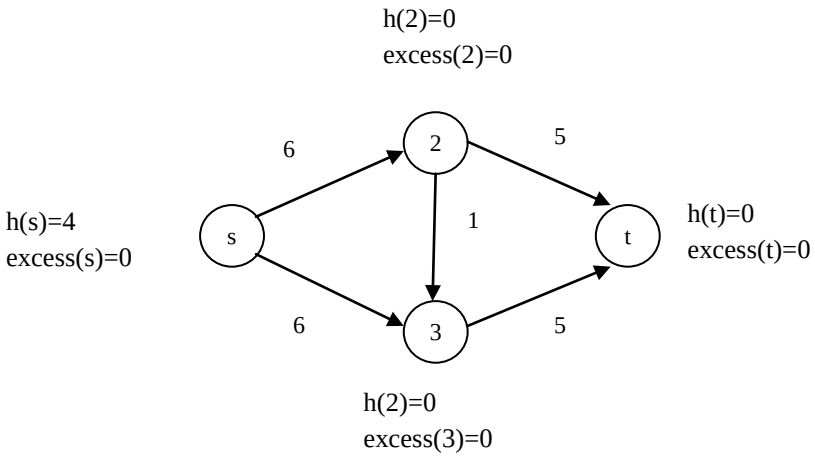
Είναι εφαρμόσιμη αν $e_f(v)>0$ και για όλες τις ακμές (v, w) που ανήκουν στο σύνολο των ακμών του υπολειπόμενου γραφήματος έχουμε $h(w)\geq h(v)$

Αύξησε το h(v) κατά 1

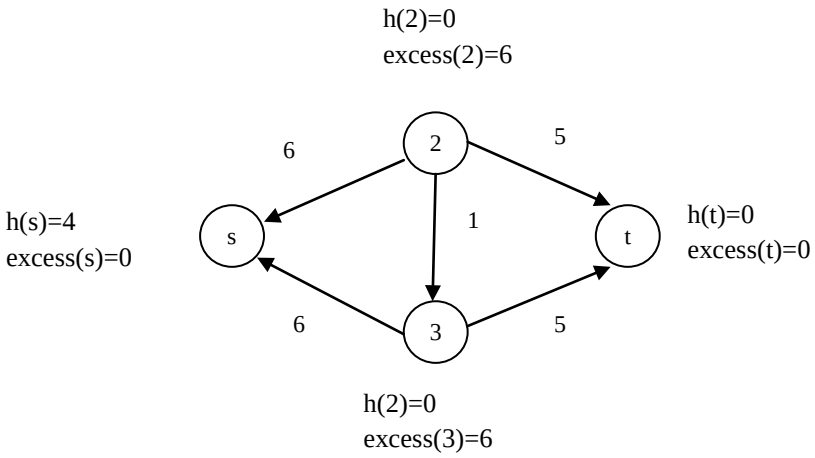
Return(f, h)

Παράδειγμα:

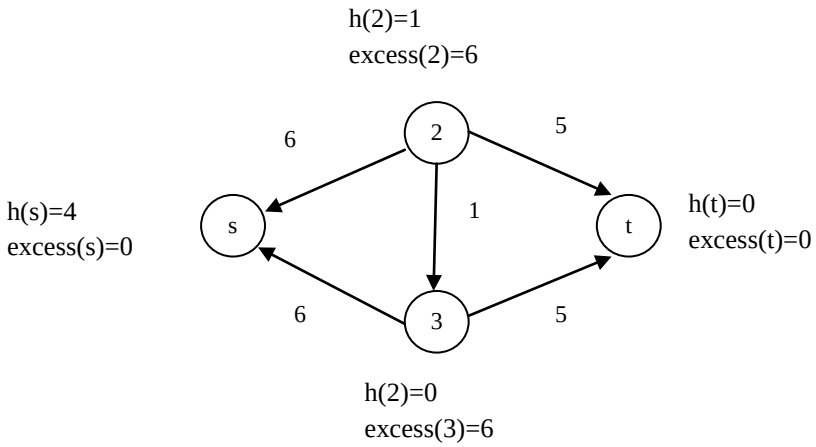
Δίκτυο Ροής G



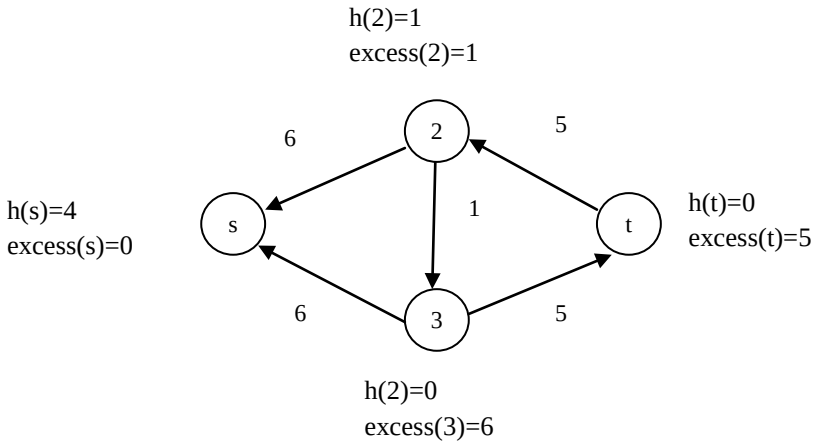
Βήμα1:



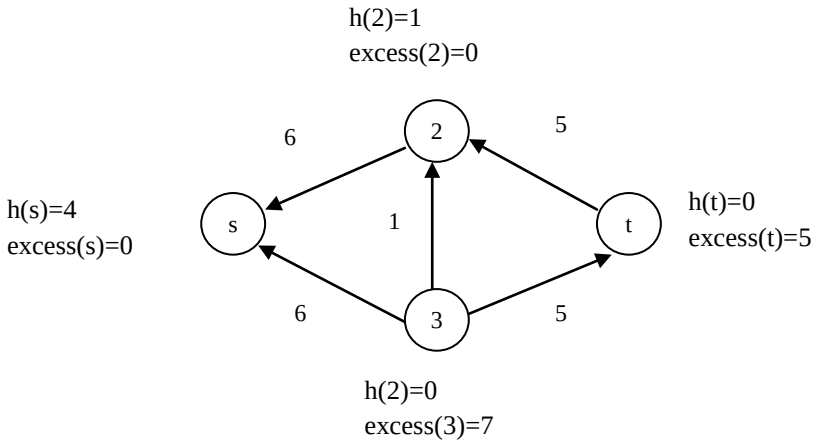
Βήμα 2:



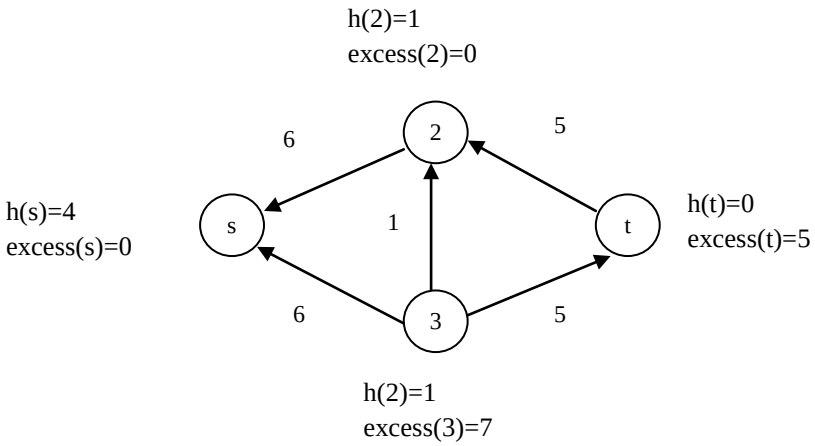
Βήμα 3:



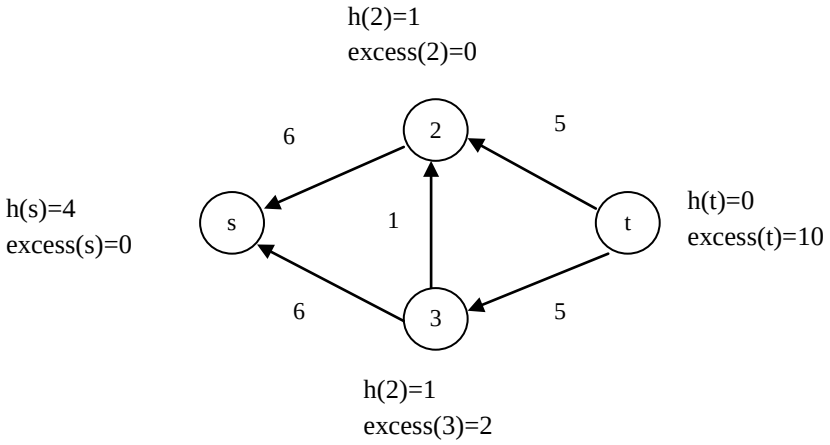
Βήμα 4:



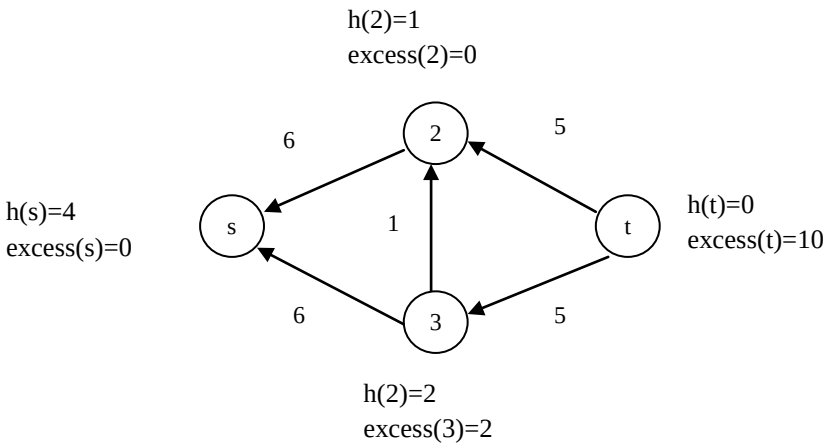
Βήμα 5:



Βήμα 6:

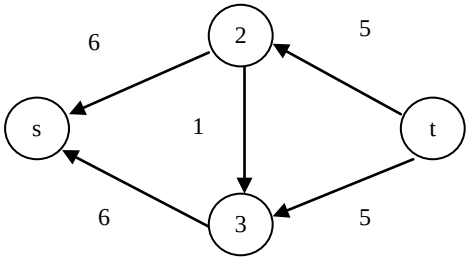


Βήμα 7:



Βήμα 8:

h(2)=1
excess(2)=1



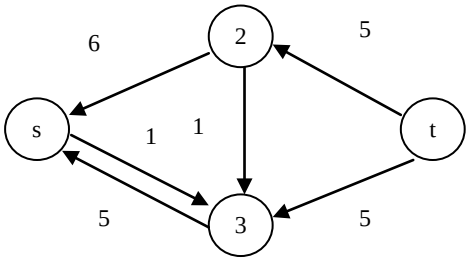
h(s)=4
excess(s)=0

h(t)=0
excess(t)=10

h(2)=2
excess(3)=1

Βήμα 9:

h(2)=1
excess(2)=1



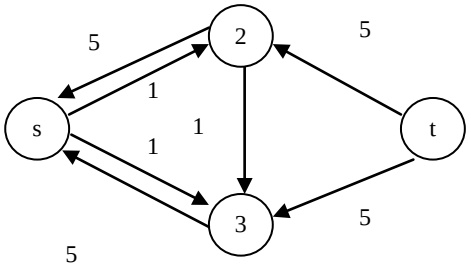
h(s)=4
excess(s)=0

h(t)=0
excess(t)=10

h(2)=5(relabeling until h(s)+1)
excess(3)=0

Βήμα 10:

h(2)=5(relabeling until h(s)+1)
excess(2)=0



h(s)=4
excess(s)=0

h(t)=0
excess(t)=10

h(2)=5
excess(3)=0

Ο αλγόριθμος Preflow Push τερματίζει όταν οι κόμβοι που έχουν excess γίνουν πιο "ψηλοί" από τον κόμβο πηγή, έχουν δηλαδή μεγαλύτερη ετικέτα και η μέγιστη ροή ισούται με 10.

Θεωρητική Ανάλυση:

Ο θεωρητικός χρόνος που χρειάζεται ο αλγόριθμος Preflow Push[1] για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι
Ο ($m*n^2$)
όπου m = αριθμός ακμών του Δικτύου Ροής(|E|)
και n = αριθμός κόμβων του Δικτύου Ροής(|V|)

Λεπτομέρειες Υλοποίησης:

Για την υλοποίηση αυτού του αλγορίθμου οι Δομές Δεδομένων που χρησιμοποιήθηκαν ήταν μια λίστα όπου αποθηκεύονται οι κόμβοι με πλεόνασμα και πίνακες. Η υλοποίηση με δυσκόλεψε ιδιαίτερα επειδή έπρεπε με κάποιο τρόπο το προγραμματιστικό κομμάτι έπρεπε να συνάδει όσο είναι εφικτό με τους αλγόριθμους της ομάδας Ford Fulkerson. Λόγω του οι αλγόριθμοι των δύο ομάδων χρησιμοποιούν μια εντελώς διαφορετική φιλοσοφία για την επίλυση του προβλήματος Μέγιστης Ροής ήταν αρκετά δύσκολο να κάνω την υλοποίηση μου δίκαιη προς όλους. Ακόμα αυτό που εντόπισα κατά την υλοποίηση του αλγορίθμου είναι ότι είχε πολύ overhead στην πράξη και έτσι αναγκάστηκα να εξεύρω τρόπους να επιλύσω αυτό το πρόβλημα. Σε γενικές γραμμές η υλοποίηση μου βασίστηκε στον ψευδοκώδικα όπως επίσης και στις παρακινήσεις της βιβλιογραφίας ώστε να γίνει μια αντικειμενική υλοποίηση.

4.2 Scaling Preflow Push ή Excess Scaling [2]

Περιγραφή Αλγορίθμου:

Ο αλγόριθμος Scaling Preflow Push[2] είναι βασισμένος στον αλγόριθμο Preflow Push[1] με κάποιες διαφοροποιήσεις που να τον κάνουν πολύ διαφορετικό. Ο αλγόριθμος Scaling Preflow Push[2] περιλαμβάνει στον τρόπο επίλυσης του την έννοια της κλιμάκωσης. Θέλοντας να βελτιωθούν οι επιλογές των ακμών και να επιλέγονται τελικά τα πιο φαρδιά μονοπάτια πρώτα για αποστολή ροής από την πηγή στον προορισμό εντάχθηκε η παράμετρος Δ. Το Δ είναι η μεγαλύτερη δύναμη του δύο που είναι μικρότερη από την μέγιστη χωρητικότητα που έχουν οι ακμές που προέρχονται από την πηγή. Εντάσσοντας την παράμετρο αυτή στην τεχνική επιλογής ακμών αυτό που κερδίζουμε είναι ότι επιλέγονται ακμές που έχουν υπολειπόμενη χωρητικότητα τουλάχιστον ίση με Δ. Δουλεύουμε λοιπόν με δυνάμεις του δύο για αυτό και όταν δεν υπάρχουν ακμές με υπολειπόμενη χωρητικότητα τουλάχιστον ίση με Δ, διαιρούμε το Δ δια 2. Έτσι ο αλγόριθμος αυτός τερματίζει όταν η παράμετρος Δ φτάσει να είναι μικρότερη του ενός και αυτό συμβαίνει επειδή όπως ξέρουμε με βάση τις παραδοχές που έχουν γίνει, δεν υπάρχει χωρητικότητα που να μην θετικός, ακέραιος αριθμός. Επίσης

άλλη μια επιπρόσθετη αλλαγή είναι το ότι ο αλγόριθμος αυτός χρησιμοποιεί όσες λίστες είναι και το μεγαλύτερο ύψος που έχουμε, για την αποθήκευση των κόμβων με πλεόνασμα και τοποθετεί τον κάθε κόμβο ανάλογα με το ύψος του στην λίστα που του αντιστοιχεί.

Ψευδοκώδικας:

Scaling Preflow Push

για κάθε ακμή (s,j) που ανήκει στις ακμές του γραφήματος στείλε τόση ροή όση και η χωρητικότητα τους

d(s)=n και d(t)=0

κάνε κάθε d(i)=1 όπου i≠s,t

$K= 1+ \lceil \log U \rceil$

For k=1 to K do

$\Delta=2^{K-\kappa}$

για κάθε κόμβο i κάνε

If($e_f(i)>\Delta/2$) **then**

LIST(d(i))

level=1;

While level<2n do

If(LIST(level)=κενό **then**

level=level+1;

Else

διάλεξε ένα κόμβο i από την λίστα LIST(level)

push/relabel

push/relabel(i)

found = false

η ακμή(i,j) γίνεται η παρούσα ακμή του κόμβου i

While found=false και (i,j) ≠null **do**

if d(i)=d(j)+1c and $r_{i,j}>0$ **then**

found=true

Else αντικατέστησε την παρούσα ακμή του κόμβου i με την επόμενη ακμή(i,j)

If found =true then

Begin

κάνε push min ($e_f(i)$, $r_{i,j,}$, $\Delta- e_f(j)$) ροή στην ακμή(i,j)

άλλαξε την εναπομένουσα χωρητικότητα $r_{i,j}$ και τα excesses $e_f(i)$, $e_f(j)$

If(νέο excess) $e_f(i)\leq \Delta/2$ **then**

διάγραψε τον κόμβο i από την λίστα LIST(d(i))

If j≠s ή t και (νέο excess) $e_f(j) > \Delta/2$ **then**

πρόσθεσε τον κόμβο j στην λίστα LIST(d(j)) και θέσε το level = level -1

End

Else

Begin

διάγραψε τον κόμβο i από την λίστα LIST(d(i))

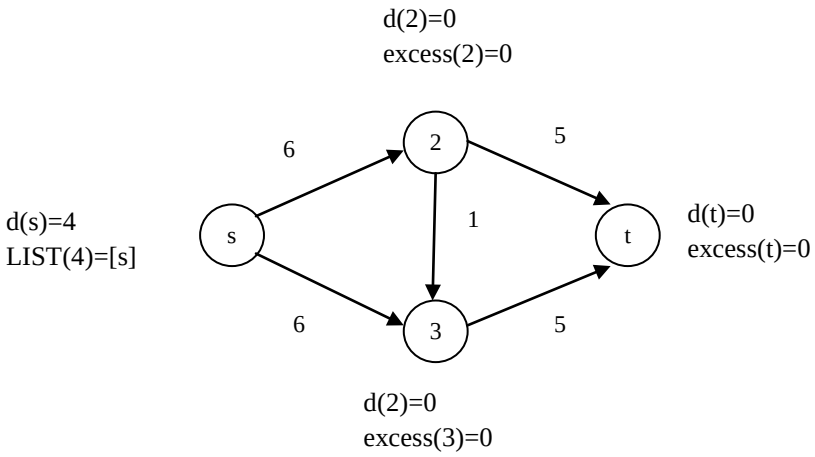
αναβάθμισε το d(i) = min{d(j)+1: (i,j) ανήκει στις ακμές του υπολειπόμενου γραφήματος και $r_{i,j>0}$}

πρόσθεσε τον κόμβο i στην λιστα LIST(d(i)) και θέσε για παρούσα ακμή του κόμβου i την πρώτη ακμή του κόμβου i που ανήκει στις ακμές του υπολειπόμενου γραφήματος

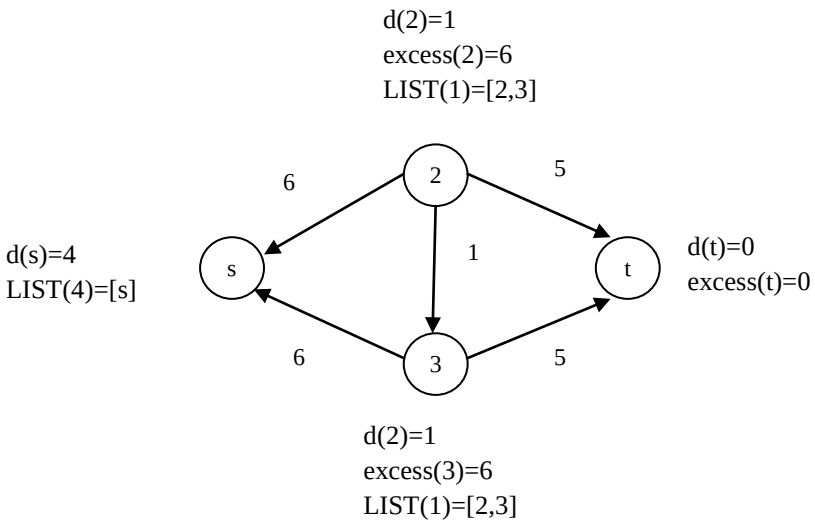
End

<u>Παράδειγμα:</u>

Δίκτυο Ροής G

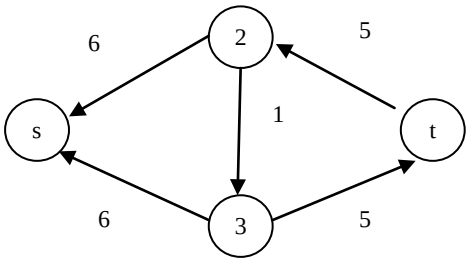


<u>Βήμα 1:</u>



Βήμα 2:

d(2)=1
excess(2)=1
LIST(1)=[3]



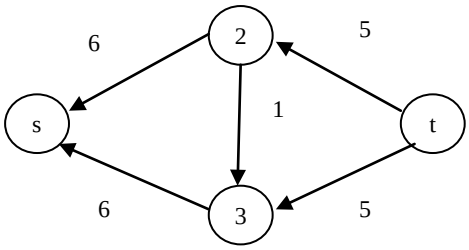
d(s)=4
LIST(4)=[s]

d(t)=0
excess(t)=5

d(2)=1
excess(3)=6
LIST(1)=[3]

Βήμα 3:

d(2)=1
excess(2)=1



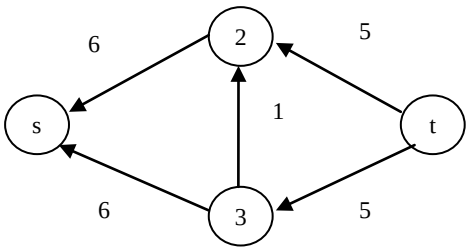
d(s)=4
LIST(4)=[s]

d(t)=0
excess(t)=10

d(2)=1
excess(3)=1

Βήμα 4:

d(2)=2
excess(2)=1
LIST(2)=[2]



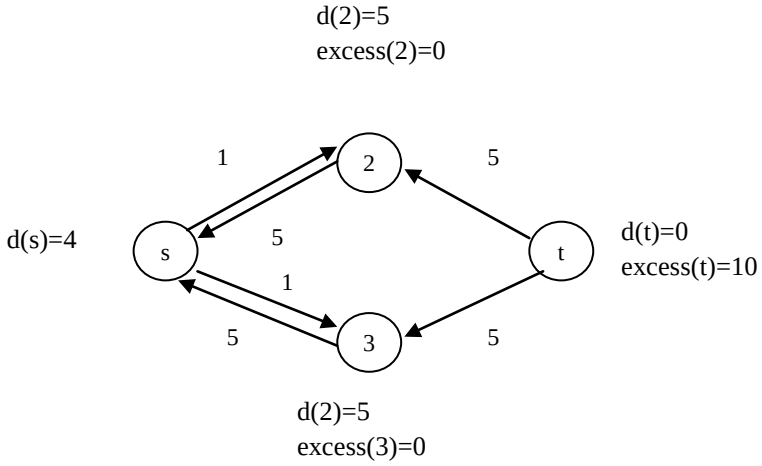
d(s)=4
LIST(4)=[s]

d(t)=0
excess(t)=10

d(2)=1
excess(3)=2

Βήμα 5:

μετά από πεπερασμένα βήματα(λιγότερα από αυτά που κάνει ο Preflow Push[1]) φτάνουμε στο εξής:



Ο αλγόριθμος Scaling Preflow Push τερματίζει όταν το Δ είναι μικρότερο του ενός, και η μέγιστη ροή σε αυτό το Δίκτυο Ροής ισούται με 10.

Θεωρητική Ανάλυση:

Ο θεωρητικός χρόνος που χρειάζεται ο αλγόριθμος Scaling Preflow Push[2] για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι

$O(n*m + n^2 \log U)$

όπου m = αριθμός ακμών του Δικτύου Ροής(|E|)

n = αριθμός κόμβων του Δικτύου Ροής (|V|)

U = μεγαλύτερη δύναμη του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο

Λεπτομέρειες Υλοποίησης:

Για την υλοποίηση αυτού του αλγορίθμου οι Δομές Δεδομένων που χρησιμοποιήθηκαν ήταν όσες λίστες είναι και το μεγαλύτερο ύψος που έχουμε επί δύο, για την αποθήκευση των κόμβων με πλεόνασμα όπως επίσης και πίνακες. Η υλοποίηση αυτού του αλγορίθμου δεν με δυσκόλεψε ιδιαίτερα μιας και ένα μεγάλο μέρος της ήταν βασισμένο στον αλγόριθμο Preflow Push[1]. Σε γενικές γραμμές η υλοποίηση μου βασίστηκε στον ψευδοκώδικα όπως επίσης και στις παρακινήσεις της ιστοσελίδας ώστε να γίνει μια αντικειμενική υλοποίηση.

Κεφάλαιο 5

Πειραματική αξιολόγηση

5.1 Πλατφόρμα Υλοποίησης και Μετρικές

Οι αλγόριθμοι όπως επίσης και όλες οι τοπολογίες στοχευόμενες και μη υλοποιήθηκαν σε γλώσσα προγραμματισμού Java. Για να γίνει όμως εφικτή η αξιολόγηση των αλγορίθμων που μελετήθηκαν , στα πλαίσια της Διπλωματικής Εργασίας, μου παραχωρήθηκε από την ομάδα του κ. Γιώργου Πάλλη χώρος στην μηχανή Nephelae. Το Nephelae είναι ένα cloud computing cluster που τρέχει Ubuntu Server 12.04 LTS όπως επίσης και την τελευταία έκδοση OpenStack . Αποτελείται από έξι servers και επί του παρόντος είναι διαμορφωμένο για χρήστες που αντιμετωπίζουν OpenStack Dashboard. Το Dashboard επικοινωνεί με ένα Nova Cloud Controller (σε ένα server),ενώ ένα άλλος back-end server τρέχει Nova Network Controller για έλεγχο του δικτύου. Οι υπόλοιποι servers που διαθέτει το Nephelae χρησιμοποιούνται για virtual machine φιλοξενία όπως ακριβώς έγινε και για την Διπλωματική Εργασία μου. Ο συνολικός αριθμός επεξεργαστών που διαθέτει το Nephelae είναι 92 με συνολικά 340 πυρήνες. Επίσης διαθέτει συνολικά 700 GB RAM και συνολικό αποθηκευτικό χώρο 87.2 TB. Οι φυσικοί πόροι που αποτελούν την Nephelae στεγάζονται στο τμήμα πληροφορικής του Πανεπιστημίου Κύπρου και διαχειρίζονται από το LiNC lab (http://linc.ucy.ac.cy/). Για τις ανάγκες των πειραμάτων μου, παραχωρήθηκε σε εμένα ένα εικονικό Linux περιβάλλον ούτως ώστε να γίνει ευκολότερη η αξιολόγηση των αλγορίθμων εκμεταλλευόμενοι την ταχύτητα και τις δυνατότητες του Nephelae. Συγκεκριμένα το Virtual Machine που μου παραχωρήθηκε είχε τα εξής χαρακτηριστικά: 8GB RAM,4 VCPU,80.0GB Disk. Το Virtual Machine αυτό τρέχει στην cloud υποδομή της Nephelae.

Η μέτρηση του χρόνου που χρειάζεται ο κάθε αλγόριθμος για να επιλύσει το πρόβλημα που του δόθηκε έγινε με τη χρήση της συνάρτησης System.currentTimeMillis() της βιβλιοθήκης java.lang

(Java) η οποία μετρά το χρόνο που χρειάζεται να τρέξει το πρόγραμμα σε Milliseconds (ms). Επίσης οι γραφικές παραστάσεις είναι όλες κατασκευασμένες σε λογαριθμική κλίμακα για να είναι πιο ξεκάθαρο το γράφημα και πιο προφανή τα αποτελέσματα.

5.2 Τοπολογίες

5.2.1 Στοχευόμενα Σενάρια

Παρακάτω περιγράφονται όλα τα στοχευόμενα σενάρια, δηλαδή όλες οι τοπολογίες οι οποίες σχεδιάστηκαν και υλοποιήθηκαν με κάποιο συγκεκριμένο σκοπό. Οι τοπολογίες αυτές σχεδιάστηκαν έχοντας σαν στόχο την αποδυνάμωση κάποιου αλγορίθμου, ή ακόμα και γενικά για να παρατηρηθεί η συμπεριφορά των αλγορίθμων βάση ενός συγκεκριμένου σεναρίου.

Τοπολογία 1

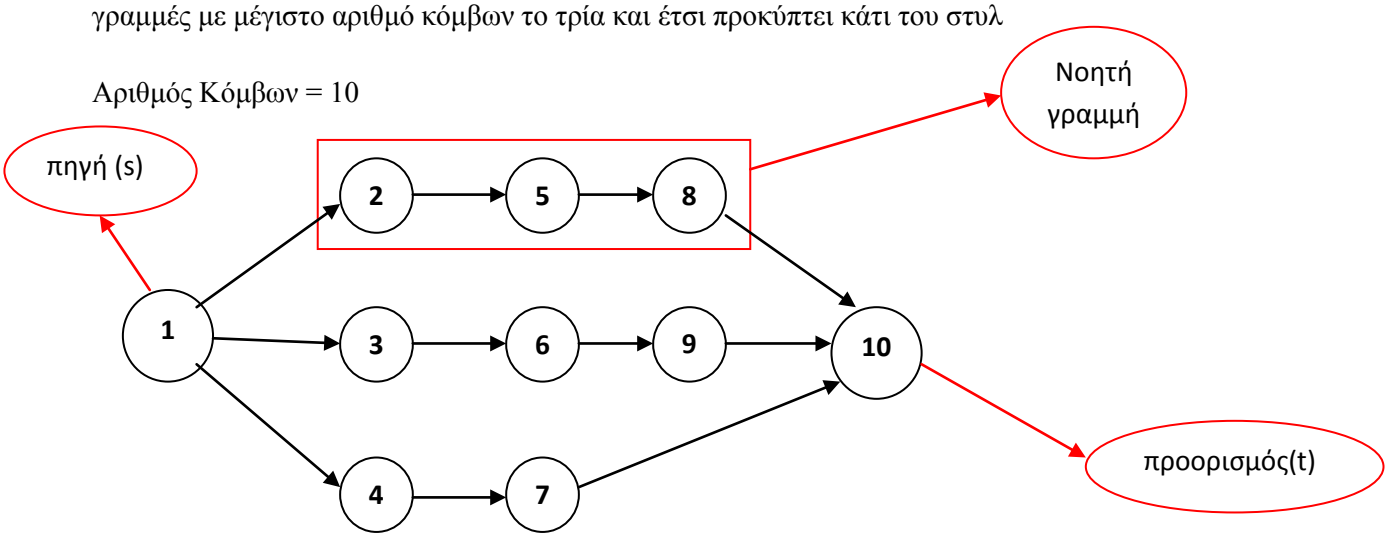
Στόχος Τοπολογίας:

Η τοπολογία αυτή είχε ως στόχο την αποδυνάμωση του αλγορίθμου Edmonds Karp[3]οπότε το σημείο που στόχευσε ήταν η τακτική επιλογής μονοπατιών που ακολουθεί για να υπολογίσει την μέγιστη ροή σε ένα γράφο.

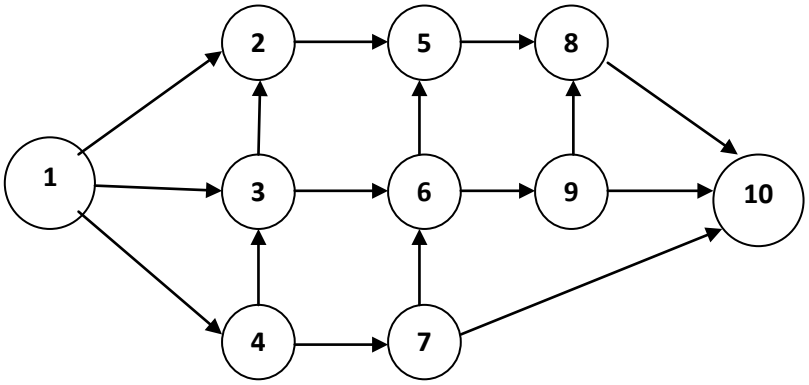
Περιγραφή Τοπολογίας:

Δημιουργούνται γράφοι οι οποίοι έχουν μια συγκεκριμένη μορφή και συγκεκριμένες χωρητικότητες έτσι ώστε το πιο μακρινό μονοπάτι που υπάρχει στο γράφο να είναι αυτό που θα μπορούσε να στείλει πιο σύντομα όλη την ροή που μπορεί να φτάσει από την πηγή στον προορισμό, έτσι ενώ ο αλγόριθμος συνεχώς θα επιλέγει τα πιο κοντινά μονοπάτια τα οποία όμως είναι και τα πιο στενά ,θα καθίσταται χειρότερος από τους υπόλοιπους αλγορίθμους. Για να είναι οι τοπολογίες που δημιουργούνται πολύ συγκεκριμένες έχουμε ορίσει ότι το ελάχιστο μονοπάτι από την πηγή στον προορισμό θα είναι δύο βήματα (s -> v , v -> t). Συγκεκριμένα ο τρόπος που δημιουργείται ένα δίκτυο, είναι να δίνεται σαν είσοδος ο αριθμός των κόμβων και αρχικά δημιουργούμαι νοητές γραμμές με μέγιστο αριθμό κόμβων το τρία και έτσι προκύπτει κάτι του στυλ

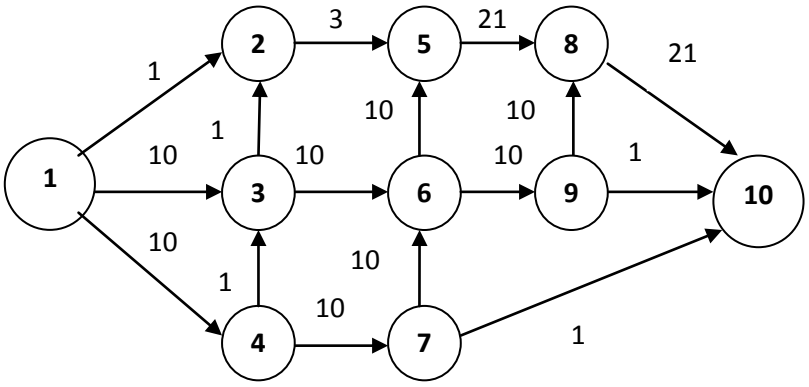
Αριθμός Κόμβων = 10



Στην συνέχεια για να δημιουργήσουμε μονοπάτια με πολλά βήματα, προσθέτουμε ακόμα ένα βήμα που δημιουργεί συνδέσεις από μεγαλύτερους σε μικρότερους κόμβους χωρίς όμως να δημιουργούμαι συνδέσεις από την πρώτη στην τελευταία νοητή γραμμή ή το αντίθετο. Ο λόγος που το κάνουμε αυτό είναι για να μην δημιουργούνται σύντομα μονοπάτια και να αναγκάζουμε έτσι τους αλγορίθμους να ακολουθήσουν τα μακρινά μονοπάτια για να φτάσουν στον προορισμό. Έτσι στην συνέχεια προκύπτει ένας γράφος του στυλ:



Τέλος τοποθετούμε τις εξής χωρητικότητες στον γράφο:



Χαρακτηριστικά Τοπολογίας:

Σε αυτή την τοπολογία όπως φαίνεται και στο παράδειγμα ο αριθμός της μέγιστης χωρητικότητας σε μια ακμή η οποία φεύγει από την πηγή και φτάνει σε ένα άλλο κόμβο (s -> v) είναι 10 και ο λόγος είναι για να μην επηρεαστεί ο χρόνος του Ford Fulkerson[1], διότι η παράμετρος αυτή παίζει σημαντικό ρόλο στον θεωρητικό του χρόνο και θα άλλαζε την ανάλυση μας σε βάρος του αλγορίθμου αυτού ενώ σε αυτή την τοπολογία στόχο έχουμε να δείξουμε την αδυναμία του Edmonds Karp[3]ως προς τον τρόπο επιλογής των μονοπατιών του.

Επίσης είναι γνωστός ο αριθμός των ακμών στην τοπολογία αυτή εφόσον έχουμε από όλους τους κόμβους ,ακμή σε δύο άλλους κόμβους δηλαδή δύο ακμές από τον καθένα, εκτός από αυτούς στην πρώτη νοητή γραμμή οι οποίοι έχουν μόνο μια ακμή προς άλλο κόμβο. Επιπρόσθετα έχουμε το κόμβο s (πηγή) ο οποίος έχει τόσες ακμές όσες είναι και οι νοητές γραμμές ενώ ο κόμβος t (προορισμός) δεν έχει καθόλου ακμές προς άλλους κόμβους.

Ακόμα για κάθε γράφο που δημιουργείται ο μέγιστος αριθμό νοητών γραμμών είναι τρία για οποιοδήποτε αριθμό κόμβων. Εμείς το ορίσαμε έτσι για να έχουμε πολύ συγκεκριμένες τοπολογίες. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 2

Στόχος Τοπολογίας:

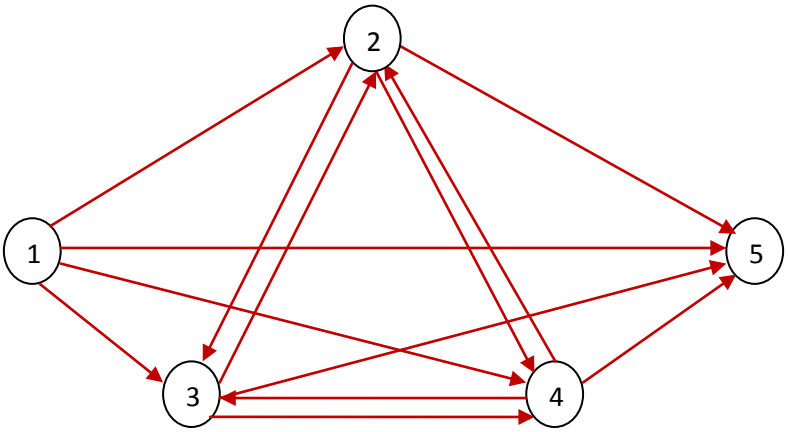
Η τοπολογία αυτή έχει ως στόχο την παρατήρηση του χρόνου που χρειάζονται οι αλγόριθμοι για να υπολογίσουν την μέγιστη ροή σε ένα γράφο όταν είναι πάρα πολύ πυκνός.

Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργεί αυτή η τοπολογία είναι πολύ πυκνοί, δηλαδή όλοι με όλους , εκτός από το τον κόμβο t που δεν δείχνει σε κανένα άλλο κόμβο και το κόμβο s (πηγή) στον οποίο δεν δείχνει κανένα όμως αυτός δείχνει σε όλους τους άλλους ακόμα και στον κόμβο t (προορισμός)

Οπότε δημιουργούνται γράφοι του στυλ:

Αριθμός Κόμβων = 5



Χαρακτηριστικά Τοπολογίας:

Κάθε ακμή στο γράφο έχει μια τυχαία χωρητικότητα από 1 μέχρι 10 και αυτό ορίστηκε έτσι για να μην επηρεαστεί κυρίως ο χρόνος του αλγορίθμου Ford Fulkerson[1]ο οποίος επηρεάζεται σε μεγάλο βαθμό από την παράμετρο αυτή. Επίσης όπως βλέπουμε, το συντομότερο μονοπάτι είναι ένα βήμα, απευθείας μονοπάτι από την πηγή στο προορισμό (s -> t). Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 3

Στόχος Τοπολογίας:

Στόχος της τοπολογίας αυτής είναι η αξιολόγηση του χρόνου που χρειάζονται οι αλγόριθμοι για να υπολογίσουν την μέγιστη ροή σε ένα γράφο, όταν οι ακμές από την πηγή σε οποιοδήποτε άλλο κόμβο έχουν πολύ μεγάλη χωρητικότητα ενώ οι υπόλοιπες ακμές του γράφου είναι πολύ στενές, έχουν δηλαδή πολύ μικρότερη χωρητικότητα σε σχέση με τις αυτές που έχουν οι ακμές που ξεκινούν από την πηγή.

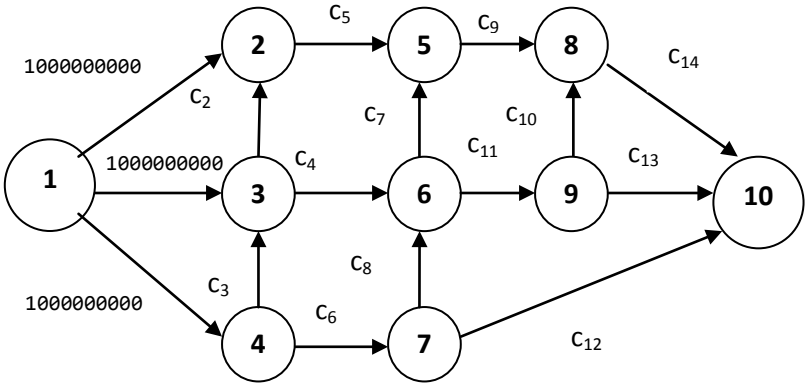
Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργούνται είναι τυχαίοι, με τυχαίες χωρητικότητες από το 0 μέχρι το 10 και στην συνέχεια σε όσες ακμές προέρχονται από την πηγή αλλάζουμε την χωρητικότητα τους σε 1000000000, για να υπάρχει μεγάλη διαφορά μεταξύ των χωρητικοτήτων των ακμών που προέρχονται από την πηγή και των υπόλοιπων ακμών του γράφου.

Οι γράφοι που προκύπτουν είναι τυχαίοι και είναι του στυλ :

Αριθμός Κόμβων = 10

->Πιθανός τυχαίος γράφος



Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι ακμές που ξεκινούν από την πηγή ως προς οποιοδήποτε κόμβο έχουν χωρητικότητα ίση με 1000000000 ενώ όλες οι υπόλοιπες ακμές έχουν τυχαία χωρητικότητα από 0 μέχρι 10. Επίσης στην κατασκευή της τοπολογίας υπάρχει ένα βήμα το οποίο εξασφαλίζει ότι ο τυχαίος γράφος που δημιουργείται είναι συνεκτικός (τουλάχιστον |V| -1 ακμές όπου |V|, ο αριθμός των κόμβων) και ακόμα ότι σίγουρα τουλάχιστον ένα μονοπάτι από την πηγή στον προορισμό. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 4

Στόχος Τοπολογίας:

Στόχος της τοπολογίας αυτής είναι να παρατηρήσουμε τον χρόνο που χρειάζονται οι αλγόριθμοι για να υπολογίσουν την μέγιστη ροή σε ένα γράφο όταν ο γράφος είναι πάρα πολύ αραιός και οι ακμές που προέρχονται από την πηγή έχουν πολύ μεγάλη χωρητικότητα.

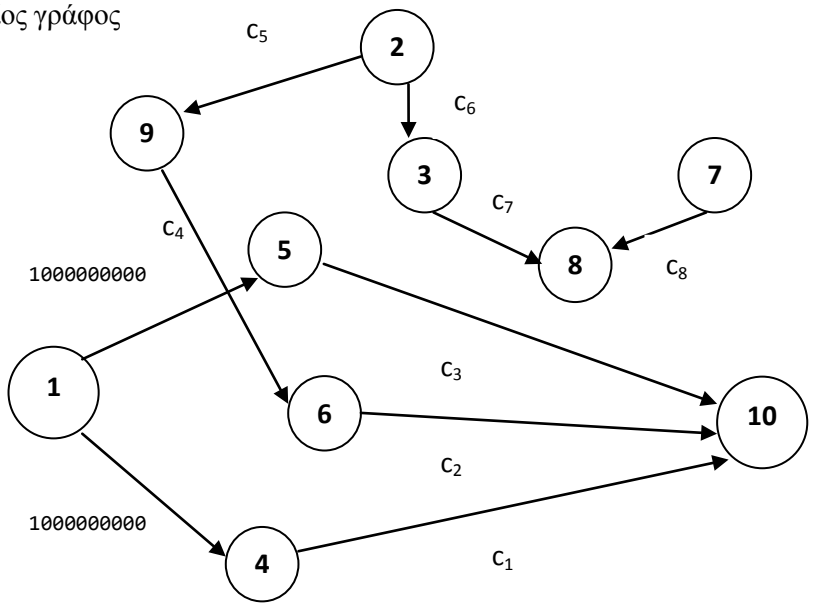
Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργούνται είναι τυχαίοι και πολύ αραιοί, με τυχαίες χωρητικότητες από το 0 μέχρι το 10 και στην συνέχεια σε όσες ακμές προέρχονται από την πηγή αλλάζουμε την χωρητικότητα τους σε 1000000000 ,για να υπάρχει μεγάλη διαφορά μεταξύ των χωρητικοτήτων των ακμών που προέρχονται από την πηγή και των υπόλοιπων ακμών του γράφου και ο γράφος να είναι πολύ αραιός.

Οι γράφοι που προκύπτουν είναι τυχαίοι και είναι του στυλ :

Αριθμός Κόμβων = 10

->Πιθανός τυχαίος γράφος



Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι ακμές που ξεκινούν από την πηγή ως προς οποιοδήποτε κόμβο έχουν χωρητικότητα ίση με 1000000000 ενώ όλες οι υπόλοιπες ακμές έχουν τυχαία χωρητικότητα από 0 μέχρι 10. Επίσης στην κατασκευή της τοπολογίας εξασφαλίζεται ότι ο τυχαίος γράφος που δημιουργείται είναι συνεκτικός (τουλάχιστον |V| -1 ακμές όπου |V|, ο αριθμός των κόμβων), αραιός και ακόμα ότι υπάρχει σίγουρα τουλάχιστον ένα μονοπάτι από την πηγή στον προορισμό. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 5

Στόχος Τοπολογίας:

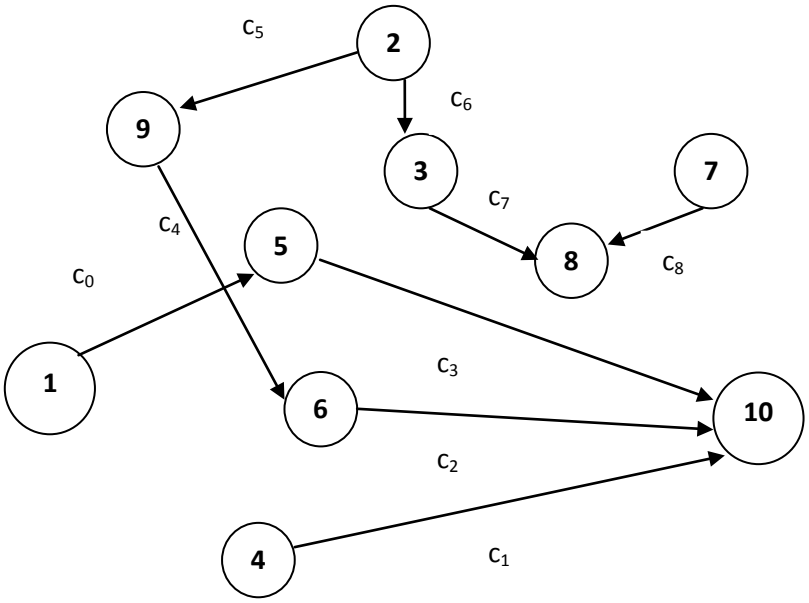
Στόχος της τοπολογίας αυτής είναι να παρατηρήσουμε τον χρόνο που χρειάζονται οι αλγόριθμοι για να υπολογίσουν την μέγιστη ροή σε ένα γράφο όταν ο γράφος είναι πάρα πολύ αραιός και οι χωρητικότητες των ακμών είναι πολύ μικρές.

Περιγραφή Τοπολογίας:

Δημιουργούνται γράφοι οι οποίοι είναι πολύ αραιοί και οι χωρητικότητες που δίνονται στις ακμές δεν ξεπερνούν το 10.

Οπότε προκύπτουν γράφοι του στυλ:

Αριθμός Κόμβων = 10



Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι χωρητικότητες των ακμών κυμαίνονται από 1 μέχρι 10 επειδή αυτό που θέλουμε είναι να ελέγξουμε την απόδοση των αλγορίθμων ως προς το χρόνο που χρειάζονται για να επιλύσουν το πρόβλημα της μέγιστης ροής, όταν ο γράφος είναι πολύ αραιός και δεν θέλουμε να επηρεαστούν οι χρόνοι τους λόγω των χωρητικοτήτων οπότε τις κρατάμε πολύ μικρές. Επίσης εξασφαλίζουμε ότι ο γράφος είναι αρκετά αραιός, συνεκτικός (τουλάχιστον |V| -1 ακμές όπου |V|, ο αριθμός των κόμβων) και επίσης ότι υπάρχει σίγουρα ένα μονοπάτι από την πηγή στον προορισμό. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 6

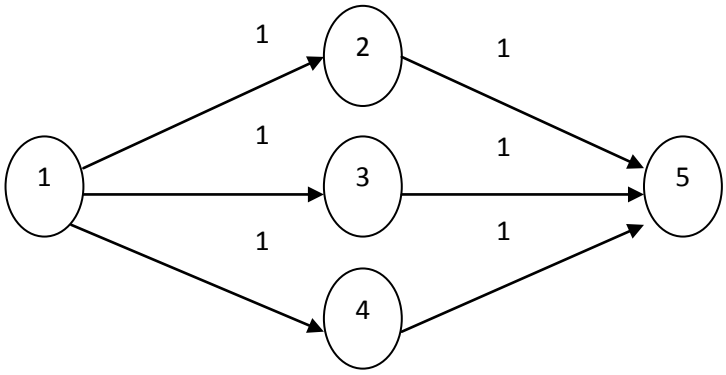
Στόχος Τοπολογίας:

Η τοπολογία αυτή έχει ως στόχο την παρατήρηση του χρόνου που χρειάζονται οι αλγόριθμοι όταν ο κόμβος s (πηγή) στέλνει μια πολύ μικρή χωρητικότητα σε όλους τους άλλους κόμβους, εκτός από το κόμβο t (προορισμός) και αυτοί με τη σειρά τους στέλνουν την ίδια μικρή χωρητικότητα που έχουν παραλάβει απευθείας στον κόμβο t (προορισμός).

Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργούνται έχουν μονοπάτια τα οποία είναι ακριβώς δύο βήματα από την πηγή στον προορισμό. Οι ακμές στους γράφους που προκύπτουν έχουν χωρητικότητα ίση με 1 και είναι του στυλ:

Αριθμός Κόμβων = 5



Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι χωρητικότητες των ακμών είναι ίσες με 1 και επίσης γνωρίζουμε ακριβώς τον αριθμό των ακμών. Στον κάθε κόμβο αντιστοιχούν δύο ακμές, η μια τον ενώνει με την πηγή και η άλλη με τον προορισμό, εκτός από τον κόμβο s (πηγή) και τον κόμβο t (προορισμός) στους οποίους δεν αντιστοιχεί καμία ακμή. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 7

Στόχος Τοπολογίας:

Η τοπολογία αυτή έχει ως στόχο την μελέτη του χρόνου που χρειάζονται οι αλγόριθμοι για να επιλύσουν το πρόβλημα της μέγιστης ροής όταν ο γράφος είναι τυχαίος όμως ο χωρητικότητες των ακμών έχουν μεγάλο εύρος τιμών, όταν δηλαδή παίρνουν και μεγάλες αλλά και μικρές τιμές και επίσης όταν ο γράφος είναι αρκετά πυκνός. Σκοπός είναι να παρατηρήσουμε τους αλγορίθμους που ο θεωρητικός τους χρόνος αλλά και η τεχνική που χρησιμοποιούν επηρεάζεται από τις χωρητικότητες των ακμών.

Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργούνται είναι τυχαίοι, αρκετά πυκνοί και ακόμα έχουν τυχαίες χωρητικότητες που κυμαίνονται όμως από 1 μέχρι 100000. Δηλαδή κάποιες ακμές είναι πολύ στενές ενώ κάποιες άλλες να έχουν πολύ μεγάλη χωρητικότητα.

Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι τιμές των χωρητικοτήτων κυμαίνονται από το 1 μέχρι το 100000. Επίσης οι γράφοι που δημιουργούνται κυρίως για μικρό αριθμό κόμβων, είναι αρκετά πυκνοί , στην συνέχεια όμως για μεγάλο αριθμό κόμβων δεν είναι τόσο πυκνοί , αλλά συνεχίζουν να είναι πυκνοί. Επιπρόσθετα εξασφαλίζεται ότι ο γράφος είναι συνεκτικός (τουλάχιστον |V| -1 ακμές όπου |V|, ο αριθμός των κόμβων) και ότι υπάρχει σίγουρα τουλάχιστον ένα μονοπάτι από την πηγή στον προορισμό. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 8

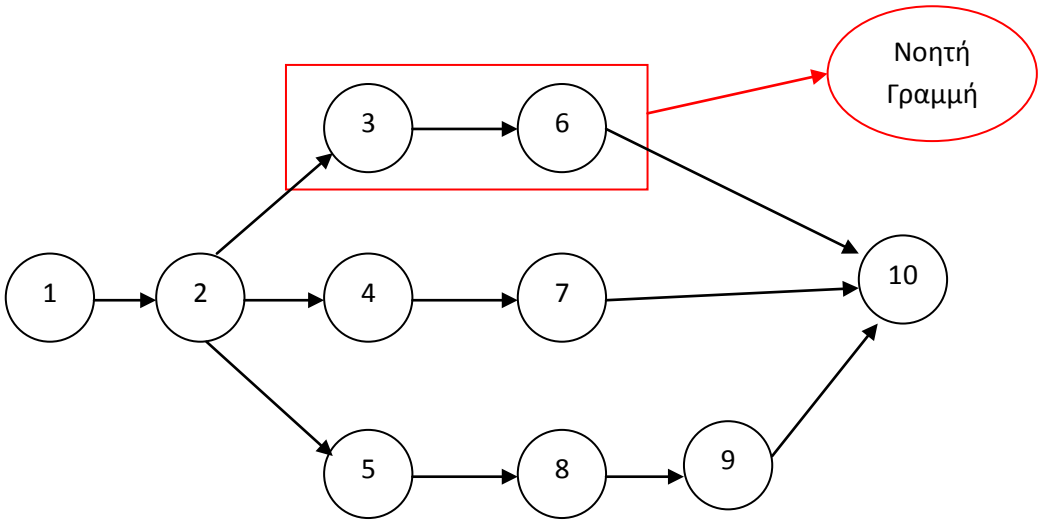
Στόχος Τοπολογίας:

Η τοπολογία αυτή είχε ως στόχο την αποδυνάμωση του αλγορίθμου Ford Fulkerson[1]οπότε το σημείο που στόχευσε ήταν η τακτική επιλογής μονοπατιών που ακολουθεί για να υπολογίσει την μέγιστη ροή σε ένα γράφο.

Περιγραφή Τοπολογίας:

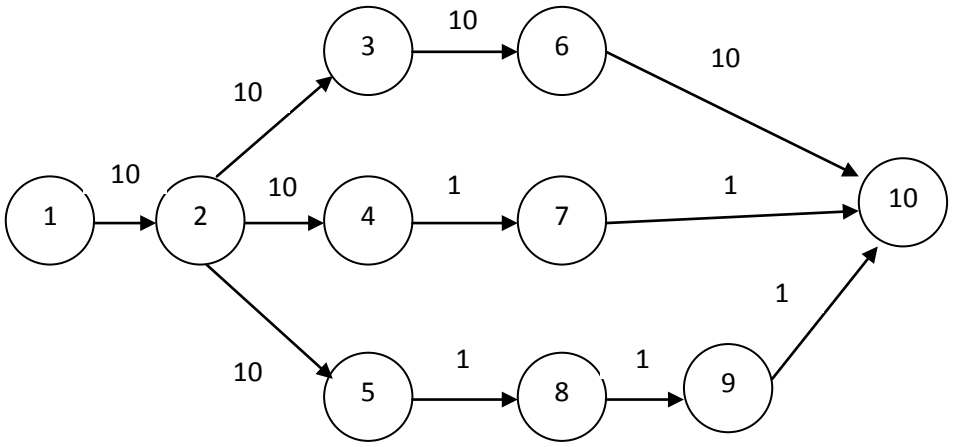
Δημιουργούνται γράφοι οι οποίοι έχουν μια συγκεκριμένη μορφή και συγκεκριμένες χωρητικότητες έτσι ώστε μονοπάτι που θα επιλέξει τελευταίο ο αλγόριθμος Ford Fulkerson, να είναι αυτό που θα μπορούσε να στείλει όλη την ροή που μπορεί να φτάσει από την πηγή στον προορισμό , έτσι ενώ ο αλγόριθμος συνεχώς θα επιλέγει τα μονοπάτια που είναι τα πιο στενά ,θα καθίσταται χειρότερος από τους άλλους αλγορίθμους. Συγκεκριμένα δίνεται, ο αριθμός των κόμβων στο πρόγραμμα, δύο από τους οποίους αποτελούν την πηγή και τον προορισμό (s , t) . Για να είναι οι τοπολογίες που δημιουργούνται πολύ συγκεκριμένες έχουμε ορίσει ότι το ελάχιστο μονοπάτι από την πηγή στον προορισμό θα είναι τρία βήματα (s ->v, v ->u , u ->t). Συγκεκριμένα ο τρόπος που δημιουργείται ένα δίκτυο, είναι να δίνεται σαν είσοδος ο αριθμός των κόμβων και αρχικά δημιουργούμαι νοητές γραμμές με μέγιστο αριθμό κόμβων το τρία και έτσι προκύπτει κάτι του στυλ

Αριθμός Κόμβων = 10



Δεν τοποθετούμε άλλες ακμές γιατί θέλουμε να κρατήσουμε τον συνολικό αριθμό των ακμών μικρό ώστε ο χρόνος του Ford Fulkerson[1]να μην επηρεαστεί καθόλου από αυτή την παράμετρο.

Τέλος τοποθετούμε τις εξής χωρητικότητες στον γράφο:



Χαρακτηριστικά Τοπολογίας:

Σε αυτή την τοπολογία όπως φαίνεται και στο παράδειγμα ο αριθμός της μέγιστης χωρητικότητας σε μια ακμή η οποία φεύγει από οποιοδήποτε κόμβο και φτάνει σε κάποιο άλλο κόμβο είναι 10 και ο λόγος είναι για να μην επηρεαστεί ο χρόνος του Ford Fulkerson, διότι η παράμετρος αυτή παίζει σημαντικό ρόλο στον θεωρητικό του χρόνο και θα άλλαζε την ανάλυση μας σε βάρος του αλγορίθμου αυτού ενώ σε αυτή την τοπολογία στόχο έχουμε να δείξουμε την αδυναμία του Ford Fulkerson[1]ως προς τον τρόπο επιλογής των μονοπατιών του.

Επίσης είναι γνωστός ο αριθμός των ακμών στην τοπολογία αυτή εφόσον έχουμε από όλους τους κόμβους ,ακμή σε ένα άλλο κόμβο δηλαδή μια ακμή από τον καθένα, εκτός από τον δεύτερο κόμβο ο οποίος έχει τόσες ακμές προς άλλους κόμβους, όσες και οι νοητές γραμμές. Επιπρόσθετα τον κόμβο t (προορισμός) ο οποίος δεν έχει καθόλου ακμές προς άλλους κόμβους. Ακόμα κρατήσαμε το γράφο σχετικά αραιό γιατί θέλαμε ο αριθμός των ακμών να παραμείνει μικρός ώστε να μην επηρεαστεί ο χρόνος του αλγορίθμου Ford Fulkerson[1]από αυτή την παράμετρο.Όπως φαίνεται και στο σχήμα ο κόμβος s (πηγή) στέλλει μόνο σε ένα κόμβο για οποιοδήποτε αριθμό κόμβων. Αυτό ορίστηκε έτσι ώστε το συνολικό άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή και καταλήγουν σε οποιοδήποτε άλλο κόμβο (C) να παραμένει μικρό και μην έχει αντίκτυπο στον χρόνο του αλγορίθμου Ford Fulkerson[1]που είναι άμεσα επηρεαζόμενος από την παράμετρο αυτή. Ακόμα για κάθε γράφο που δημιουργείται ο μέγιστο αριθμό νοητών γραμμών είναι τρία, για οποιοδήποτε αριθμό κόμβων. Εμείς το ορίσαμε έτσι για να έχουμε πολύ συγκεκριμένες τοπολογίες. Επίσης εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

Τοπολογία 9

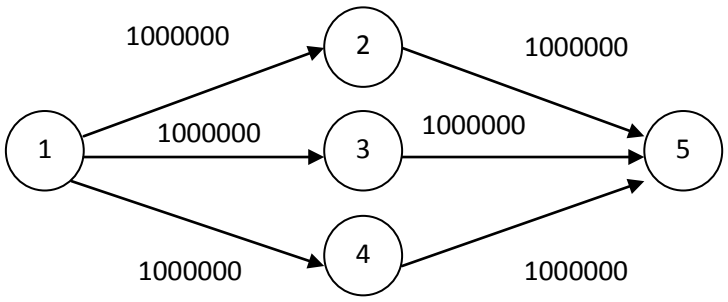
Στόχος Τοπολογίας:

Η τοπολογία αυτή έχει ως στόχο την παρατήρηση του χρόνου που χρειάζονται οι αλγόριθμοι όταν ο κόμβος s (πηγή) στέλνει μια πολύ μεγάλη χωρητικότητα σε όλους τους άλλους κόμβους, εκτός από το κόμβο t (προορισμός) και αυτοί με τη σειρά τους στέλνουν την ίδια μεγάλη χωρητικότητα που έχουν παραλάβει απευθείας στον κόμβο t (προορισμός).

Περιγραφή Τοπολογίας:

Οι γράφοι που δημιουργούνται έχουν μονοπάτια τα οποία είναι ακριβώς δύο βήματα από την πηγή στον προορισμό. Οι ακμές στους γράφους που προκύπτουν έχουν χωρητικότητα ίση με 1000000 και είναι του στυλ:

Αριθμός Κόμβων = 5



Χαρακτηριστικά Τοπολογίας:

Στην τοπολογία αυτή οι χωρητικότητες των ακμών είναι ίσες με 1000000 και επίσης γνωρίζουμε ακριβώς τον αριθμό των ακμών. Στον κάθε κόμβο αντιστοιχούν δύο ακμές, η μια τον ενώνει με την πηγή και η άλλη με τον προορισμό, εκτός από τον κόμβο s (πηγή) και τον κόμβο t (προορισμός) στους οποίους δεν αντιστοιχεί καμία ακμή. Ακόμα εξασφαλίσαμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των γράφων φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

5.2.2 Μη Στοχευόμενα Σενάρια

Στόχος:

Η δημιουργία τυχαίων σεναρίων έχουν ως στόχο την εξαγωγή συμπερασμάτων, όσον αφορά τον χρόνο που χρειάζονται οι αλγόριθμοι που μελετήθηκαν στην διπλωματική εργασία, για να υπολογίσουν την μέγιστη ροή που μπορεί να διοχετευθεί μέσα από ένα γράφο ξεκινώντας από τον πρώτο κόμβο (πηγή)και φτάνοντας στον τελευταίο (προορισμός)όταν η τοπολογία είναι τυχαία. Η συμπεριφορά των τυχαίων αυτών σεναρίων παρατηρήθηκε δίνοντας τους σαν είσοδο μικρό αλλά και μεγάλο αριθμό κόμβων χωρίς όμως να αλλάζει το εύρος των τιμών των χωρητικοτήτων. Αυτό έγινε για να είναι αντικειμενική η αξιολόγηση των αλγορίθμων, αλλά και για να διαφανούν οι ικανότητες του καθενός ξεχωριστά σε σχέση με τους υπόλοιπους. Λόγω του ότι ο κάθε αλγόριθμος δουλεύει με μια συγκεκριμένη φιλοσοφία είναι δύσκολο με μικρούς γράφους να αντιληφθούμε ξεκάθαρα τις δυνατότητες του καθενός ξεχωριστά.

Χαρακτηριστικά:

Οι χωρητικότητες των ακμών των γράφων που δημιουργούνται αρχικοποιούνται με τιμές από το 1 μέχρι το 10, λόγω του ότι ο χρόνος κάποιων αλγορίθμων είναι άμεσα επηρεαζόμενος από την παράμετρο αυτή. Ο αριθμός των ακμών είναι άγνωστος όμως γνωρίζουμε ότι οι γράφοι που δημιουργούνται είναι αρκετά πυκνοί. Επίσης για κάθε τυχαία τοπολογία που δημιουργείται εξασφαλίζεται ότι ο γράφος είναι συνεκτικός και ακόμα ότι υπάρχει σίγουρα ένα μονοπάτι από την πηγή στον προορισμό. Ακόμα εξασφαλίζουμε ότι κανένας κόμβος δεν έχει ακμή που να καταλήγει απευθείας στον εαυτό του. Επιπρόσθετα στην κατασκευή των τυχαίων τοπολογιών φροντίσαμε έτσι ώστε κανένας κόμβος αρχικά να μην δείχνει στην πηγή και ακόμα ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο.

5.3 Χαρακτηριστικά Υλοποίησης

5.3.1 Στοχευόμενα Σενάρια

Η κάθε στοχευόμενη τοπολογία που δημιουργείτο, αποθηκευόταν σε ένα αρχείο και στην συνέχεια ο κάθε αλγόριθμος ξεχωριστά, καλώντας μια συνάρτηση διάβαζε την τοπολογία την οποία έπρεπε να επεξεργαστεί. Στον συνολικό χρόνο που χρειαζόταν ο κάθε αλγόριθμος για την κάθε τοπολογία δεν συμπεριλαμβανόταν ο χρόνος που χρειαζόταν για να διαβάσει τα δεδομένα από το αρχείο και να γεμίσει τον πίνακα των χωρητικοτήτων. Ακόμα σημαντικό είναι να αναφερθεί ότι για κάθε αριθμό κόμβων που δινόταν στους αλγορίθμους, έτρεχαν και οι πέντε από δώδεκα φορές ο καθένας, ώστε να υπολογίζεται στο τέλος ένας μέσος όρος που να είναι αντιπροσωπευτικός για το χρόνο που χρειάζεται ο κάθε ένας για το συγκεκριμένο αριθμό κόμβων. Επίσης για να είναι οι γραφικές παραστάσεις ομοιόμορφες, έγινε χρήση μιας τεχνικής όπου από τις δώδεκα συνολικά τιμές που είχαμε για τον κάθε ένα από τους αλγορίθμους αφαιρείτο η πιο μεγάλη και η πιο μικρή τιμή που

είχαν παρουσιάσει . Οπότε αν κάποιος αλγόριθμος σε κάποια από τις δώδεκα φορές που έτρεξε για τον συγκεκριμένο αριθμό κόμβων παρουσίασε κάποια πολύ μεγάλη τιμή ή κάποια πολύ μικρή τιμή μόνο μια φορά, αυτό να μην είχε ως αποτέλεσμα να επηρεαστεί ο μέσος όρος του σε σημείο που να μην είναι ενδεικτικός. Σε κάθε σενάριο ισχύει ότι κανένας κόμβος δεν δείχνει αρχικά στην πηγή και επίσης ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο. Επίσης σε κάθε τοπολογία ο κόμβος πηγή είναι ο πρώτος κόμβος και ο κόμβος προορισμός ισούται με τον αριθμό των κόμβων. Τρέξαμε κάθε σενάριο για δέκα , πενήντα , εκατό , πεντακόσιους και χίλιους κόμβους για να παρουσιάσουμε έτσι την απόδοση των αλγορίθμων για κάθε σενάριο και για μεγάλους αλλά και για μικρούς γράφους λόγω του ότι ο χρόνος κάποιων αλγορίθμων επηρεάζεται από τον αριθμό των κόμβων του γράφου . Στις γραφικές παραστάσεις παρουσιάζουμε και την θεωρητική ανάλυση και την πραγματική ανάλυση του κάθε αλγορίθμου για να μπορούμε να συγκρίνουμε τον θεωρητικό με τον πραγματικό χρόνο του αλγορίθμου. Όλοι οι γράφοι είναι αποθηκευμένοι σε πίνακες γειτνίασης λόγω του ότι οι περισσότερες από τις τοπολογίες που σχεδιάστηκαν δημιουργούν πολύ πυκνούς γράφους και είναι πιο αποδοτική η χρήση πινάκων. Οι αλγόριθμοι τρέχουν ο ένας μετά τον άλλο με τη χρήση ενός batch (.sh) αρχείου ώστε να ελευθερώνει ο αλγόριθμος που τρέχει τους πόρους πριν αρχίσει να τρέχει ο επόμενος για να είναι αντικειμενική η καταμέτρηση του χρόνου.

5.3.1 Μη Στοχευόμενα Σενάρια

Η κάθε τυχαία τοπολογία που δημιουργείτο, αποθηκευόταν σε ένα αρχείο και στην συνέχεια ο κάθε αλγόριθμος ξεχωριστά, καλώντας μια συνάρτηση διάβαζε την τοπολογία την οποία έπρεπε να επεξεργαστεί. Στον συνολικό χρόνο που χρειαζόταν ο κάθε αλγόριθμος για την κάθε τοπολογία δεν συμπεριλαμβανόταν ο χρόνος που χρειαζόταν για να διαβάσει τα δεδομένα από το αρχείο και να γεμίσει τον πίνακα των χωρητικοτήτων. Ακόμα σημαντικό είναι να αναφερθεί ότι για κάθε αριθμό κόμβων που δινόταν στους αλγορίθμους , έτρεχαν και οι πέντε από δώδεκα(12) φορές ο καθένας, ώστε να υπολογίζεται στο τέλος ένας μέσος όρος που να είναι αντιπροσωπευτικός για το χρόνο που χρειάζεται ο κάθε ένας για το συγκεκριμένο αριθμό κόμβων. Επίσης για να είναι οι γραφικές παραστάσεις πιο ομαλές , έγινε χρήση μιας τεχνικής όπου από τις δώδεκα συνολικά τιμές που είχαμε για τον κάθε ένα από τους αλγορίθμους αφαιρείτο η πιο μεγάλη και η πιο μικρή τιμή που είχαν παρουσιάσει . Οπότε αν κάποιος αλγόριθμος σε κάποια από τις δώδεκα φορές που έτρεξε για τον συγκεκριμένο αριθμό κόμβων παρουσίασε κάποια πολύ μεγάλη τιμή ή κάποια πολύ μικρή τιμή μόνο μια φορά, αυτό να μην είχε ως αποτέλεσμα να επηρεαστεί ο μέσος όρος του σε σημείο που να μην είναι ενδεικτικός. Αν δηλαδή ένας αλγόριθμος σε κάποια από τις τυχαίες τοπολογίες υστερούσε κατά πολύ με αποτέλεσμα αυτό να επηρέαζε τον μέσο όρο του σε υπερβολικό βαθμό η πιο πάνω τεχνική το διόρθωσε. Σε κάθε σενάριο ισχύει ότι κανένας κόμβος δεν δείχνει αρχικά στην πηγή και επίσης ότι ο κόμβος προορισμός δεν δείχνει αρχικά σε κανένα άλλο κόμβο. Επίσης σε κάθε τοπολογία ο κόμβος πηγή είναι ο πρώτος κόμβος και ο κόμβος προορισμός ισούται με τον αριθμό των κόμβων. Τρέξαμε κάθε σενάριο για δέκα, εικοσιπέντε, πενήντα, εκατό , διακόσιους πενήντα , πεντακόσιους , χίλιους,

δυόμισι χιλιάδες και πέντε χιλιάδες κόμβους για να παρουσιάσουμε έτσι την απόδοση των αλγορίθμων για κάθε σενάριο και για μεγάλους αλλά και για μικρούς γράφους λόγω του ότι ο χρόνος κάποιων αλγορίθμων επηρεάζεται από τον αριθμό των κόμβων του γράφου . Στη συγκριτική γραφική παράσταση παρουσιάζουμε μόνο την πραγματική ανάλυση του κάθε αλγορίθμου διότι εφόσον οι τοπολογίες είναι εντελώς τυχαίες θα ήταν ανούσιο να δημιουργούσαμε την θεωρητική ανάλυση βάση των αποτελεσμάτων. Όλοι οι γράφοι είναι αποθηκευμένοι σε πίνακες γειτνίασης διότι οι γράφοι που δημιουργούνται είναι σχετικά πολύ πυκνοί. Οι αλγόριθμοι τρέχουν ο ένας μετά τον άλλο με τη χρήση ενός batch (.sh) αρχείου ώστε να ελευθερώνει ο αλγόριθμος που τρέχει τους πόρους πριν αρχίσει να τρέχει ο επόμενος για να είναι αντικειμενική η καταμέτρηση του χρόνου.

5.4 Αποτελέσματα

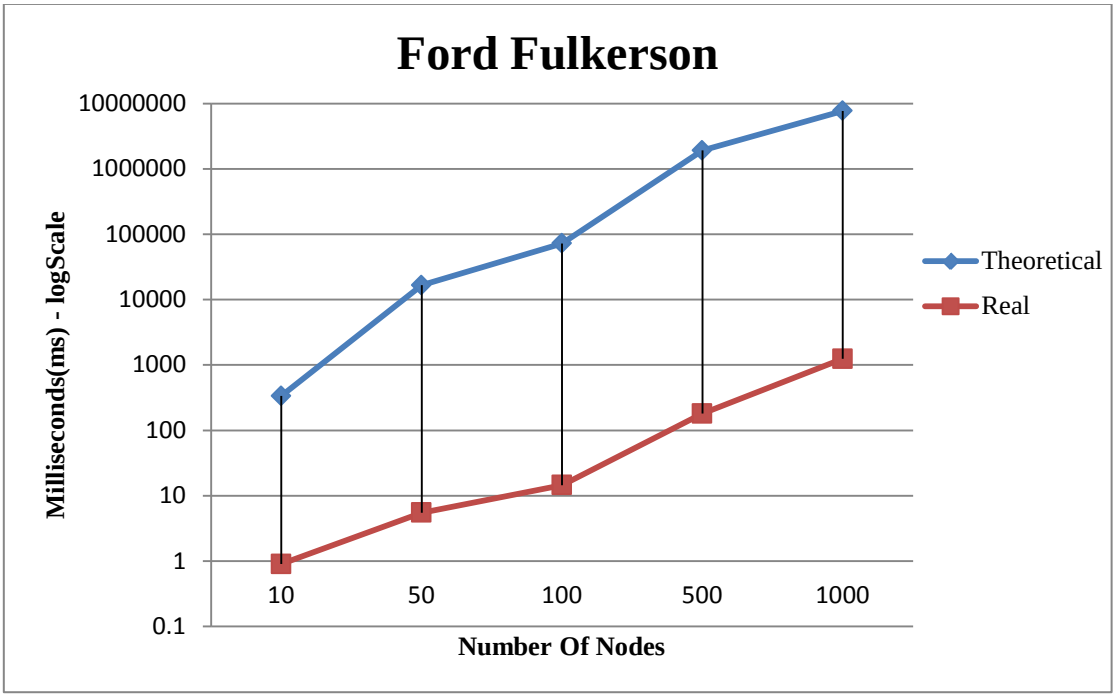
Παρακάτω φαίνονται όλα τα αποτελέσματα που προέκυψαν από την πειραματική αξιολόγηση των αλγορίθμων όσον αφορά τα στοχευόμενα αλλά και τα τυχαία σενάρια. Αρχικά παρουσιάζονται οι γραφικές παραστάσεις που συγκρίνουν την πραγματική συμπεριφορά του αλγόριθμου με την θεωρητική του και στην συνέχεια παρουσιάζονται οι συγκριτικές γραφικές παραστάσεις.

5.4.1.1 Αλγόριθμος Ford Fulkerson

Τοπολογία 1

Στην πρώτη τοπολογία στόχος είναι η αποδυνάμωση του αλγορίθμου Edmonds Karp[3]και στον τρόπο που ο αλγόριθμος αυτός διαλέγει το μονοπάτι που θα επεξεργαστεί, οπότε αναμένουμε ότι ο αλγόριθμος Ford Fulkerson[1]θα έχει ακριβώς τη ίδια συμπεριφορά με την θεωρητική του ανάλυση για αυτό το σενάριο διότι ο χρόνος του δεν επηρεάζεται άμεσα. Αυτό που αναμένουμε είναι ο χρόνος του αλγορίθμου να μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων αφού με βάση την τοπολογία έτσι αυξάνεται και ο αριθμός των ακμών. Επίσης ο χρόνος του Ford Fulkerson[1]επηρεάζεται από το άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή προς οποιοδήποτε άλλο κόμβο. Σε αυτή την τοπολογία το άθροισμα αυτών των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή προς οποιοδήποτε άλλο κόμβο διατηρείται σχετικά μικρό και αυξάνεται σε σχέση με τον αριθμό των κόμβων. Επίσης η μέγιστη χωρητικότητα που μπορεί να έχει μια ακμή είναι δέκα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 1.

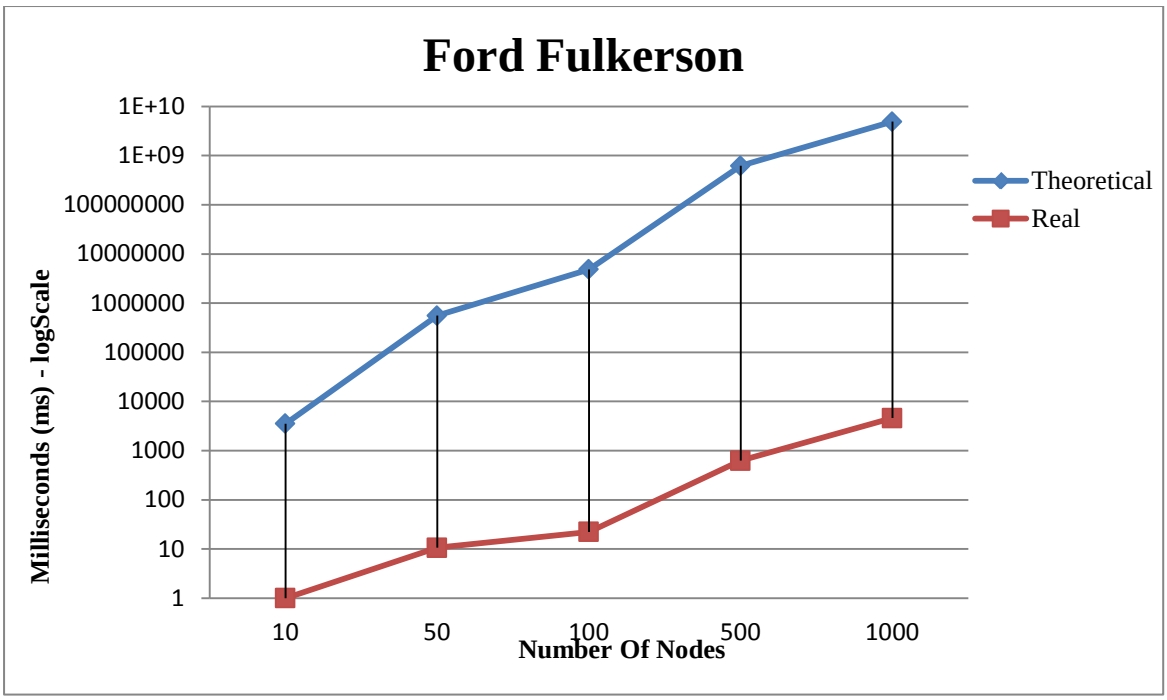


Όπως αναμέναμε η ανάλυση του πραγματικού χρόνου που χρειάζεται ο αλγόριθμος Ford Fulkerson[1]για να επιλύσει το πρόβλημα της Εύρεσης Μέγιστης Ροής σε ένα γράφο που δημιουργείται με βάση την Τοπολογία 1 έχει την ίδια συμπεριφορά με την ανάλυση του θεωρητικού χρόνου του αλγορίθμου με βάση των αριθμό των ακμών που είναι σταθερός και εξαρτάται από τον αριθμό των κόμβων όπως επίσης και από το άθροισμα των χωρητικοτήτων από την πηγή σε οποιοδήποτε άλλο κόμβο.

Τοπολογία 2

Η τοπολογία αυτή δημιουργεί πολύ πυκνούς γράφους, διότι όλοι οι κόμβοι συνδέονται απευθείας με όλους του υπόλοιπους κόμβους οπότε κατά συνέπεια ο αριθμός των ακμών είναι πάρα πολύ μεγάλος και αυτή είναι μια παράμετρος που επηρεάζει τον αλγόριθμό με βάση την θεωρητικό του χρόνο. Αντίθετα από τον αριθμό των ακμών, το άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή προς όλους του άλλους κόμβους παραμένει σχετικά μικρό διότι η μέγιστη χωρητικότητα που μπορεί να πάρει μια ακμή σε αυτή την τοπολογία είναι το δέκα, άρα το συνολικό άθροισμα αυξάνεται σε σχέση με τον αριθμό των κόμβων, όμως όχι ραγδαία. Οπότε αυτό που περιμένουμε να δούμε είναι το γράφημα του πραγματικού χρόνου να αυξάνεται σε σχέση με τον αριθμό των κόμβων και επίσης περιμένουμε ότι θα μοιάζει με το γράφημα του θεωρητικού χρόνου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 2



Όπως αναμέναμε , τα δύο γραφήματα που προέκυψαν είναι πολύ όμοια με το γράφημα του πραγματικού χρόνου να γίνεται ακόμα καλύτερο από το γράφημα του θεωρητικού όσο μεγαλώνει ο αριθμός των κόμβων.

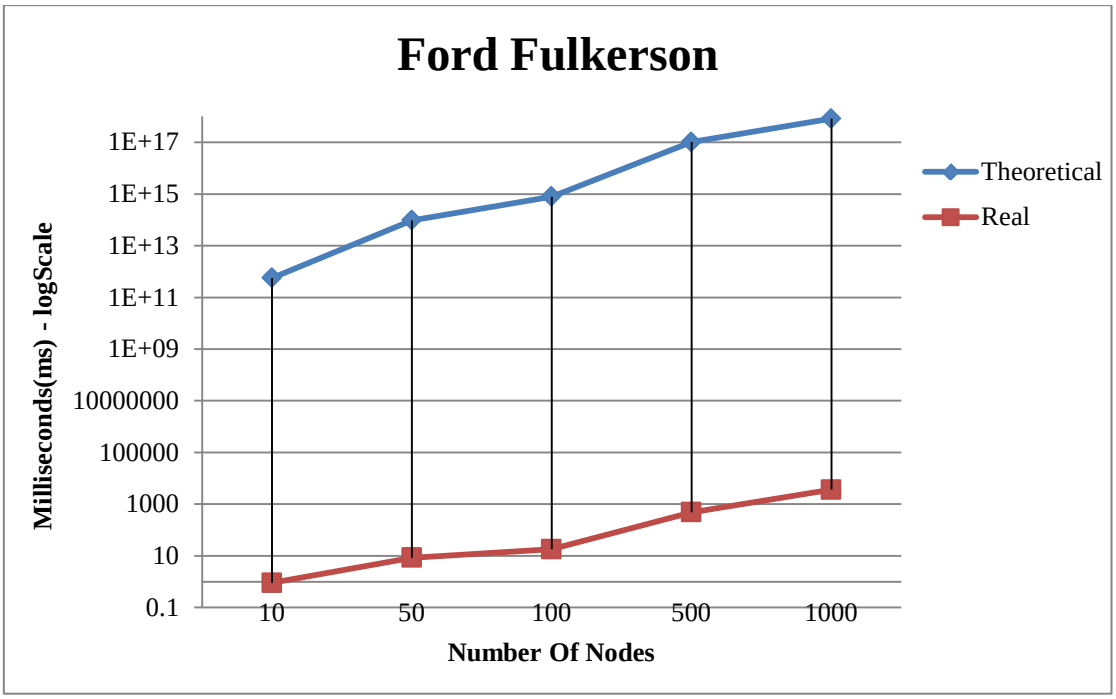
Εικάζουμε ότι αυτό παρατηρείται διότι λόγω του ότι ο γράφος είναι πολύ πυκνός με απευθείας ακμές από όλους τους κόμβους προς όλους τους άλλους. Αυτό επηρεάζει θετικά την απόδοση των αλγορίθμων που στις πλείστες περιπτώσεις θα επιλέξουν τα πιο μικρά μονοπάτια.

Τοπολογία 3

Ο αλγόριθμος Ford Fulkerson[1]επηρεάζεται άμεσα από τον σχεδιασμό της τοπολογίας αυτής διότι οι ακμές από την πηγή προς οποιοδήποτε άλλο κόμβο έχουν πολύ μεγάλη χωρητικότητα. Οπότε αφού το άθροισμα των χωρητικοτήτων αυτών είναι ένας πολύ μεγάλο αναμένουμε ότι θα επηρεάσει πολύ τον χρόνο που χρειάζεται ο αλγόριθμος για να επιλύσει ένα πρόβλημα που δημιουργείται με βάση αυτή την τοπολογία. Δεν μπορούμε να προσδιορίσουμε την επιρροή που θα έχει ο αριθμός των ακμών που υπάρχουν σε κάθε γράφο διότι είναι τυχαίος. Ξέρουμε όμως ότι οι γράφοι που δημιουργούνται ειδικά για μικρό αριθμό κόμβων είναι αρκετά πυκνοί λόγω του τρόπου που προγραμματίστηκε ο κώδικας για την τοπολογία αυτή. Οπότε αυτό που αναμένουμε είναι να δούμε

δύο παρόμοια μεταξύ γραφήματα διότι και οι δυο παράμετροι που επηρεάζουν τον θεωρητικό χρόνο του αλγορίθμου περιμένουμε ότι θα ασκήσουν επιρροή και στον πραγματικό του χρόνο.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 3



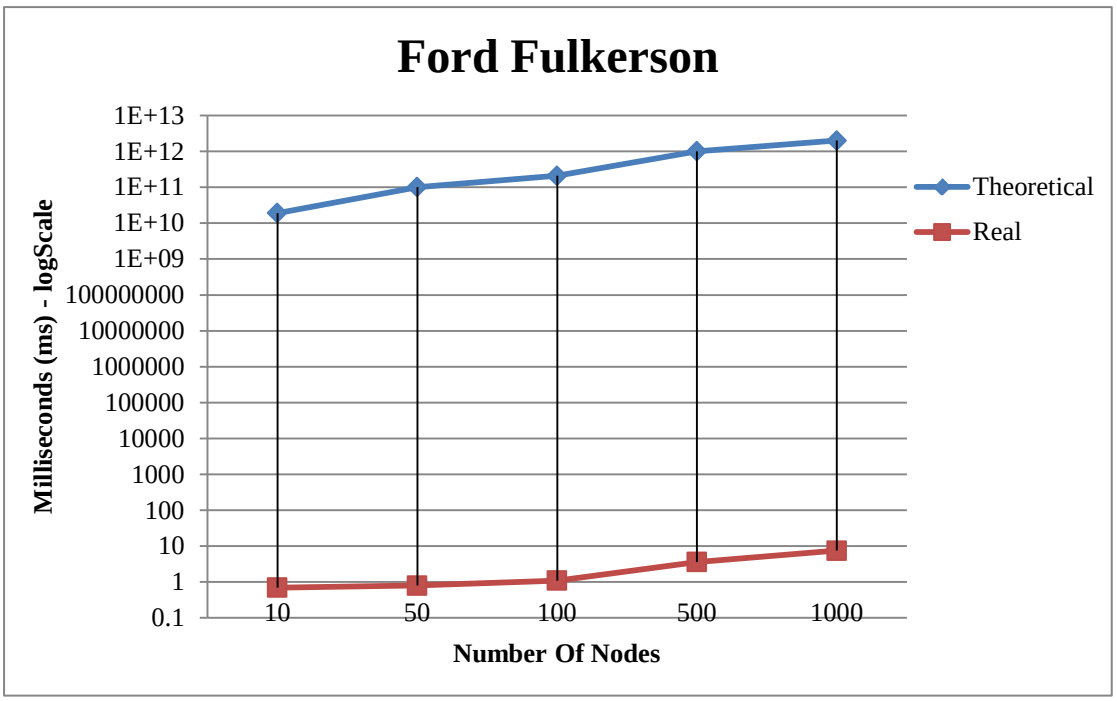
Όπως ήταν αναμενόμενο τα δύο γραφήματα είναι σχεδόν ακριβώς ίδια όσον αφορά την συμπεριφορά τους και αυτό μας επιβεβαιώνει ότι πράγματι ο πραγματικός χρόνος του αλγορίθμου επηρεάστηκε ακριβώς όπως το περιμέναμε από τις δύο παραμέτρους που επηρεάζει η Τοπολογία 3.

Τοπολογία 4

Ο αλγόριθμος αυτός είναι άμεσα επηρεαζόμενος από την τοπολογία αυτή λόγω του ότι ο χρόνος επηρεάζεται από δύο παραμέτρους, μία εκ των οποίων εδώ είναι στόχος. Η παράμετρος που επηρεάζεται το άθροισμα των χωρητικοτήτων όλων των ακμών που ξεκινούν από την πηγή και καταλήγουν σε οποιοδήποτε άλλο κόμβο. Στην Τοπολογία 4 οι ακμές αυτές έχουν μια τεράστια χωρητικότητα. Η άλλη παράμετρος που επηρεάζει αυτόν τον αλγόριθμο που είναι ο αριθμός των ακμών που υπάρχουν στον γράφο στην συγκεκριμένη τοπολογία ακόμα και για πολύ μεγάλο αριθμό κόμβων παραμένει πολύ μικρός. Οπότε αυτό που αναμένουμε λόγω του ότι ο γράφος είναι πολύ αραιός είναι ο χρόνος του αλγορίθμου να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και κατά συνέπεια και των ακμών με ήπια συμπεριφορά. Αυτό στηρίζεται στο ότι επειδή ο γράφος είναι πολύ

αραιός, με βάση τον σχεδιασμό της τοπολογίας δεν θα υπάρχουν πολλές ακμές οι οποίες θα ξεκινούν από την πηγή και θα καταλήγουν σε οποιοδήποτε άλλο κόμβο και έτσι το άθροισμα των χωρητικοτήτων όλων των ακμών που ξεκινούν από την πηγή δεν θα παίξει τόσο σημαντικό ρόλο στον πραγματικό του χρόνο.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 4

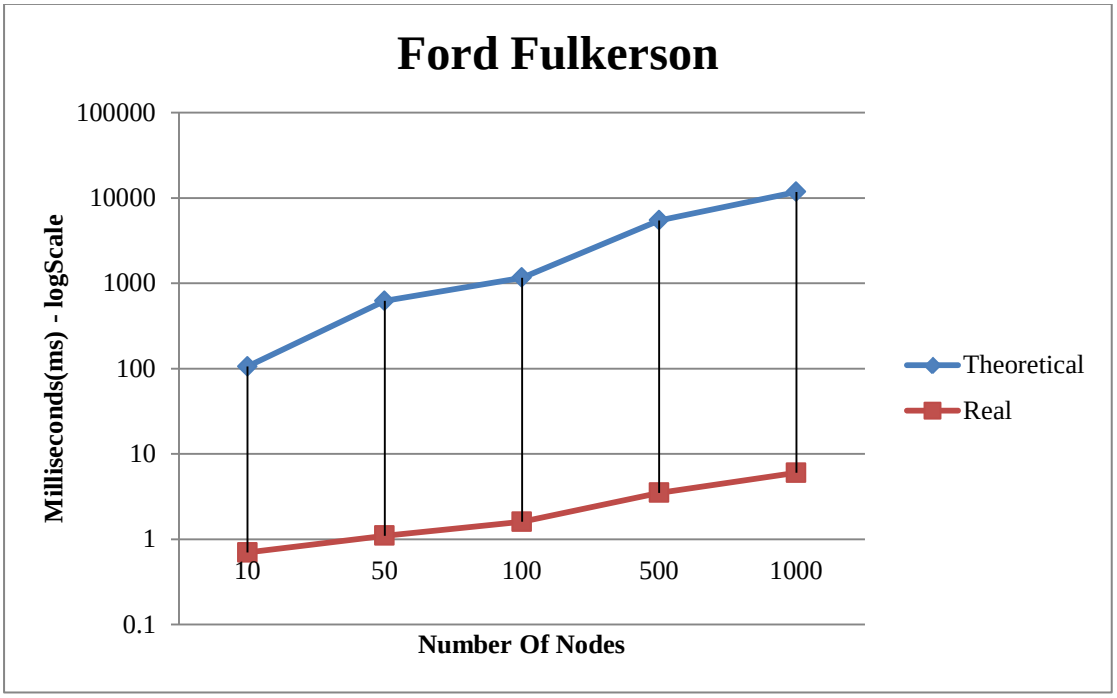


Αυτό που παρατηρούμε είναι ότι η γραφική παράσταση που προέκυψε από τη αξιολόγηση του αλγορίθμου σε αυτή την τοπολογία είναι πράγματι αυτό αναμέναμε. Τα δύο γραφήματα είναι εντελώς όμοια, και πράγματι η συμπεριφορά του γραφήματος του πραγματικού χρόνου που χρειάζεται ο αλγόριθμος για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα γράφο είναι όντως ήπια, αυξάνεται δηλαδή με σταθερό ρυθμό.

Τοπολογία 5

Στην συγκεκριμένη τοπολογία δεν υπάρχει κάτι που να επηρεάζει θεωρητικά σε μεγάλο βαθμό τον χρόνο του αλγορίθμου διότι οι παράμετροι που θα μπορούσαν να επηρεάσουν τον χρόνο του είναι ο αριθμός των ακμών και το άθροισμα των χωρητικοτήτων των ακμών από την πηγή σε οποιοδήποτε άλλο κόμβο. Βάση του σχεδιασμού της τοπολογίας ο αριθμός των ακμών είναι πολύ μικρός . Επίσης το άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή είναι ένας σχετικά μικρός αριθμός αφού οι χωρητικότητες που μπορούν να πάρουν οι ακμές είναι πολύ μικρές τιμές. Οπότε αυτό που αναμένουμε να δούμε είναι τον πραγματικό χρόνο να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων. Περιμένουμε ότι η γραφική παράσταση που θα προκύψει θα απεικονίζει δύο ίδιας συμπεριφοράς γραφήματα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 5

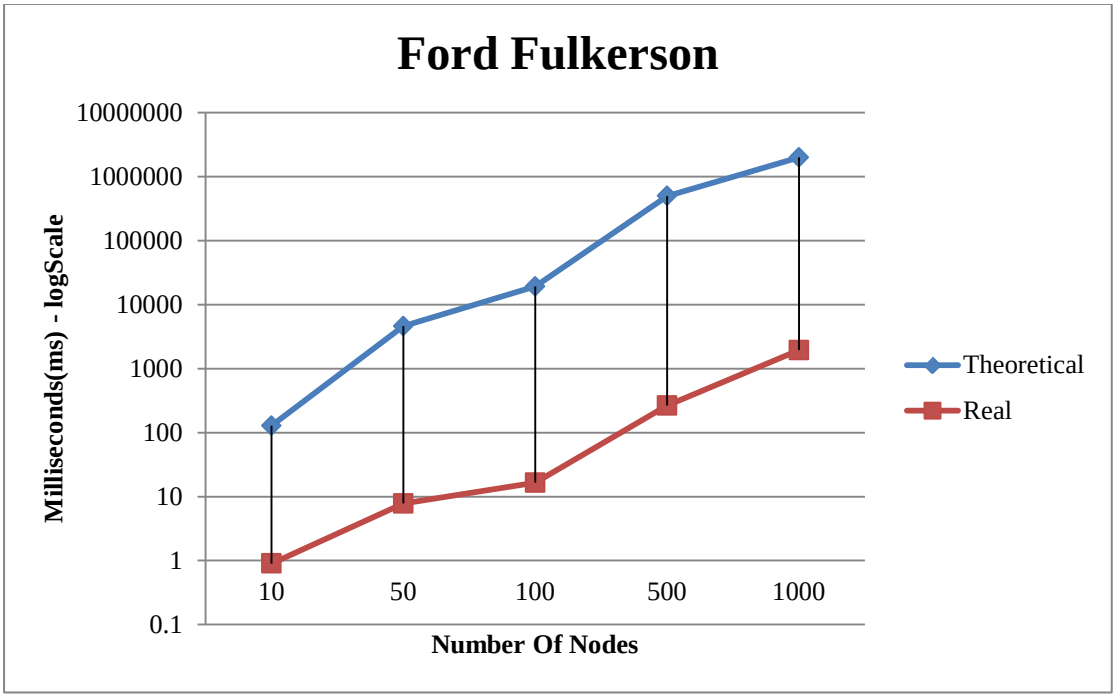


Παρατηρούμε ότι τα δύο γραφήματα που προέκυψαν είναι πολύ παρόμοια μεταξύ τους, όμως παρατηρούμε επίσης ότι από ένα σημείο και μετά η συμπεριφορά του γραφήματος του θεωρητικού χρόνου παρουσιάζει μια πιο έντονη συμπεριφορά σε σχέση με το γράφημα του πραγματικού χρόνου. Η διαφορά ανάμεσα στα δύο γραφήματα είναι αμελητέα.

Τοπολογία 6

Παρατηρώντας τις δύο παραμέτρους που επηρεάζουν τον χρόνο του αλγορίθμου Ford Fulkerson[1]βλέπουμε ότι με βάση την Τοπολογία 6 αυτή ο αριθμός των ακμών είναι σχεδόν διπλάσιος από τον αριθμό των κόμβων, άρα είναι ένας μικρός σχετικά αριθμός. Επίσης το άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή είναι σχεδόν με τον αριθμό των κόμβων αφού η χωρητικότητα της κάθε ακμής είναι ίση με ένα. Οπότε αυτό που αναμένουμε είναι να δούμε δύο πολύ όμοια μεταξύ τους γραφήματα που θα αυξάνονται όσο μεγαλώνει ο αριθμός των κόμβων.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 6

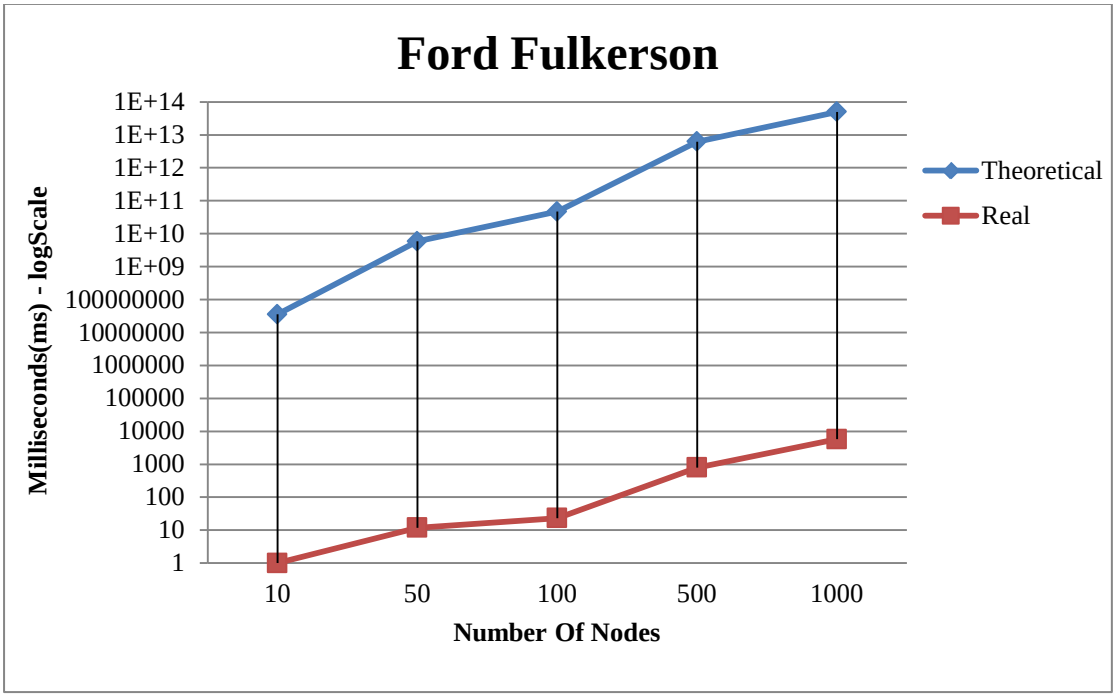


Όπως φαίνεται ξεκάθαρα και στην γραφική παράσταση η συμπεριφορά του θεωρητικού χρόνου του αλγόριθμου είναι ίδια με την συμπεριφορά του πραγματικού χρόνου που χρειάζεται ο αλγόριθμος για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα γράφο που δημιουργείται με βάση την Τοπολογία 6.

Τοπολογία 7

Στην Τοπολογία 7 δημιουργούνται τυχαίοι και σχετικά πυκνοί γράφοι με ένα μεγάλο εύρος τιμών στις χωρητικότητες των ακμών. Οι πιο πάνω παράμετροι που επηρεάζουν την τοπολογία επηρεάζουν άμεσα και τον αλγόριθμο Ford Fulkerson. Οπότε αναμένουμε ότι σίγουρα ο χρόνος του θα αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων, διότι θα μεγαλώνει ταυτόχρονα και ο αριθμός των ακμών. Επίσης λόγω του σχεδιασμού της τοπολογίας το άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή μεγαλώνει όσο μεγαλώνει ο αριθμός των κόμβων, με βάση τα αποτελέσματα που εξασφαλίσαμε από τα πειράματα μας. Οπότε αυτό που αναμένουμε να δούμε είναι το πραγματικό χρόνο του αλγορίθμου να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων. Επίσης αναμένουμε ότι η συμπεριφορά των δύο γραφημάτων που θα δημιουργηθούν θα είναι παρόμοια.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 7

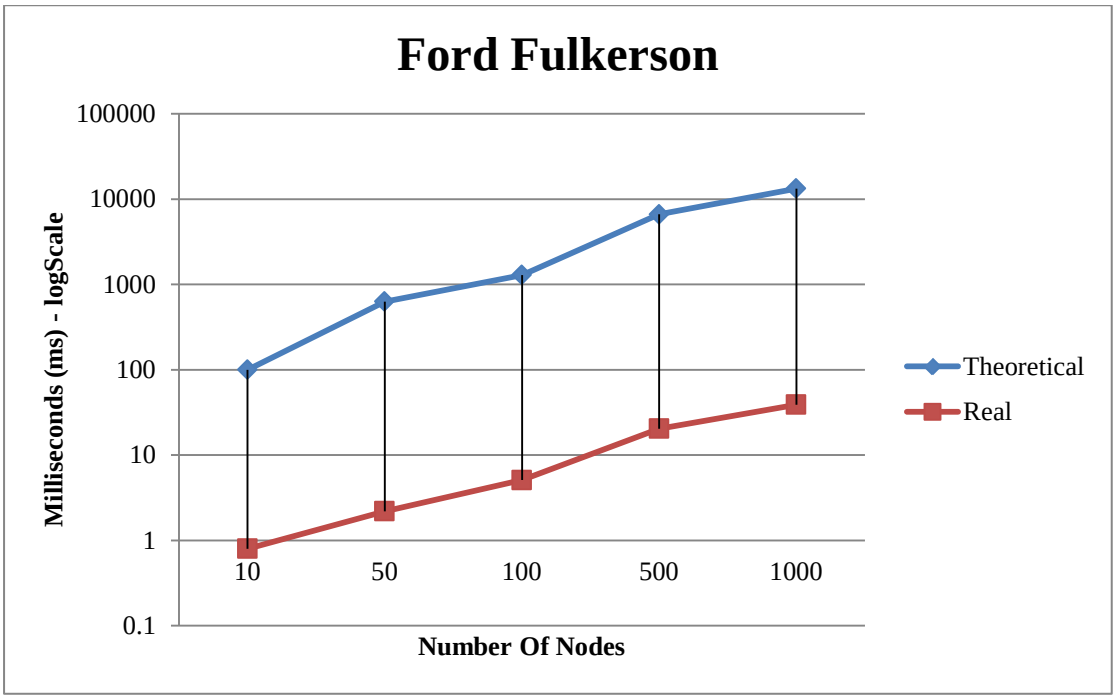


Η γραφική παράσταση που προέκυψε είναι ακριβώς αυτό που αναμέναμε για τους λόγους που περιγράψαμε πιο πάνω.

Τοπολογία 8

Στόχος σε αυτή την τοπολογία είναι ο ίδιος ο αλγόριθμος Ford Fulkerson. όμως η τοπολογία αυτή δεν επηρεάζει το χρόνο του αλγορίθμου ως προς τις παραμέτρους που τον επηρεάζουν αλλά ως προς την τεχνική με την οποία διαλέγει τα μονοπάτια μέσω των οποίων θα μπορέσει να στείλει την μέγιστη ροή που μπορεί από την πηγή στον προορισμό. όπως Ξέρουμε ότι ο χρόνος του αλγορίθμου επηρεάζεται από το άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή. Στην συγκεκριμένη τοπολογία το άθροισμα αυτό είναι μια σχετικά μικρή τιμή διότι η μεγαλύτερη χωρητικότητα που μπορεί να πάρει μια ακμή είναι δέκα. Οπότε αυτό που αναμένουμε να δούμε είναι ότι ο πραγματικός χρόνος του αλγορίθμου θα αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων, διότι ταυτόχρονα μεγαλώνει και ο αριθμός των ακμών που επηρεάζουν θεωρητικά το χρόνο του αλγορίθμου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 8

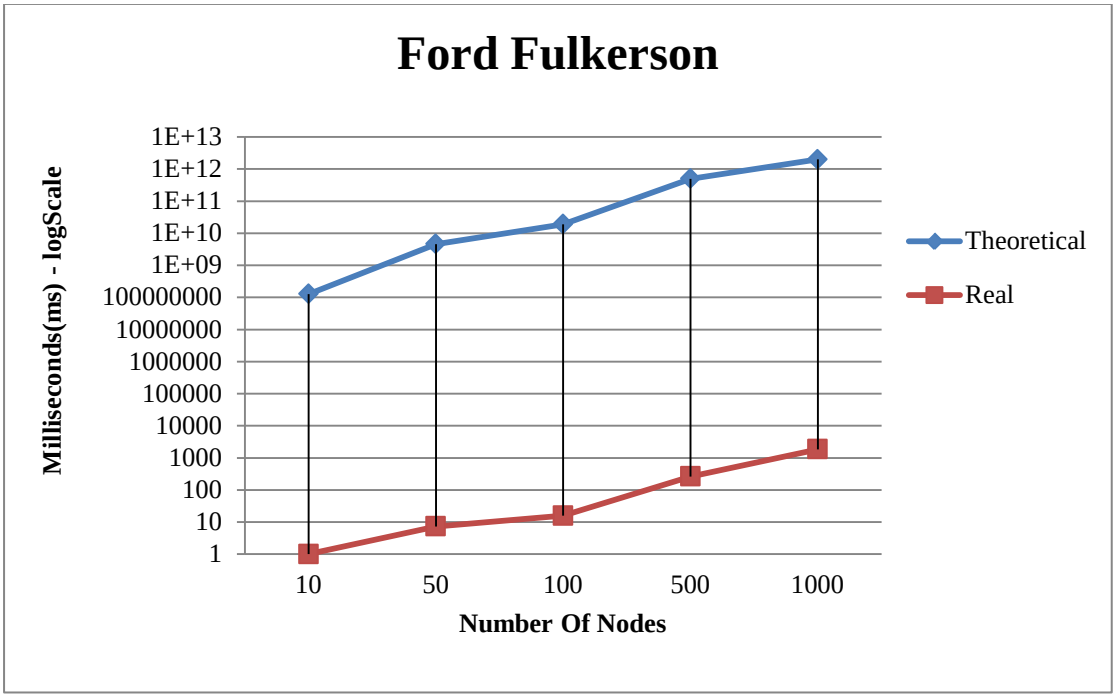


Η γραφική παράσταση που προέκυψε είναι ακριβώς αυτό που αναμέναμε για τους λόγους που περιγράψαμε πιο πάνω.

Τοπολογία 9

Τα χαρακτηριστικά αυτής της τοπολογίας μας υποδηλώνουν ότι ο χρόνος του αλγορίθμου αυτού θα μεγαλώνει όσο μεγαλώνει ο αριθμός των κόμβων. Ο λόγος που πιστεύουμε ότι θα συμβεί αυτό είναι επειδή όσο μεγαλώνει ο αριθμός των κόμβων με τον ίδιο ακριβώς ρυθμό μεγαλώνει και το άθροισμα όλων των χωρητικοτήτων των ακμών που προέρχονται από την πηγή. Επίσης ξέρουμε εκ των προτέρων τον αριθμό των ακμών σε σχέση με τον αριθμό των κόμβων. Ο αριθμός των ακμών είναι σχεδόν διπλάσιος από τον αριθμό των κόμβων, αυτό ισχύει για οποιοδήποτε αριθμό κόμβων. Ακόμα περιμένουμε ότι τα δύο γραφήματα που θα προκύψουν θα είναι σχεδόν εντελώς όμοια διότι δεν πιστεύουμε ότι υπάρχει κάποια άλλη παράμετρος εκτός από αυτές που ήδη αναφέραμε που να επηρεάζουν το ν χρόνο του αλγορίθμου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Ford Fulkerson[1]στην Τοπολογία 9



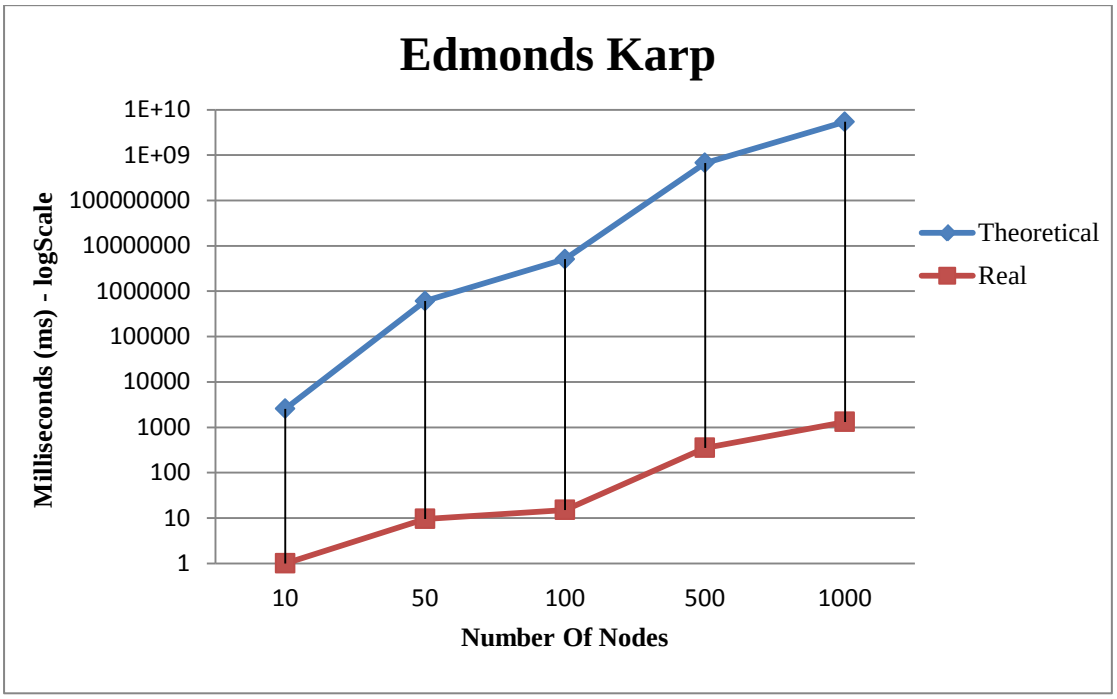
Η γραφική παράσταση είναι αυτή που αναμέναμε ακριβώς.

5.4.1.2 Αλγόριθμος Edmonds Karp[3]

Τοπολογία 1

Ο αλγόριθμος Edmonds Karp[3]αποτελεί το στόχο της τοπολογίας αυτής. Σε αυτή την τοπολογία θέλαμε να αποδυναμώσουμε τον αλγόριθμο Edmonds Karp[3]και έτσι στοχεύσαμε στον τρόπο που επιλέγει το μονοπάτι μέσω του οποίου θα στείλει ροή από την πηγή στον προορισμό. Ο αλγόριθμος αυτός λόγω του ότι διαλέγει τα πιο σύντομα μονοπάτια για να στείλει τη ροή του και στην τοπολογία αυτή είναι τα πιο στενά περιμένουμε ότι εκτός από τις παραμέτρους που τον επηρεάζουν, που είναι ο αριθμός των ακμών και των κόμβων του γράφου, ο τρόπος επιλογής θα προσθέσει κάποια επιπλέον καθυστέρηση στον πραγματικό του χρόνο. Οπότε αυτό που αναμένουμε είναι ότι ο πραγματικός χρόνος που παρουσιάζει ο αλγόριθμος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων και των ακμών άρα θα είναι παρόμοιος με τον θεωρητικό χρόνο με λίγο πιο έντονη συμπεριφορά στον πραγματικό χρόνο λόγω του σχεδιασμού της τοπολογίας.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3]στην Τοπολογία 1.

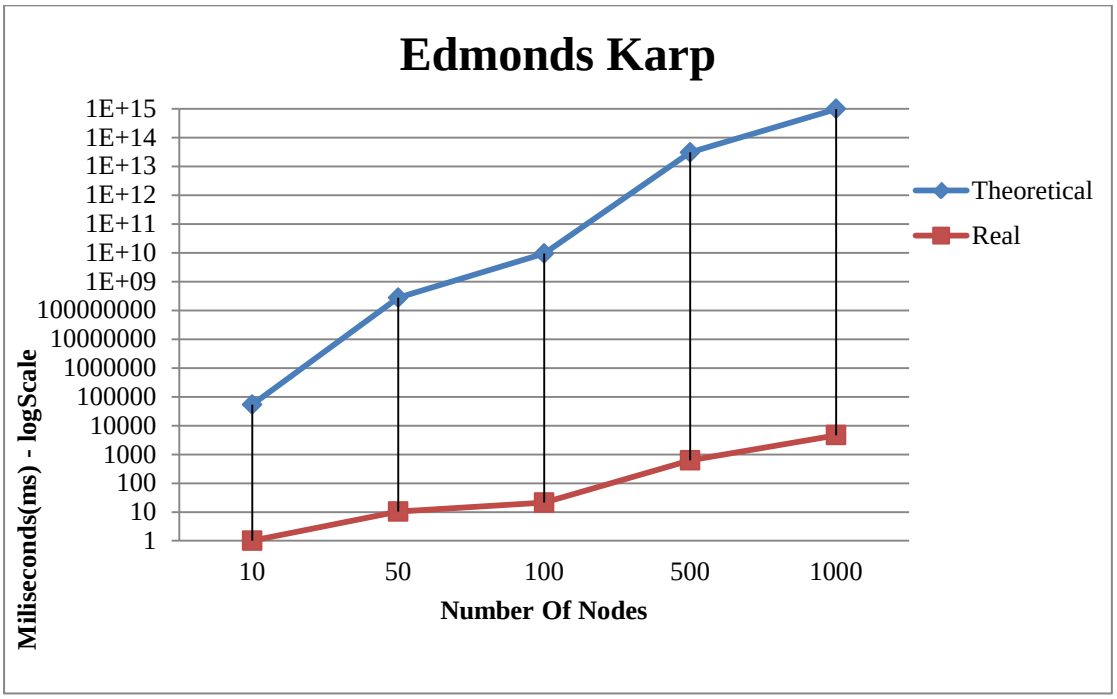


Όπως αναμέναμε ο πραγματικός χρόνος του αλγορίθμου είναι παρόμοιος με τον θεωρητικό όμως λόγω του ότι τα μονοπάτια που δημιουργούνται δεν είναι τόσο μεγάλα (δηλαδή δεν αποτελούνται από πολλά βήματα) , δεν επηρεάζεται τόσο πολύ ο χρόνος από τον αριθμό των ακμών έτσι δεν μπορούμε να διακρίνουμε την πιο έντονη συμπεριφορά στον πραγματικό χρόνο που θα προέκυπτε από τον τρόπο επιλογής των μονοπατιών.

Τοπολογία 2

Στην τοπολογία αυτή λόγω του ότι οι γράφοι που δημιουργούνται είναι πολύ πυκνοί, αυτό έχει σαν αποτέλεσμα την ύπαρξη ενός συγκεκριμένου για κάθε αριθμό κόμβων, μεγάλου αριθμού ακμών. Ο αριθμός των ακμών στον αλγόριθμο αυτό καθορίζουν τον χρόνο που χρειάζεται για να υπολογίσει την Μέγιστη Ροή σε ένα γράφο. Οπότε αυτό που αναμένουμε είναι εφόσον ο θεωρητικός χρόνος πρέπει να αυξάνεται σε όσο αυξάνεται ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών, είναι να παρατηρούμε την ίδια αύξηση και στο γράφημα του πραγματικού χρόνου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3]στην Τοπολογία 2.



Παρατηρώντας τα αποτελέσματα που προέκυψαν βλέπουμε ότι η θεωρητική και η πραγματική ανάλυση είναι όμοιες για μικρό αριθμό κόμβων, ενώ για μεγάλο αριθμό κόμβων και ακμών αντί να χειροτερεύει ο πραγματικός χρόνος ακριβώς όπως και ο θεωρητικός, παραμένει σχετικά σταθερός και αυξάνεται με πολύ μικρό ρυθμό. Οπότε παρατηρούμε μια διαφοροποίηση μεταξύ του θεωρητικού και του πραγματικού χρόνου όσο αυξάνεται αριθμός των κόμβων και κατ' επέκταση και ο αριθμός των ακμών.

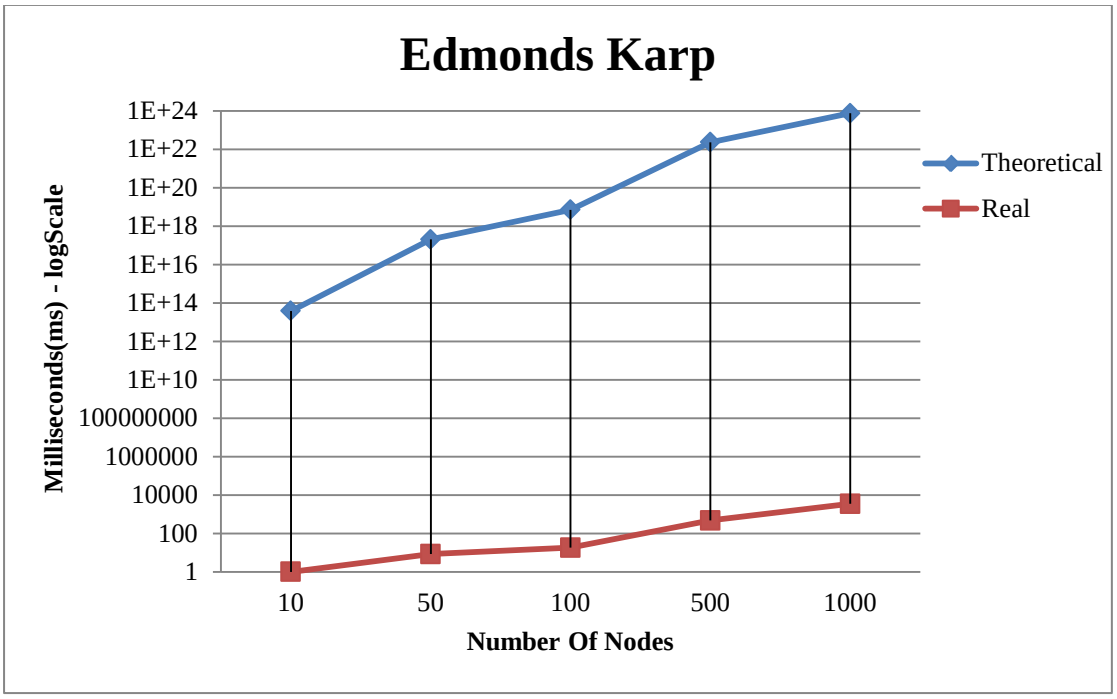
Εικάζουμε ότι αυτό συνέβηκε επειδή ο αλγόριθμος Edmonds Karp[3]επειδή επιλέγει πάντα τα πιο μικρά μονοπάτια, δηλαδή τα μονοπάτια με τα πιο λίγα βήματα από την πηγή στον προορισμό και λόγω του ότι σε αυτή την τοπολογία υπάρχουν πολλά τέτοια μονοπάτια, μπορεί να στείλει ίσως αρκετή από τη ροή μέσω αυτών των σύντομων μονοπατιών χωρίς να χρειάζεται να ακολουθήσει

άλλα μεγαλύτερα μονοπάτια. Έτσι για αυτό το λόγο εξοικονομεί χρόνο και γίνεται καλύτερος σε σχέση με τον θεωρητική συμπεριφορά που έπρεπε να παρουσιάζει.

Τοπολογία 3

Ο χρόνος του αλγορίθμου αυτού εξαρτάται από τον αριθμό των ακμών και από τον αριθμό των κόμβων. Η μόνη παράμετρος που επηρεάζεται από τον σχεδιασμό της τοπολογίας είναι ο αριθμός των ακμών, επειδή λόγω του ότι οι γράφοι που δημιουργούνται είναι πολύ πυκνοί, αναμένουμε ότι ο πραγματικός χρόνος του αλγορίθμου θα αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων. Οπότε περιμένουμε ότι το γράφημα του πραγματικού χρόνου που χρειάζεται ο αλγόριθμος για να επιλύσει το πρόβλημα Εύρεσης Μέγιστης Ροής στους γράφους που προκύπτουν με βάση την Τοπολογία 3, να έχει ίδια συμπεριφορά με το θεωρητικό γράφημα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3]στην Τοπολογία 2



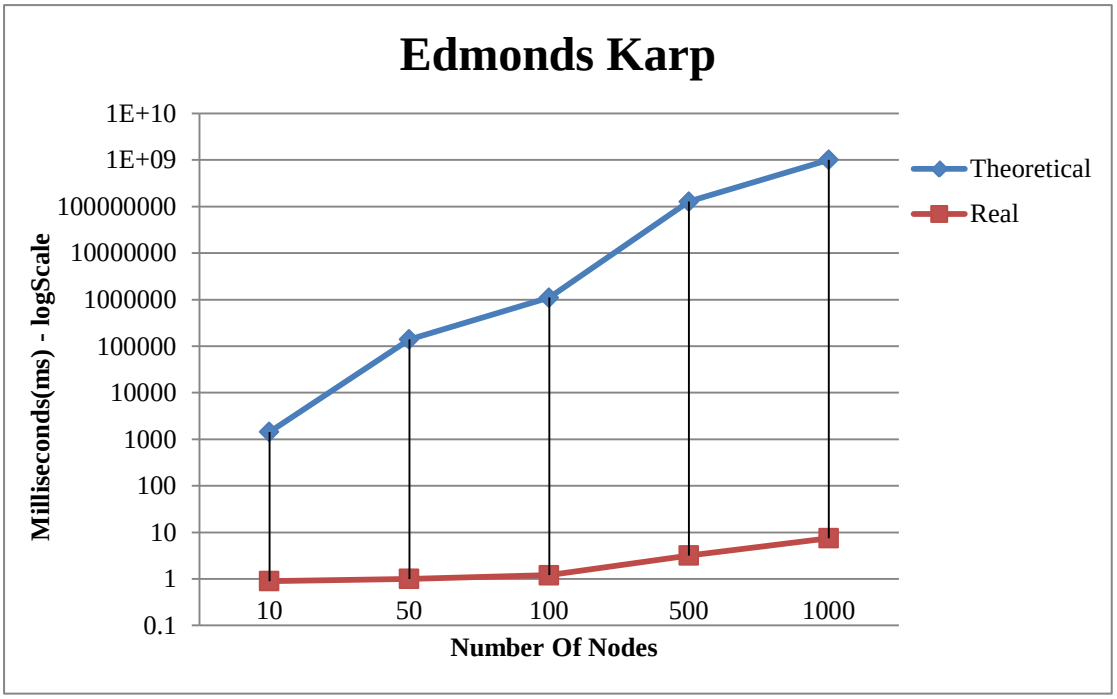
Όπως αναμέναμε τα δύο γραφήματα είναι ίδιας συμπεριφοράς ενώ βλέπουμε επίσης ότι ο πραγματικός χρόνος του αλγορίθμου βελτιώνεται σε σχέση με τον θεωρητικό του χρόνο όσο μεγαλώνει ο αριθμός των κόμβων. Όσο πιο μεγάλοι είναι δηλαδή οι γράφοι που δημιουργούνται τόσο πιο καλό χρόνο έχει ο αλγόριθμος.

Εικάζουμε ότι αυτό συμβαίνει επειδή οι γράφοι που δημιουργούνται είναι αρκετά πυκνοί λόγω του προγραμματισμού της τοπολογίας , που σε κάποιες περιπτώσεις είναι σχεδόν όλοι οι κόμβοι με όλους του άλλους κόμβους ενωμένοι απευθείας με μια ακμή, τα μονοπάτια που ακολουθεί ο αλγόριθμος είναι πολύ σύντομα και λόγω της τεχνικής του κυρίως αλλά και λόγω των μικρών μονοπατιών που δημιουργούνται από την ίδια την τοπολογία.

Τοπολογία 4

Η πιο σημαντική παράμετρος που επηρεάζει το χρόνο αυτού του αλγορίθμου είναι ο αριθμός των ακμών που σε αυτή την τοπολογία είναι πολύ μικρός λόγω του ότι ο γράφος είναι πολύ αραιός. Οπότε αυτό που αναμένουμε είναι ο πραγματικός χρόνος του αλγορίθμου, όπως και ο θεωρητικός χρόνος, να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων. Επίσης περιμένουμε ότι η αύξηση θα γίνεται με σταθερό ρυθμό.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3]στην Τοπολογία 4



Η γραφική παράσταση που προέκυψε είναι όντως αυτό που αναμέναμε ως ένα σημείο. Τόσο ο πραγματικός όσο και ο θεωρητικός χρόνος αυξάνονται όσο μεγαλώνει ο αριθμός των κόμβων με σχετικά σταθερό ρυθμό. Αυτό που παρατηρούμε είναι ότι ο πραγματικός χρόνος είναι λίγο καλύτερος από ότι θεωρητικά περιμέναμε.

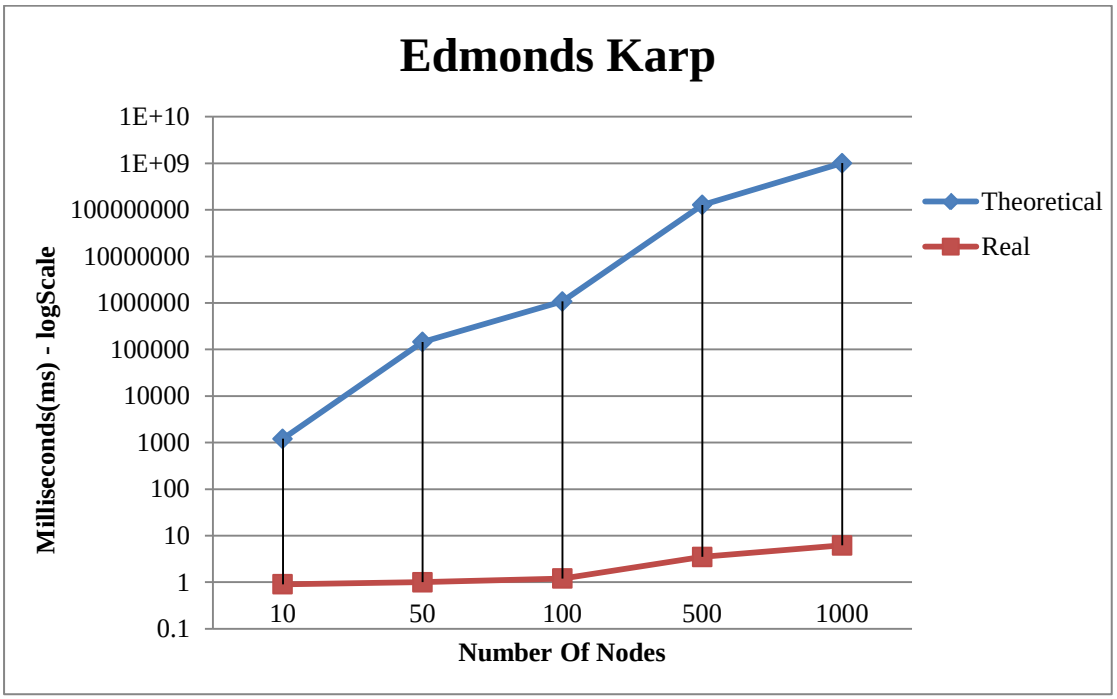
Εικάζουμε ότι αυτό συμβαίνει επειδή λόγω του ότι ο γράφος είναι αραιός, δεν υπάρχουν πολλά μονοπάτια που να ξεκινούν από την πηγή και να φτάνουν στον προορισμό αυτό κάνει τη δουλεία του αλγορίθμου καλύτερη.

Τοπολογία 5

Ο χρόνος του αλγόριθμου Edmonds Karp[3]είναι επηρεαζόμενος από των αριθμό των ακμών και από τον αριθμό των κόμβων. Με βάση τον σχεδιασμό της τοπολογίας ξέρουμε ότι ο αριθμός των

ακμών είναι μια σχετικά μικρή τιμή. Οπότε δεν υπάρχει κάποια παράμετρος που να επηρεάζει τον χρόνο του αλγόριθμου έχοντας ως αποτέλεσμα μια έντονη συμπεριφορά στην γραφική παράσταση. Θεωρητικά ο χρόνος που προκύπτει πρέπει να είναι τουλάχιστον κυβικός σε σχέση με τον αριθμό των κόμβων. Αυτό λοιπόν που αναμένουμε να δούμε στην συγκριτική γραφική παράσταση είναι δύο γραφήματα τα οποία είναι όμοια μεταξύ τους και τα οποία αυξάνονται όσο μεγαλώνει ο χρόνος .

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3] στην Τοπολογία 5



Παρατηρώντας την γραφική παράσταση αυτό που βλέπουμε είναι ότι πράγματι ο πραγματικός χρόνος αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων με πιο ήπιο ρυθμό όμως από ότι αναμέναμε.

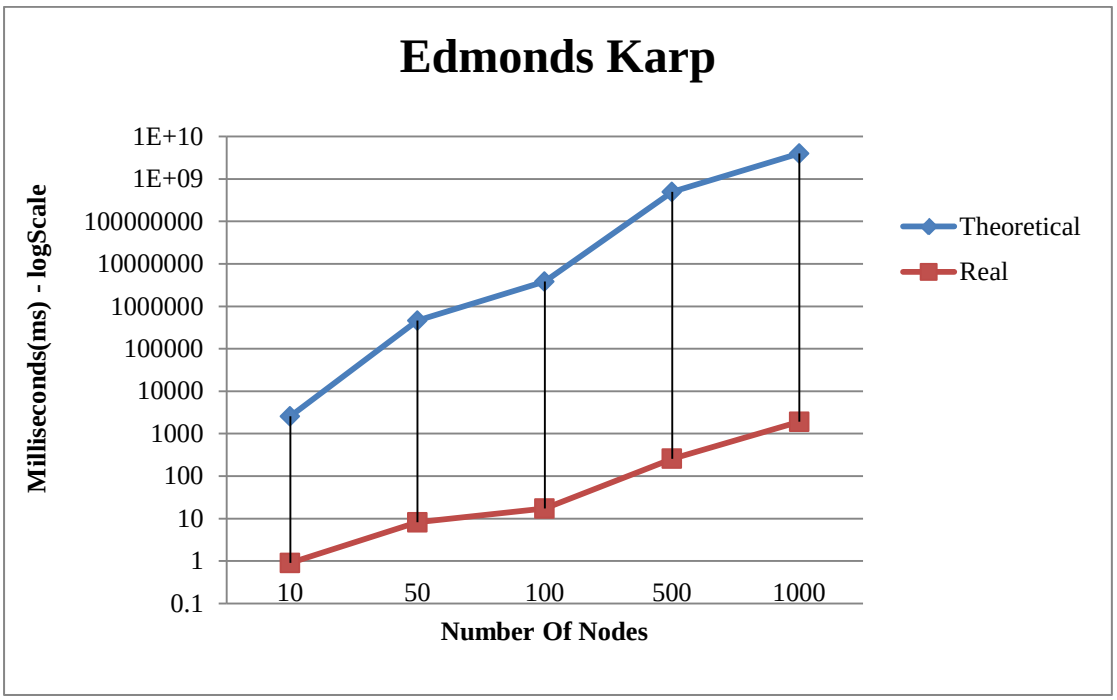
Εικάζουμε ότι αυτό συμβαίνει επειδή οι γράφοι που δημιουργούνται δεν περιέχουν πολλά μονοπάτια από τη πηγή στον προορισμό ,λόγω της τακτικής δημιουργίας των γράφων στην συγκεκριμένη τοπολογία και έτσι διευκολύνεται ακόμα περισσότερο η δουλειά του αλγορίθμου ο οποίος ακολουθεί πρώτα το πιο σύντομο μονοπάτι που υπάρχει που ίσως είναι και μοναδικό ακόμα και για μεγάλο αριθμό κόμβων. Οπότε ναι υπάρχουν οι ακμές στον γράφο τις οποίες εμείς υπολογίζουμε στον θεωρητικό χρόνο όμως αυτές οι ακμές δεν δημιουργούν μονοπάτια, υπάρχει σίγουρα όμως ένα μονοπάτι από την πηγή στον προορισμό.

Τοπολογία 6

Η παράμετρος που επηρεάζει τον αλγόριθμο Edmonds Karp[3] και υπάρχει πιθανότητα να επηρεαστεί από την τοπολογία είναι ο αριθμός των ακμών. Σε αυτή την τοπολογία ο αριθμός των ακμών είναι σχεδόν διπλάσιος από τον αριθμό των κόμβων. Οπότε αυτό που αναμένουμε είναι τόσο

το γράφημα του πραγματικού χρόνου, όσο και το γράφημα του θεωρητικού χρόνου θα αυξάνονται όσο μεγαλώνει ο αριθμός των κόμβων.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3] στην Τοπολογία 6

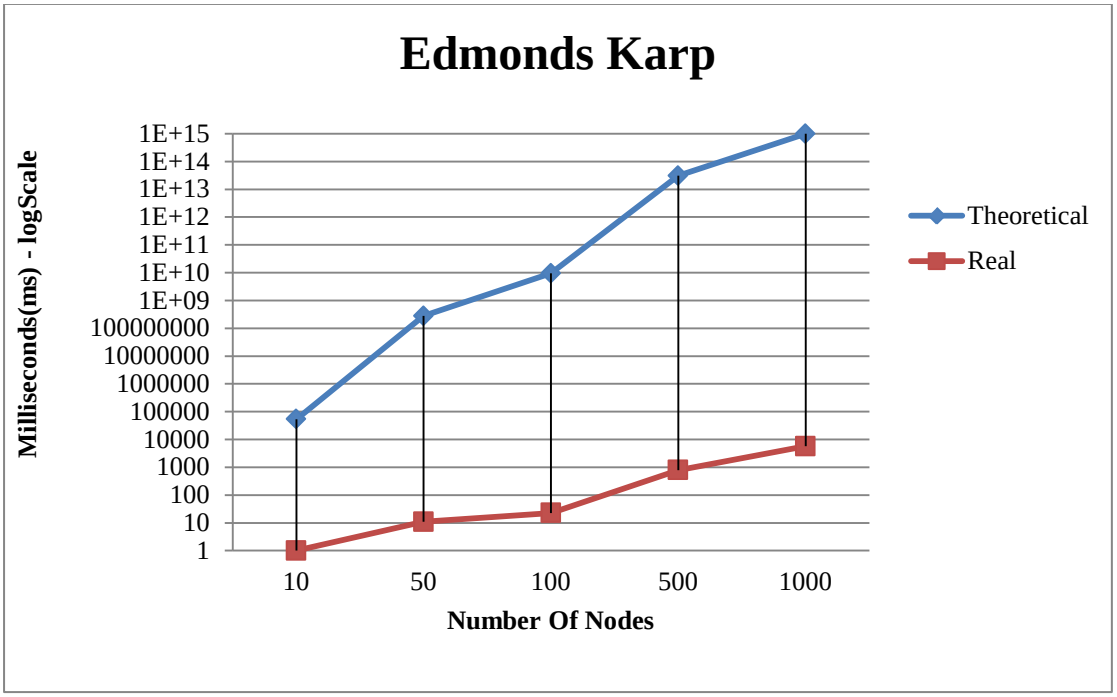


Όπως φαίνεται και από την γραφική παράσταση πράγματι ο χρόνος που χρειάζεται ο αλγόριθμος για να δώσει την μέγιστη ροή παίρνοντας σαν δεδομένο ένα γράφο που δημιουργήθηκε με βάση την Τοπολογία 6 αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών. Επίσης τα δύο γραφήματα που προέκυψαν είναι αρκετά όμοια μεταξύ τους με μια μικρή αμελητέα διαφορά.

Τοπολογία 7

Οι παράμετροι που επηρεάζουν τον χρόνο που χρειάζεται ο αλγόριθμος είναι όπως γνωρίζουμε ο αριθμός των ακμών και ο αριθμός των κόμβων. Βάση του σχεδιασμού της τοπολογία αυτή ξέρουμε ότι οι γράφοι που δημιουργούνται είναι αρκετά πυκνοί. Οπότε εφόσον ξέρουμε ότι, όσο μεγαλώνει ο αριθμός των κόμβων, ταυτόχρονα μεγαλώνει και ο αριθμός των ακμών αναμένουμε η γραφική παράσταση του χρόνου του αλγορίθμου θα έχει μια ανοδική πορεία. Επίσης υπολογίζουμε ότι ο πραγματικός χρόνος του αλγορίθμου θα έχει την ίδια συμπεριφορά με3 τον θεωρητικό του χρόνου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3] στην Τοπολογία 7



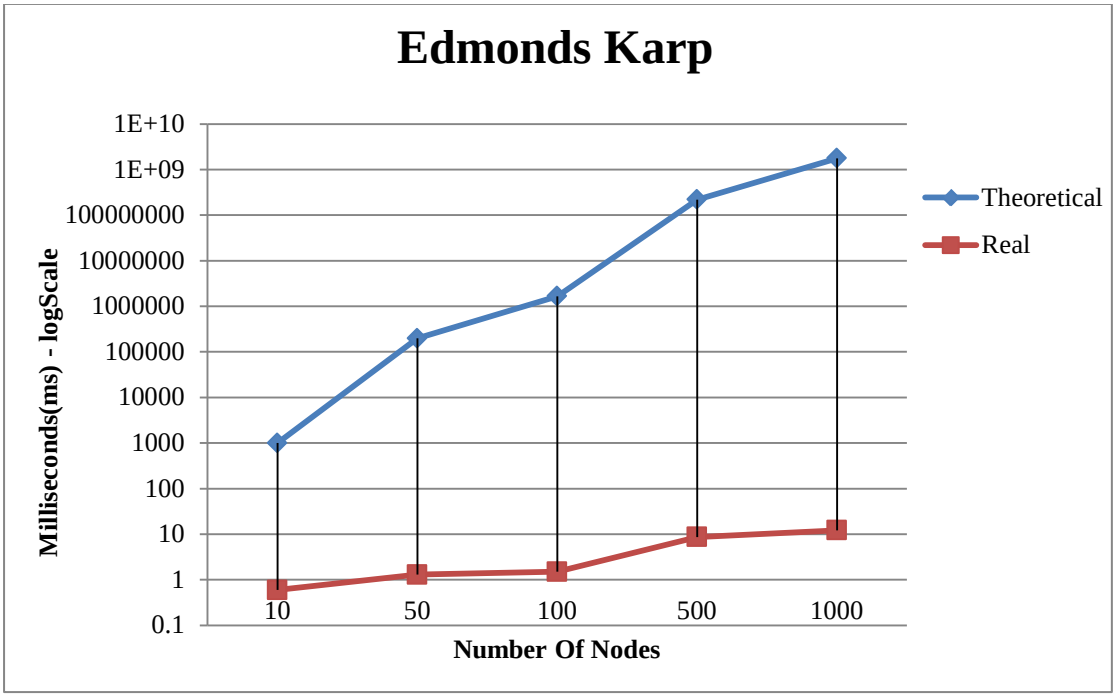
Η γραφική παράσταση που προέκυψε είναι εν μέρη αυτό που αναμέναμε. Τα δύο γραφήματα είναι μεταξύ τους όμοια, όμως παρατηρούμε ότι από ένα σημείο και μετά η γραφική παράσταση του πραγματικού χρόνου έχει πιο ήπια άνοδο από την γραφική παράσταση του θεωρητικού χρόνου.

Εικάζουμε ότι αυτό συμβαίνει λόγω του ότι ο γράφος είναι πολύ πυκνός οπότε ο αλγόριθμος Edmonds Karp[3] ίσως εκμεταλλεύεται την ιδιότητα αυτή και καταφέρνει να στείλει όλη του την ροή μέσω των πιο σύντομων μονοπατιών που υπάρχουν στον γράφο, τα οποία με βάση την τεχνική που χρησιμοποιεί ο αλγόριθμος θα επιλεγούν πρώτα για αποσταλεί η ροή από τη πηγή στον προορισμό.

Τοπολογία 8

Ο αλγόριθμος αυτός επηρεαζόμενος από τον αριθμό των ακμών και τον αριθμό των κόμβων αναμένουμε ότι θα μας δώσει ένα γράφημα με ανοδική πορεία. Δηλαδή όσο μεγαλώνει ο αριθμός των κόμβων περιμένουμε ότι και ο πραγματικός χρόνος του αλγορίθμου θα αυξάνεται. Επίσης περιμένουμε ότι η πραγματική ανάλυση του αλγορίθμου θα έχει παρόμοια συμπεριφορά με την θεωρητική του ανάλυση.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3] στην Τοπολογία 8



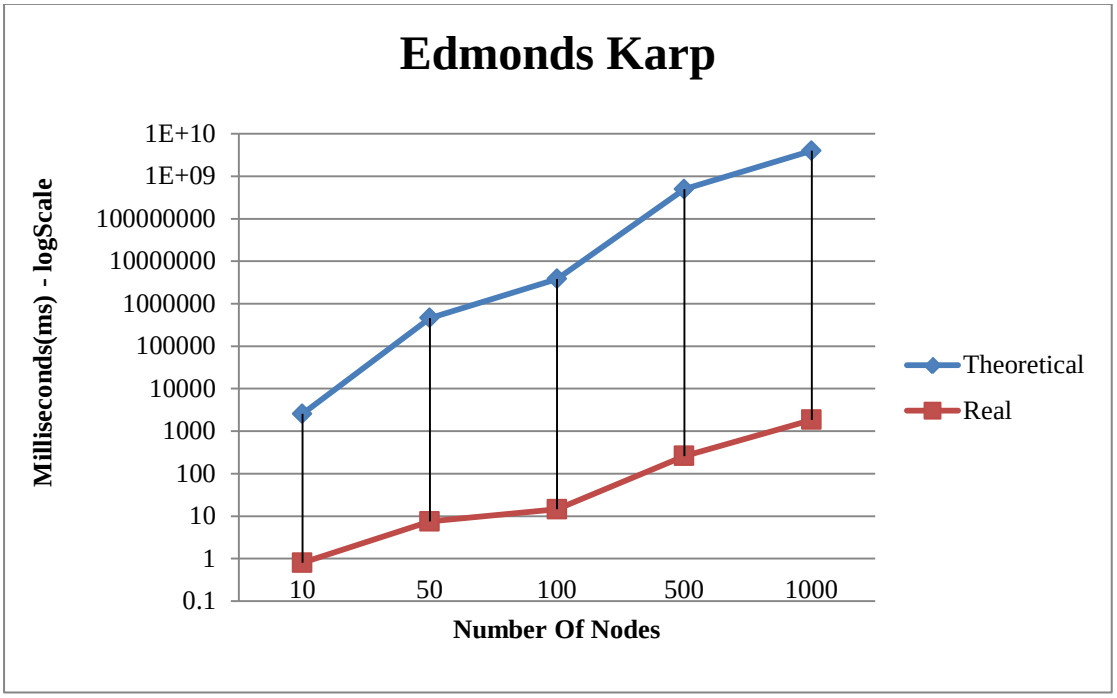
Παρατηρώντας την γραφική παράσταση που προέκυψε, διαπιστώνουμε ότι πράγματι ο πραγματικός χρόνος του αλγορίθμου έχει παρόμοια συμπεριφορά με τον θεωρητικό του χρόνο. Βλέπουμε ακόμα ότι όσο μεγαλώνει ο αριθμός των κόμβων η διαφορά μεταξύ του θεωρητικού και του πραγματικού χρόνου μεγαλώνει.

Εικάζουμε ότι αυτό συμβαίνει λόγω του σχεδιασμού της τοπολογίας, με βάση την οποία ο αλγόριθμος Edmonds Karp[3] στέλλει όλη την ροή με επιλέγοντας αμέσως ένα μονό μονοπάτι όσο μεγάλος και αν είναι οπ γράφος. Έτσι ειδικά όταν ο γράφος έχει μεγάλο αριθμό κόμβων η διαφορά είναι πιο μεγάλη και πιο εμφανής.

Τοπολογία 9

Ο αλγόριθμος αυτός είναι επηρεαζόμενος κυρίως από τον αριθμό των ακμών που υπάρχουν στον γράφο και επίσης είναι επηρεαζόμενος από τον αριθμό των κόμβων. Στην Τοπολογία 9 ο αριθμός των ακμών είναι συγκεκριμένος για οποιοδήποτε αριθμό κόμβων, είναι σχεδόν ο διπλάσιος αριθμός του αριθμού των κόμβων. Οπότε αυτό που αναμένουμε είναι το γράφημα του πραγματικού χρόνου που θα προκύψει να αυξάνεται με ένα σταθερό ρυθμό όσο μεγαλώνει ο αριθμός των κόμβων. Επιπρόσθετα αναμένουμε να δούμε δύο πολύ παρόμοια γραφήματα πάνω στην γραφική παράσταση.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Edmonds Karp[3] στην Τοπολογία 9



Η γραφική παράσταση είναι αυτό που αναμέναμε με την διαφορά ότι παρατηρούμε πως ο πραγματικός χρόνος είναι καλύτερος από τον θεωρητικό όσο μεγαλώνει ο αριθμός των κόμβων.

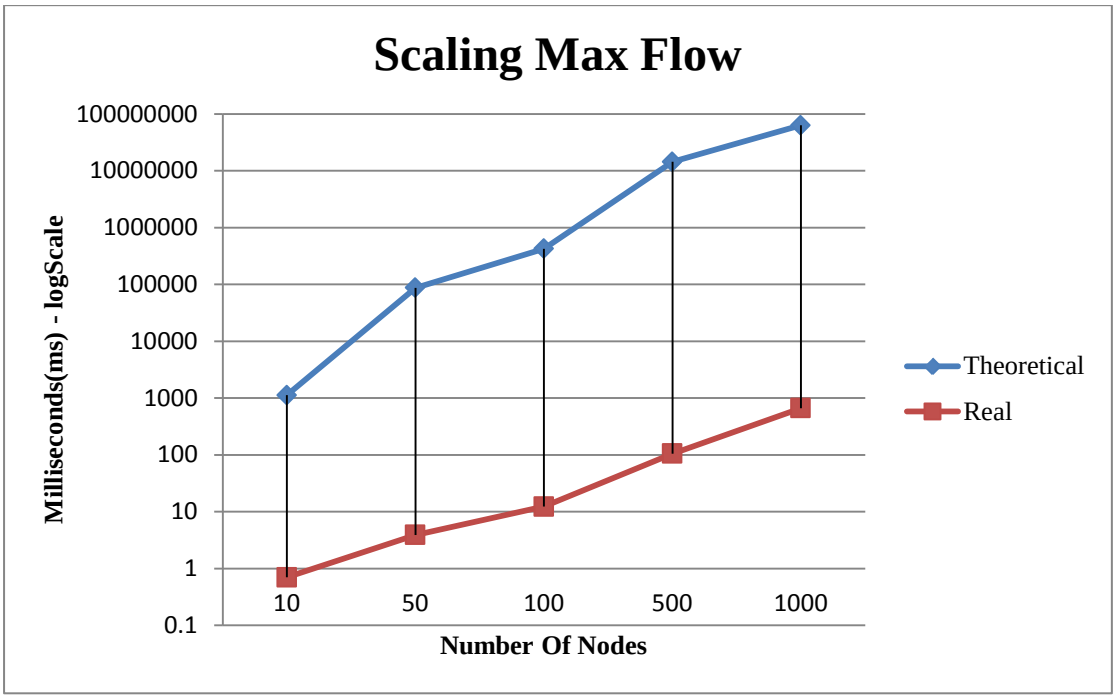
Εικάζουμε ότι αυτό συμβαίνει επειδή ο αλγόριθμος Edmonds Karp[3] επιλογής των πιο σύντομων μονοπατιών πρώτα για αποστολή ροής από την πηγή στον προορισμό και εδώ όλα τα μονοπάτια είναι ίσα οπότε ίσως αυτό να εξαλείφει κυρίως για μεγάλους κόμβους κάποιο overhead.

5.4.1.3 Αλγόριθμος Scaling Max Flow[1]

Τοπολογία 1

Αυτό που αναμένουμε είναι ο πραγματικός χρόνος του αλγορίθμου είναι να έχει ακριβώς ίδια συμπεριφορά με την θεωρητική ανάλυση του χρόνου που χρειάζεται ο αλγόριθμος Scaling Max Flow[1] για να επιλύσει το πρόβλημα της Εύρεσης Μέγιστης Ροής σε ένα γράφο που δημιουργείται με βάση την Τοπολογία 1. Ο λόγος που αναμένουμε κάτι τέτοιο είναι επειδή ο χρόνος του αλγορίθμου επηρεάζεται από τον αριθμό των ακμών και επίσης από το λογάριθμο του αθροίσματος των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι σχετικά μικρός , όπως επίσης και ο λογάριθμος του αθροίσματος των χωρητικοτήτων οπότε περιμένουμε ότι δεν θα τροποποιηθεί ο χρόνος διαφορετικά από τον θεωρητικό . Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 1.

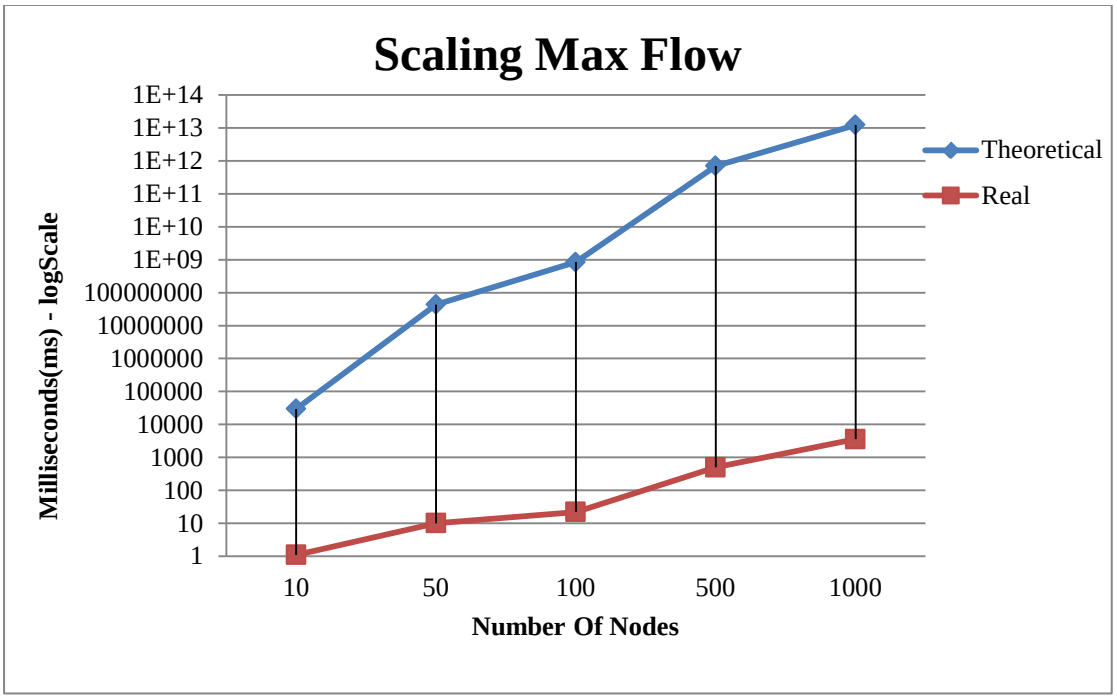


Όπως αναμέναμε η ανάλυση του πραγματικού χρόνου που χρειάζεται ο αλγόριθμος Scaling Max Flow[1] για να επιλύσει το πρόβλημα της Εύρεσης Μέγιστης Ροής σε ένα γράφο που δημιουργείται με βάση την Τοπολογία 1 έχει την ίδια συμπεριφορά με την ανάλυση του θεωρητικού χρόνου του αλγορίθμου με βάση των αριθμό των ακμών που είναι σταθερός και εξαρτάται από τον αριθμό των κόμβων όπως επίσης και από το άθροισμα των χωρητικοτήτων από την πηγή σε οποιοδήποτε άλλο κόμβο που η λογαριθμική τους τιμή είναι πολύ μικρή.

Τοπολογία 2

Ο χρόνος του αλγορίθμου αυτού επηρεάζεται άμεσα από τον αριθμό των ακμών και στη τοπολογία αυτή ο αριθμός των ακμών είναι πολύ μεγάλος. Επιπρόσθετα ο χρόνος του επηρεάζεται από τον λογάριθμο του αθροίσματος των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή και καταλήγουν σε ένα άλλο κόμβο . Όμως λόγω του ότι η μέγιστη χωρητικότητα που μπορεί να έχει μια ακμή με βάση τον σχεδιασμό της τοπολογίας αυτής είναι δέκα αυτό σημαίνει ότι ο αλγόριθμος του αθροίσματος που αναφέραμε πιο πάνω είναι μια πολύ μικρή τιμή. Αυτό που αναμένουμε λοιπόν να δούμε είναι, αύξηση στον πραγματικό χρόνο όσο μεγαλώνει ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών, με αργό και σταθερό ρυθμό.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 2.



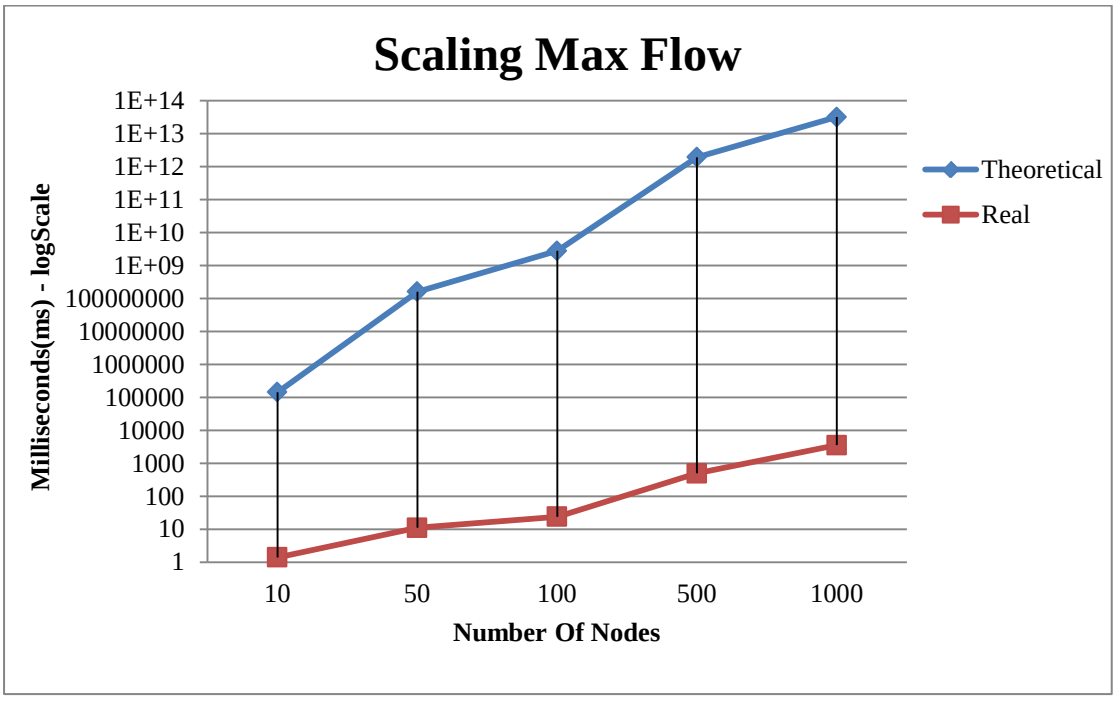
Όπως ήταν αναμενόμενο το γράφημα της πραγματικής ανάλυσης συνάδει με αυτό της θεωρητικής όπως είχαμε προβλέψει. Παρατηρούμε όμως ότι από ένα σημείο και μετά ο πραγματικός είναι καλύτερος από ότι θα έπρεπε να είναι θεωρητικά. Αυτό αρχίζει συμβαίνει εν όσο ο αριθμός των κόμβων μεγαλώνει, όπως επίσης και ο αριθμός των ακμών.

Εικάζουμε ότι αυτό παρατηρείται διότι λόγω του ότι ο γράφος είναι πολύ πυκνός με απευθείας ακμές από όλους τους κόμβους προς όλους τους άλλους. Αυτό επηρεάζει θετικά την απόδοση των αλγορίθμων που στις πλείστες περιπτώσεις θα επιλέξουν τα πιο μικρά μονοπάτια.

Τοπολογία 3

Ο χρόνος του αλγορίθμου όπως είναι γνωστό επηρεάζεται από τον αριθμό των ακμών και από τον λογάριθμο του αθροίσματος των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή και καταλήγουν σε ένα άλλο κόμβο. Σε αυτή την τοπολογία ξέρουμε ότι λόγω του σχεδιασμού ο λογάριθμος που αναφέραμε θα είναι μια επιλήψιμη τιμή και συγχρόνως μια τιμή στην ουσία αμελητέα. Ο λόγος που συμβαίνει αυτό είναι επειδή ενώ το άθροισμα του οποίου εμείς θα πάρουμε τον λογάριθμο είναι πολύ μεγάλος οπότε και ο λογάριθμος που θα προκύψει θα είναι μια τιμή που θα επηρεάσει κάπως το χρόνο, όχι όμως σε σημείο που να επηρεαστεί η συνολική εικόνα του χρόνου που χρειάζεται ο αλγόριθμος. Τελικά αναμένουμε ότι ο πραγματικός χρόνος του αλγορίθμου θα είναι πολύ παρόμοιος με τον θεωρητικό του χρόνο.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 3.



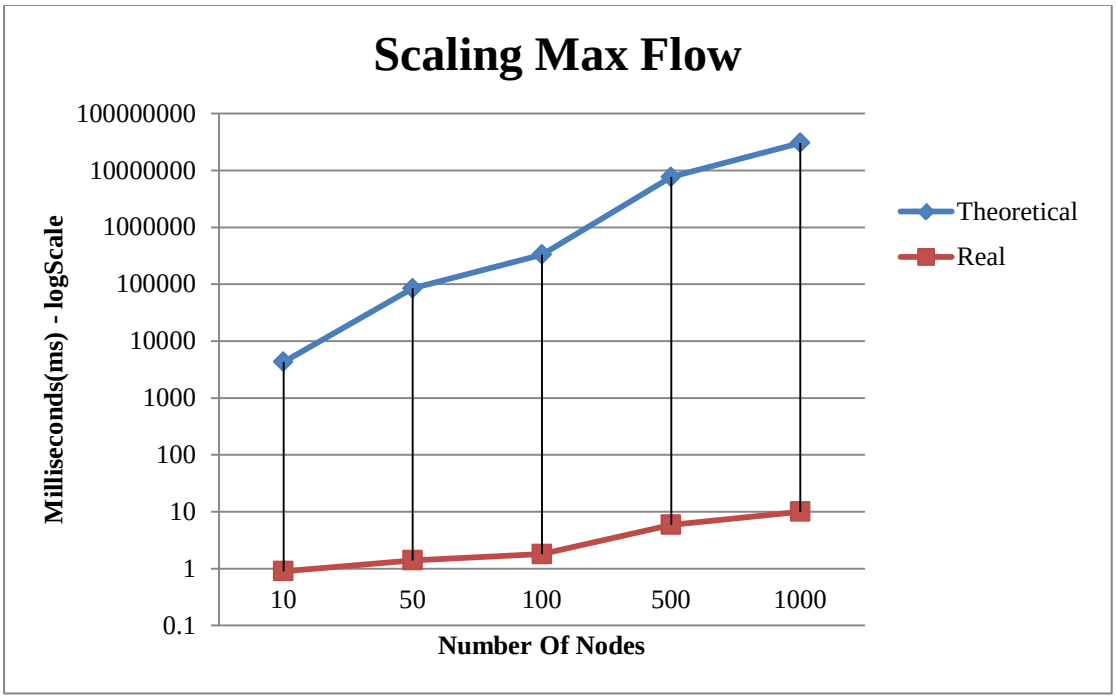
Όπως αναμέναμε τα δύο γραφήματα μοιάζουν μεταξύ τους όμως παρατηρούμε μια πιο έντονη συμπεριφορά του θεωρητικού χρόνου σε σχέση με το πραγματικό. Κυρίως για μεγάλους γράφους η διαφορά στο πόσο πιο έντονος είναι ο θεωρητικός χρόνος είναι πολύ προφανής.

Εικάζουμε ότι αυτό συμβαίνει επειδή ο αλγόριθμος Scaling Max Flow[1] στην τεχνική επιλογής των μονοπατιών που ακολουθεί αξιοποιεί την μέγιστη των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή και φτάνουν σε οποιοδήποτε άλλο κόμβο . Έτσι συμπεραίνουμε ότι επειδή σε αυτήν την τοπολογία η μέγιστη χωρητικότητα είναι ένας πολύ μεγάλος αριθμός, αυτό χρησιμεύει στον αλγόριθμο ειδικά για μεγάλο αριθμό κόμβων.

Τοπολογία 4

Ο χρόνος του αλγορίθμου αυτού εξαρτάται από τον αριθμό των ακμών και τον λογάριθμό του αθροίσματος των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή. Στην συγκεκριμένη τοπολογία ο αριθμός των ακμών είναι μικρός, λόγω του ότι οι γράφοι που δημιουργούνται είναι αραιοί. Επίσης ο λογάριθμος του αθροίσματος θα είναι μια μικρή τιμή παρ' όλο που το άθροισμα θα είναι πολύ μεγάλη τιμή λόγω του σχεδιασμού της τοπολογίας. Οπότε αυτό που αναμένουμε είναι ο πραγματικός χρόνος του αλγορίθμου για αυτή την τοπολογία να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και ακόμα αναμένουμε ότι το γράφημα του πραγματικού χρόνου θα είναι πολύ παρόμοιο με το γράφημα του θεωρητικού χρόνου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 4



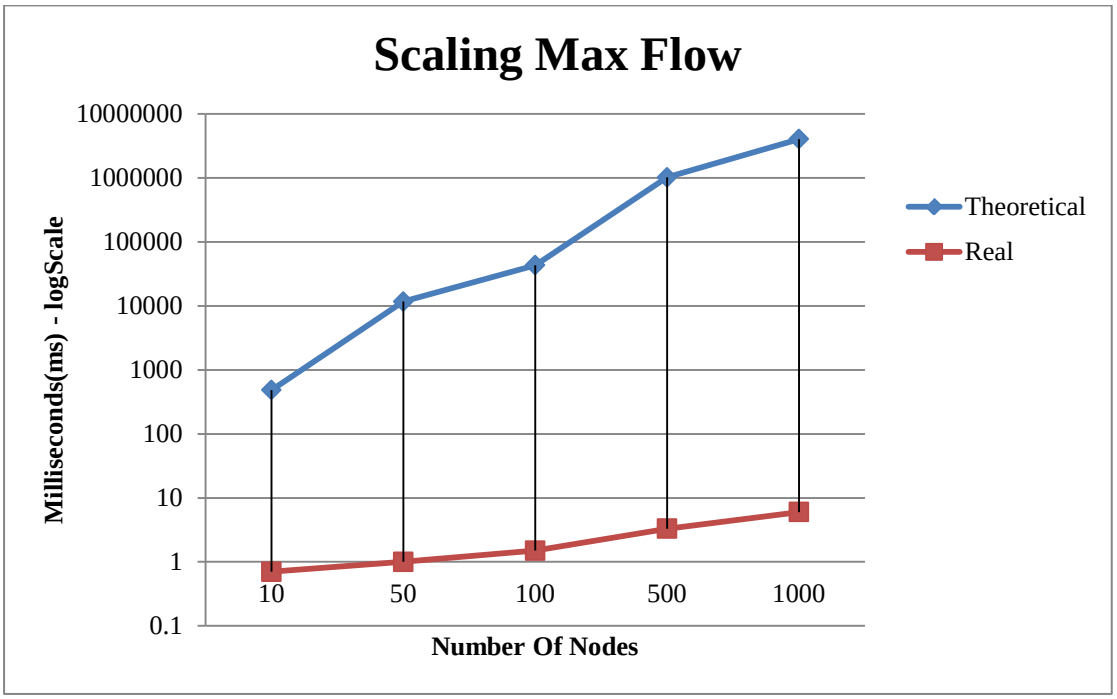
Όπως αναμέναμε ο χρόνος του πραγματικού χρόνου αυξάνεται ενώ μεγαλώνει ο αριθμός των κόμβων, δηλαδή ο αριθμός των ακμών. Ακόμα μπορούμε να παρατηρήσουμε ότι τα δύο γραφήματα είναι πολύ παρόμοια μεταξύ τους. Ο πραγματικός χρόνος όμως αυξάνεται με πιο ήπιο ρυθμό από τον θεωρητικό από ένα σημείο και μετά. Η διαφορά αυτή όμως είναι αμελητέα.

Τοπολογία 5

Ο χρόνος του αλγορίθμου Scaling Max Flow[1] εξαρτάται από τον αριθμό των ακμών και τον λογάριθμό του αθροίσματος των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή. Στην

συγκεκριμένη τοπολογία ο αριθμός των ακμών είναι μικρός, λόγω του ότι οι γράφοι που δημιουργούνται είναι αραιοί. Επίσης μικρός είναι και ο λογάριθμος του αθροίσματος διότι, το άθροισμα θα είναι μικρή τιμή, επειδή οι τιμές των χωρητικοτήτων των ακμών είναι πολύ μικρές. Οπότε αυτό που αναμένουμε είναι ο πραγματικός χρόνος του αλγορίθμου για αυτή την τοπολογία να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων. Ακόμα αναμένουμε ότι το γράφημα του πραγματικού χρόνου θα είναι πολύ παρόμοιο με το γράφημα του θεωρητικού χρόνου.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 5



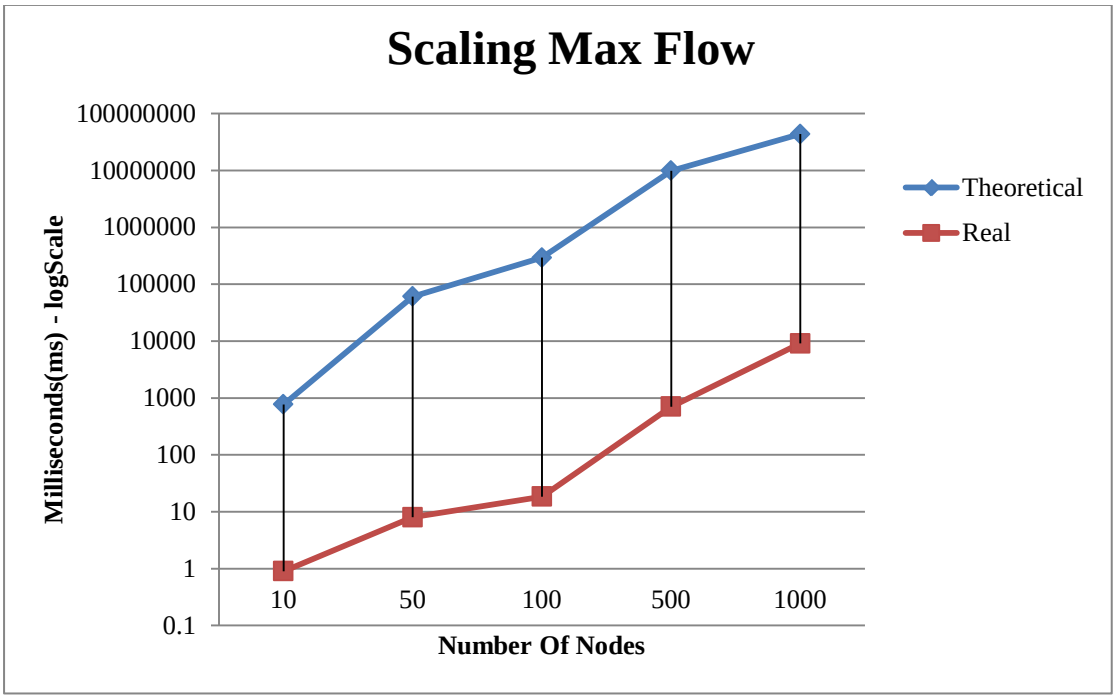
Βλέποντας την γραφική παράσταση παρατηρούμε ότι τα δύο γραφήματα που προκύπτουν με τα δεδομένα που είχαμε στην διάθεση μας σύμφωνα με την τοπολογία είναι όμοια μεταξύ τους. Η διαφορά που εντοπίζουμε είναι στο ότι ο πραγματικός χρόνος παρουσιάζει πιο ήπια συμπεριφορά από τον θεωρητικό χρόνο.

Εικάζουμε ότι αυτό συμβαίνει επειδή οι γράφοι που δημιουργούνται δεν περιέχουν πολλά μονοπάτια από τη πηγή στον προορισμό ,λόγω της τακτικής δημιουργίας των γράφων στην συγκεκριμένη τοπολογία και έτσι διευκολύνεται ακόμα περισσότερο η δουλειά του αλγορίθμου .Οπότε, ναι υπάρχουν οι ακμές στον γράφο τις οποίες εμείς υπολογίζουμε στον θεωρητικό χρόνο όμως αυτές οι ακμές δεν δημιουργούν μονοπάτια τα οποία να οδηγούν στον προορισμό, υπάρχει σίγουρα όμως ένα μονοπάτι από την πηγή στον προορισμό.

Τοπολογία 6

Οι παράμετροι από τις από τις οποίες εξαρτάται ο χρόνος του αλγορίθμου Scaling Max Flow[1] σε αυτή την τοπολογία αυξάνονται με σταθερό τρόπο όσο αυξάνεται ο αριθμός των κόμβων. Ο χρόνος του αλγορίθμου εξαρτάται από τον αριθμό των ακμών που στην Τοπολογία 6 ισούται σχεδόν με τον διπλάσιο αριθμό κόμβων που υπάρχουν στον γράφο. Επίσης εξαρτάται όπως ξέρουμε και από το λογάριθμο του αθροίσματος των χωρητικοτήτων των ακμών που προέρχονται από την πηγή που με βάση αυτήν την τοπολογία ξέρουμε ισούται σχεδόν με τον αριθμό των κόμβων. Άρα αυτό που περιμένουμε να δούμε είναι τον θεωρητικό αλλά και τον πραγματικό χρόνο να αυξάνονται όσο μεγαλώνει ο αριθμός των κόμβων.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 6



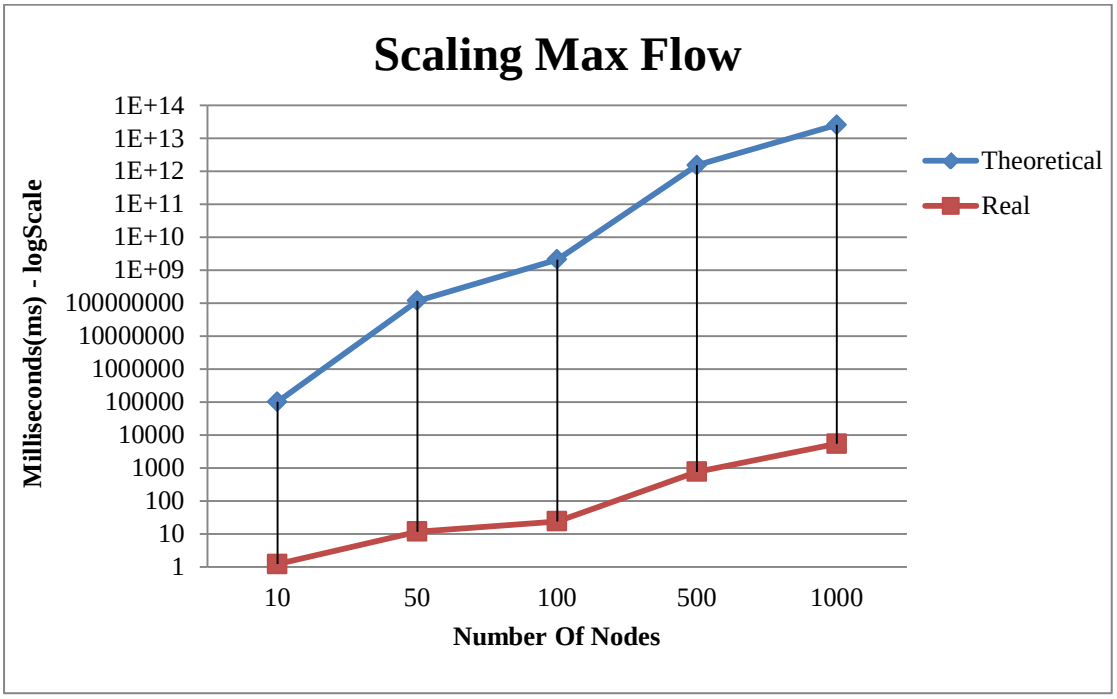
Όπως ήταν αναμενόμενο τα δύο γραφήματα είναι σχεδόν εντελώς όμοια μεταξύ τους. Επίσης ο πραγματικός χρόνος αυξάνεται όπως ακριβώς περιμέναμε.

Τοπολογία 7

Η τοπολογία 7 ίσως επηρεάσει τον χρόνο του αλγορίθμου Scaling Max Flow[1] διότι οι παράμετροι που καθορίζουν τον χρόνο του βρίσκονται στο προσκήνιο. Η μία παράμετρος είναι ο αριθμός των ακμών και η άλλη είναι όπως ξέρουμε ο λογάριθμος του αθροίσματος των χωρητικοτήτων των ακμών που προέρχονται από την πηγή. Ο αριθμός των ακμών όπως ξέρουμε είναι

αρκετά μεγάλος λόγω του ότι οι γράφοι που δημιουργούνται είναι αρκετά πυκνοί. Είναι πολύ πυκνοί γράφοι ειδικά για μικρό αριθμό κόμβων. Επίσης το άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή ενδέχεται να είναι μεγάλο διότι οι γράφοι είναι τυχαίοι και κατά συνέπεια και οι χωρητικότητες των ακμών επιλέγονται επίσης με τυχαίο τρόπο. Το σημαντικό όμως είναι ότι σε αυτή τη τοπολογία το εύρος των τιμών από το οποίο έχουν να επιλέξουν είναι πολύ μεγάλο οπότε ίσως επιλεγούν και μεγάλες τιμές και ακόμα λόγω της πυκνότητας των γράφων που προκύπτουν ίσως υπάρχουν πολλές ακμές από τη πηγή προς οποιοδήποτε άλλο κόμβο. Άρα κατά κάποιο τρόπο ο λογάριθμος αυτού του αθροίσματος θα είναι μια τιμή όχι τόσο μεγάλη διότι ο λογάριθμος ακόμα και μιας τεράστιας τιμής είναι μικρός αριθμός αλλά σίγουρα θα είναι μια επιλήψιμη τιμή. Οπότε αυτό που αναμένουμε να δούμε είναι μια ανοδική γραφική παράσταση όσο μεγαλώνει ο αριθμός των κόμβων και επίσης δύο όμοια μεταξύ τους γραφήματα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 7



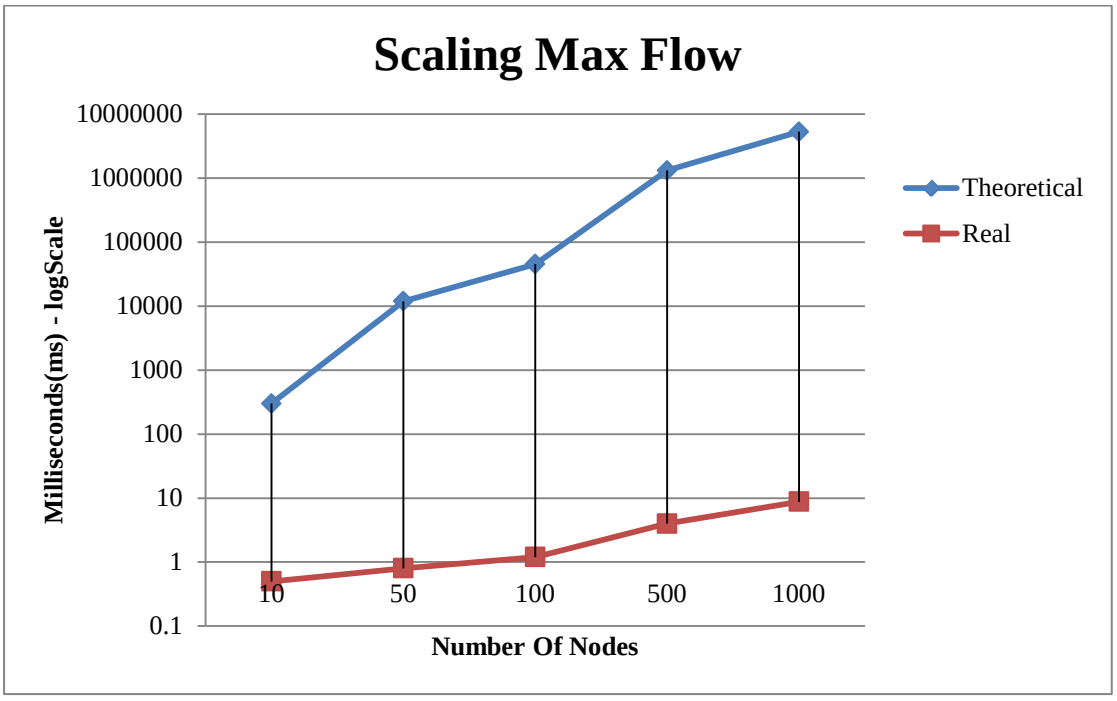
Η γραφική παράσταση που προέκυψε είναι αυτό που αναμέναμε. Βλέπουμε όμως μια διαφορά μεταξύ των δύο γραφημάτων όσο μεγαλώνει ο αριθμός των κόμβων.

Εικάζουμε ότι αυτό συμβαίνει επειδή ο αλγόριθμος αυτός όσο μεγαλώνει ο αριθμός των κόμβων σε αυτή την τοπολογία, αποβάλλει το προγραμματιστικό overhead και βελτιώνεται ο χρόνος του.

Τοπολογία 8

Ο αλγόριθμος αυτός όπως γνωρίζουμε επηρεάζεται από τον αριθμό των ακμών και από τον λογάριθμο του συνολικού αθροίσματος των χωρητικοτήτων των ακμών που προέρχονται από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην συγκεκριμένη τοπολογία όσο μεγάλος και να είναι ο γράφος ξέρουμε ότι το άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή θα έχει μια μικρή σταθερή τιμή οπότε ο λογάριθμος θα είναι μια πολύ μικρή τιμή. Επίσης ξέρουμε ότι λόγω του σχεδιασμού της τοπολογίας ο αριθμός τρων ακμών δεν είναι μεγάλος. Οπότε αυτό που αναμένουμε να δούμε είναι το πραγματικό του χρόνου να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων με σταθερό ρυθμό. Επιπλέον περιμένουμε ότι τα δύο γραφήματα ου θα προκύψουν θα είναι παρόμοια.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 8



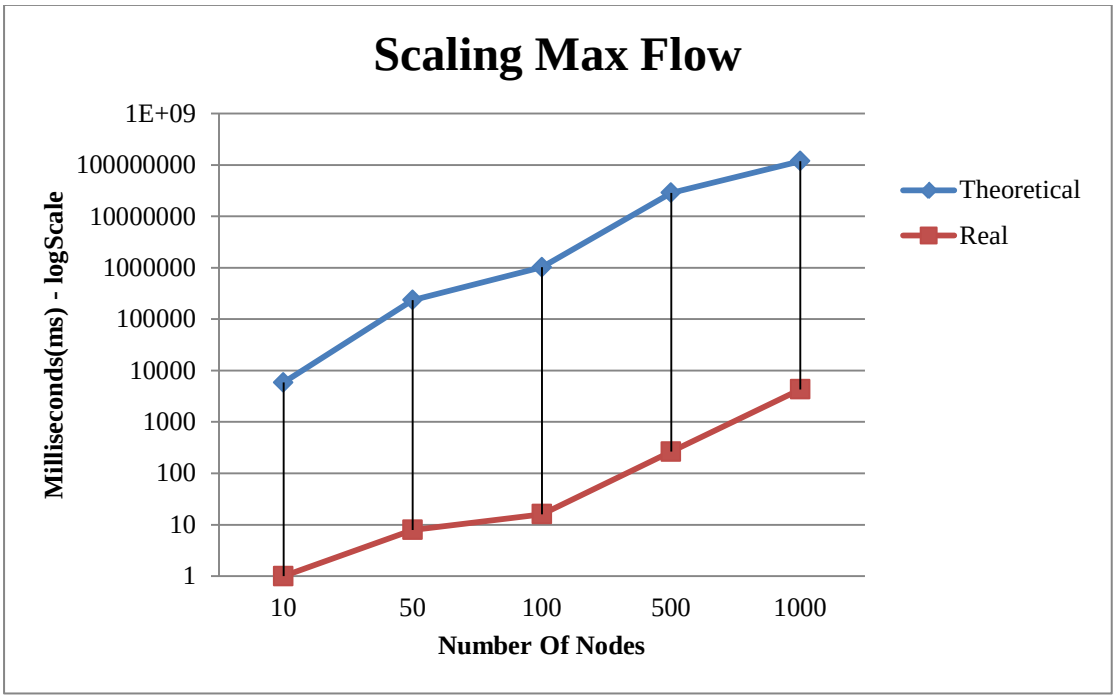
Παρατηρούμε ότι τα δύο γραφήματα που προέκυψαν είναι ίδιας συμπεριφοράς . Βλέπουμε όμως ότι υπάρχει μια διαφορά μεταξύ των δύο γραφημάτων η οποία δεν είναι σταθερή για κάθε αριθμό κόμβων.

Εικάζουμε ότι αυτό συμβαίνει επειδή λόγω του σχεδιασμού της τοπολογίας ο αλγόριθμος δεν κάνει άχρηστες επαναλήψεις. Συγκεκριμένα αναφερόμαστε στις επαναλήψεις που κάνει ο αλγόριθμος Scaling Max Flow[1](βλ. Κεφάλαιο Περιγραφής Αλγορίθμου)

Τοπολογία 9

Ο αλγόριθμος αυτός ξέρουμε ότι επηρεάζεται από τον αριθμό των ακμών και από τον λογάριθμο του συνολικού αθροίσματος των χωρητικοτήτων των ακμών που προέρχονται από την πηγή. Στην Τοπολογία 9 ο αριθμός των ακμών είναι ο διπλάσιος του αριθμού των κόμβων, για οποιοδήποτε γράφο. Επίσης το άθροισμα των χωρητικοτήτων των ακμών που προέρχονται από την πηγή αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και μάλιστα είναι μια πολύ μεγάλη τιμή. όμως όπως ξέρουμε ο λογάριθμος μιας μεγάλης τιμής είναι μια πολύ πιο μικρή τιμή. Οπότε αυτό που αναμένουμε που έχουμε στην διάθεση μας είναι τόσο ο πραγματικός όσο και ο θεωρητικός χρόνος να παρουσιάσουν ένα σταθερά ανοδικό γράφημα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Max Flow[1] στην Τοπολογία 9



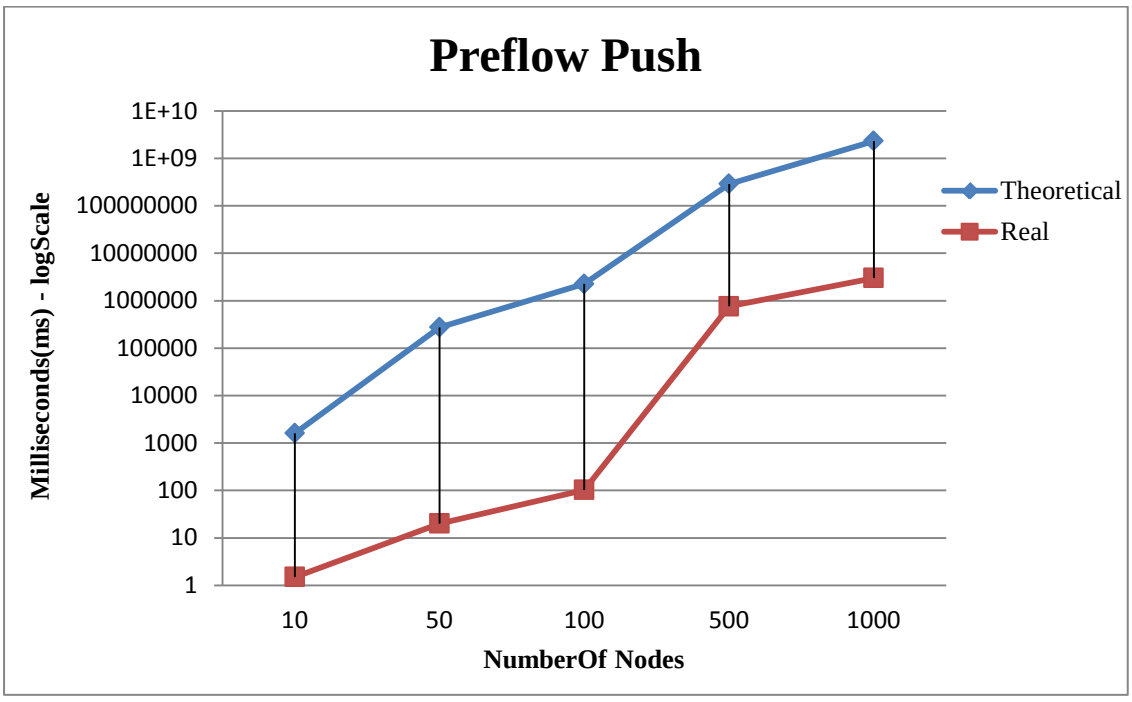
Η γραφική παράσταση που προέκυψε είναι ακριβώς αυτό που αναμέναμε έχουμε μπροστά δύο σχεδόν εντελώς ίδια γραφήματα με ανοδική πορεία.

5.4.1.4 Αλγόριθμος Preflow Push[1]

Τοπολογία 1

Η τοπολογίας αυτή λόγω του σκοπού για τον οποίο σχεδιάστηκε έχει συγκεκριμένο αριθμό ακμών για κάθε αριθμό κόμβων που δίνεται στον αλγόριθμο. Οπότε, εφόσον θεωρητικά οι μόνες παράμετροι που επηρεάζουν τον χρόνο του αλγορίθμου είναι ο αριθμός των ακμών και ο αριθμός των κόμβων αυτό που αναμένουμε είναι η συμπεριφορά της πραγματικής ανάλυσης να συνάδει με την θεωρητική ανάλυση.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push στην Τοπολογία 1.



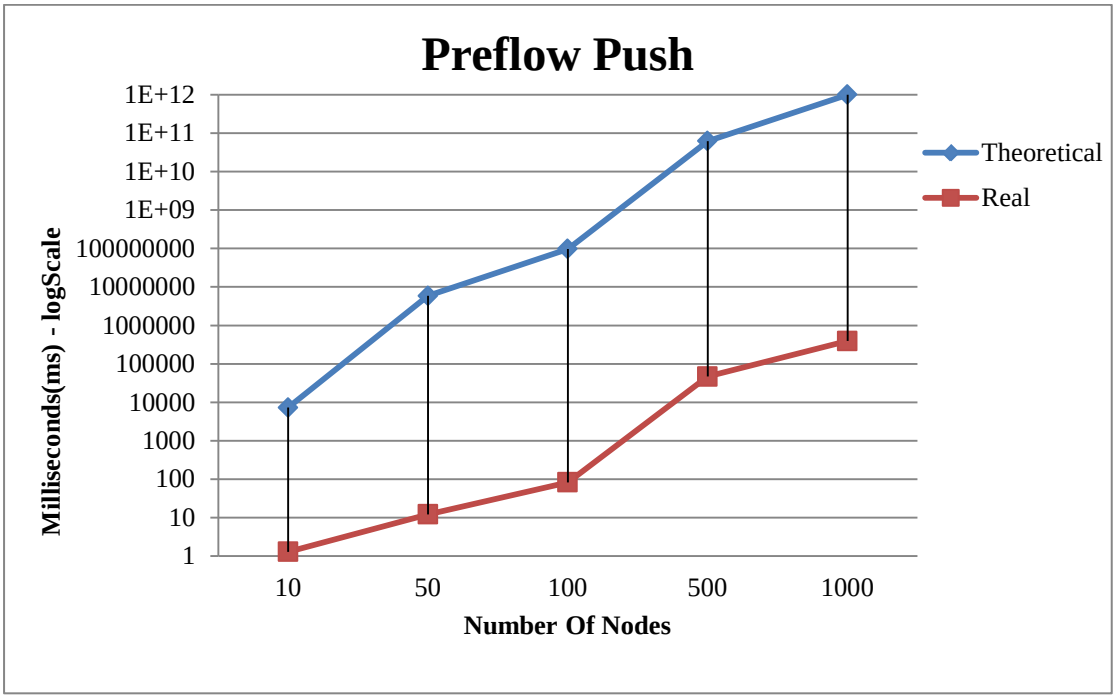
Βλέποντας τα αποτελέσματα της πιο πάνω γραφικής παράστασης παρατηρούμε ότι δεν είναι αυτά που αναμέναμε. Ενώ αρχικά η θεωρητική και η πραγματική συμπεριφορά του αλγορίθμου είναι παρόμοιες στην συνέχεια όσο μεγαλώνει ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών ο πραγματικός χρόνος του αλγορίθμου τείνει ακόμα και να πλησιάσει τον θεωρητικό πράγμα το οποίο δεν αναμέναμε να δούμε.

Εικάζουμε ότι αυτό συνέβηκε επειδή ο αλγόριθμος Preflow Push[1] παρουσιάζει ένα επιπλέον overhead στην υλοποίηση του λόγω των πολλών υποχρεωτικών ελέγχων που πραγματοποιεί . Έτσι υποθέτουμε ότι ο θεωρητικά σταθερός χρόνος που χρειάζεται ο αλγόριθμος για να υπολογίσει Μέγιστη Ροή στου γράφους που δημιουργούνται με βάση την Τοπολογία 1 συν το επιπλέον overhead είναι αυτά που δημιουργούν την πιο πάνω γραφική και την κάνουν να δείχνει μια διαφορετική συμπεριφορά σε σχέση με τη θεωρητική ανάλυση.

Τοπολογία 2

Στον χρόνο του αλγορίθμου αυτού σημαντικό ρόλο παίζει ο αριθμός των ακμών και ο αριθμός των κόμβων. Οπότε περιμένουμε ότι ο πραγματικός χρόνος του αλγορίθμου αυξάνεται σε σχέση με των αριθμό των κόμβων που σχετίζεται άμεσα και με τον αριθμό των ακμών. Αναμένουμε να δούμε λοιπόν δύο παρόμοια γραφήματα διότι πιστεύουμε ότι δεν υπάρχουν άλλοι παράμετροι που θα επηρεάσουν τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος για να βρει την Μέγιστη Ροή σε ένα γράφο.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 2



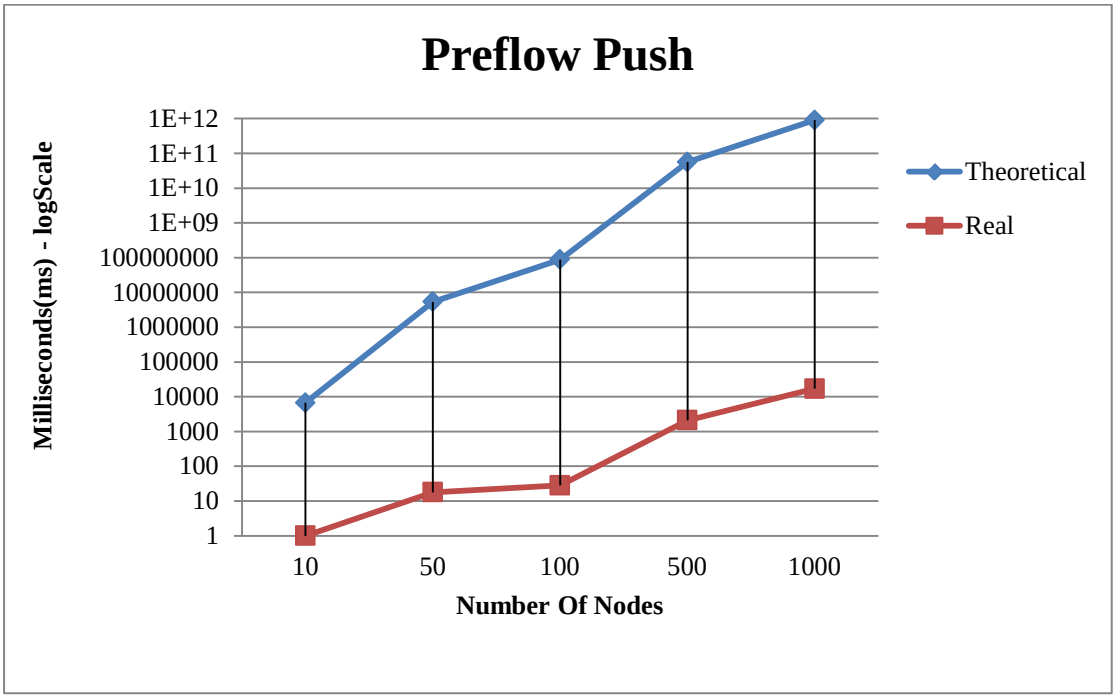
Όπως είναι ξεκάθαρο στην γραφική παράσταση βλέπουμε δύο παρόμοια γραφήματα με τον πραγματικό χρόνο αυξάνεται με σταθερό ρυθμό και να ακολουθεί τον θεωρητικό χρόνο που χρειάζεται ο αλγόριθμος για να επιλύσει προβλήματα που δημιουργούνται με βάση την Τοπολογία 2. Παρατηρούμε όμως ότι αρχικά για μικρό αριθμό κόμβων η διαφορά μεταξύ των δύο γραφημάτων είναι μικρότερη και στην συνέχεια μεγαλώνει και σταθεροποιείται η μεταξύ τους διαφορά.

Εικάζουμε ότι ο λόγος που αρχικά παρουσιάζεται αυτό το φαινόμενο είναι ότι ο αλγόριθμος αυτός παρουσιάζει κάποιο επιπλέον overhead το οποίο για μικρούς γράφους και με βάση την τοπολογία είναι σημαντικό και φαίνεται ότι επηρεάζει το συνολικό χρόνο του αλγορίθμου . Ενώ για μεγάλους γράφους λόγω της τοπολογίας που είναι σχετικά απλή αφού υπάρχει ακμή από όλους τους κόμβους, προς όλους τους άλλους και κατά συνέπεια και πολλά μικρά μονοπάτια το overhead αυτό εξαλείφεται και είναι ασήμαντο ως προς τον συνολικό χρόνο.

Τοπολογία 3

Ο χρόνος του αλγορίθμου αυτού επηρεάζεται από τον αριθμό των ακμών και των αριθμό των κόμβων. Στην συγκεκριμένη τοπολογία λόγω του σχεδιασμού της δημιουργούνται αρκετά πυκνοί γράφοι. Ειδικά για μικρό αριθμό κόμβων οι γράφοι που δημιουργούνται είναι σχεδόν όλοι προς όλους. Οπότε αυτό που αναμένουμε είναι ότι η πυκνότητα των γράφων θα επηρεάσει την συμπεριφορά των γραφημάτων κάνοντας την πιο έντονη. Επίσης αναμένουμε ότι ο πραγματικός χρόνος θα αυξάνεται σε σχέση με την αύξηση του αριθμού των κόμβων. Τελικά αυτό που περιμένουμε να δούμε είναι δύο αρκετά όμοια γραφήματα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 3

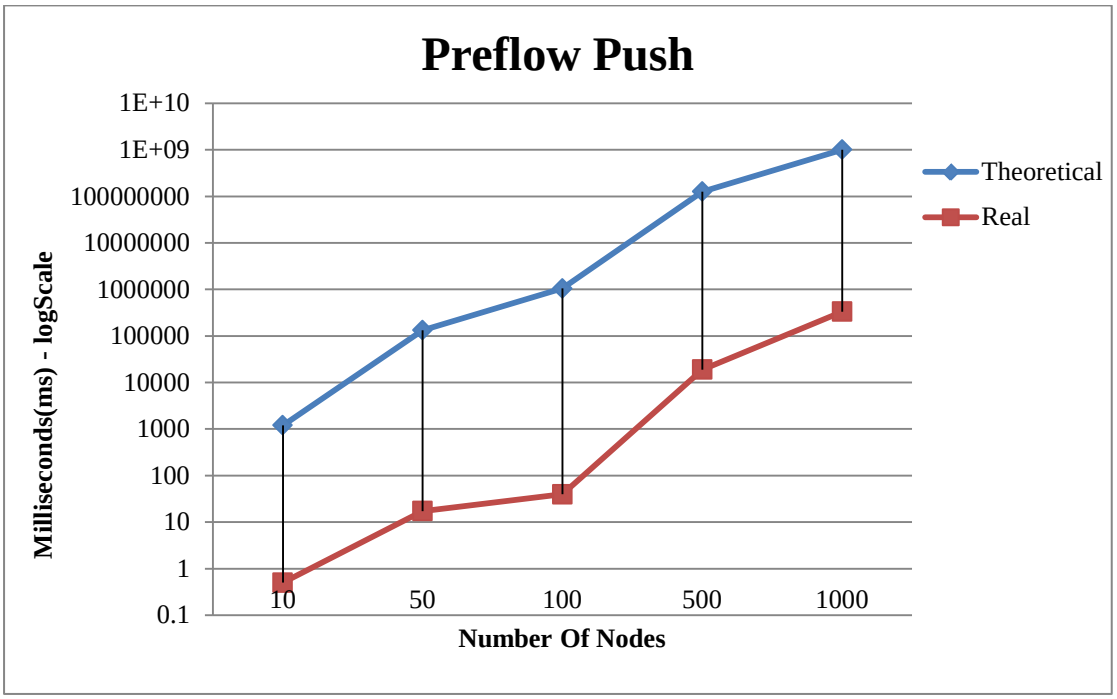


Η γραφική παράσταση που προέκυψε είναι σχεδόν αυτό που αναμέναμε. Τα δύο γραφήματα είναι παρόμοια με τη διαφορά ότι η συμπεριφορά του αλγορίθμου στην πράξη είναι λίγο καλύτερη από ότι θα έπρεπε να είναι θεωρητικά. Η διαφορά αυτή όμως δεν είναι επιλήψιμη διότι είναι μικρή και ίσως οφείλεται και στο ότι ο σχεδιασμός των γράφων γίνεται με τυχαίο τρόπο, με βάση την Τοπολογία 3 .

Τοπολογία 4

Θεωρητικά ο χρόνος του αλγορίθμου Preflow Push[1] επηρεάζεται από δύο παραμέτρους, των αριθμό των κόμβων και των αριθμό των ακμών. Στην συγκεκριμένη τοπολογία έχουμε μικρό αριθμό ακμών διότι δημιουργούνται αραιοί, συνεκτικοί γράφοι. Οπότε αυτό που αναμένουμε είναι να δούμε τον πραγματικό χρόνο να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων με σταθερό ρυθμό. Επίσης περιμένουμε να δούμε δύο παρόμοια γραφήματα όσον αφορά τον πραγματικό και τον θεωρητικό χρόνο διότι δεν υπάρχει κάποια παράμετρος που να επηρεάζει τον πραγματικό χρόνο ώστε να τον διαφοροποιεί σε σχέση με τον θεωρητικό.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 4

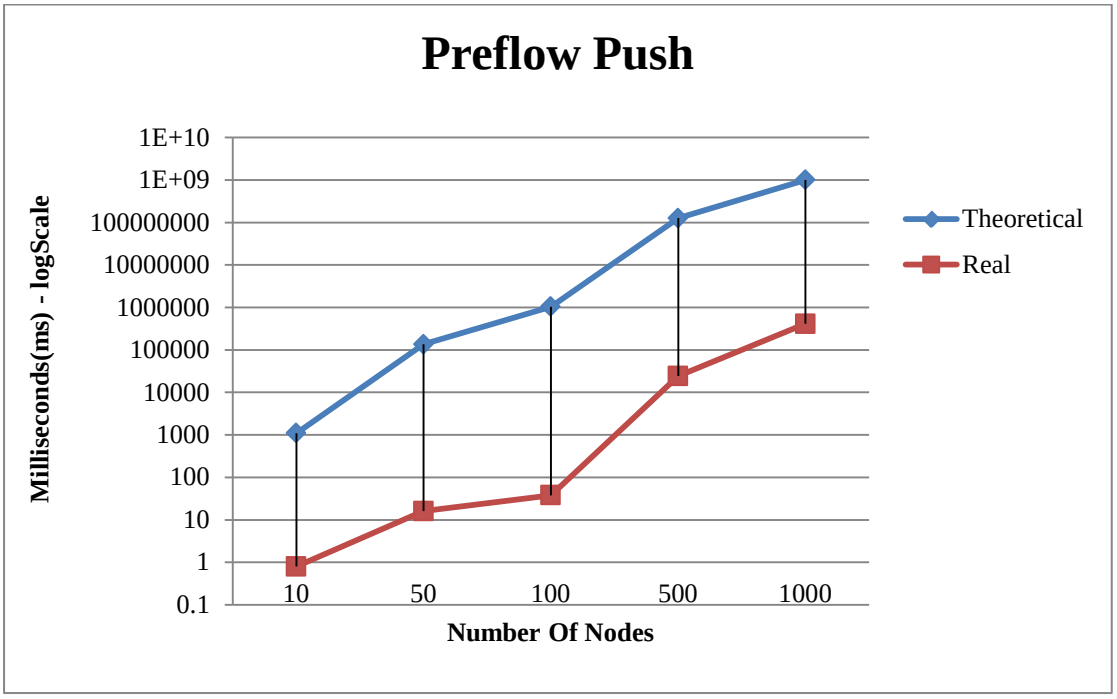


Όπως βλέπουμε η γραφική παράσταση είναι ακριβώς αυτό που αναμέναμε. Τα δύο γραφήματα είναι εντελώς ίδια μεταξύ τους. Αυτό σημαίνει ότι ο αλγόριθμος συμπεριφέρεται όπως ακριβώς ήταν αναμενόμενο.

Τοπολογία 5

Θεωρητικά ο χρόνος του αλγορίθμου Preflow Push[1] επηρεάζεται από δύο παραμέτρους, των αριθμό των κόμβων και των αριθμό των ακμών. Στην συγκεκριμένη τοπολογία έχουμε μικρό αριθμό ακμών διότι δημιουργούνται αραιοί, συνεκτικοί γράφοι. Οπότε αυτό που αναμένουμε είναι να δούμε τον πραγματικό χρόνο να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων με σταθερό ρυθμό. Επίσης περιμένουμε να δούμε δύο παρόμοια γραφήματα όσον αφορά τον πραγματικό και τον θεωρητικό χρόνο διότι δεν υπάρχει κάποια παράμετρος που να επηρεάζει τον πραγματικό χρόνο ώστε να τον διαφοροποιεί σε σχέση με τον θεωρητικό.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 5

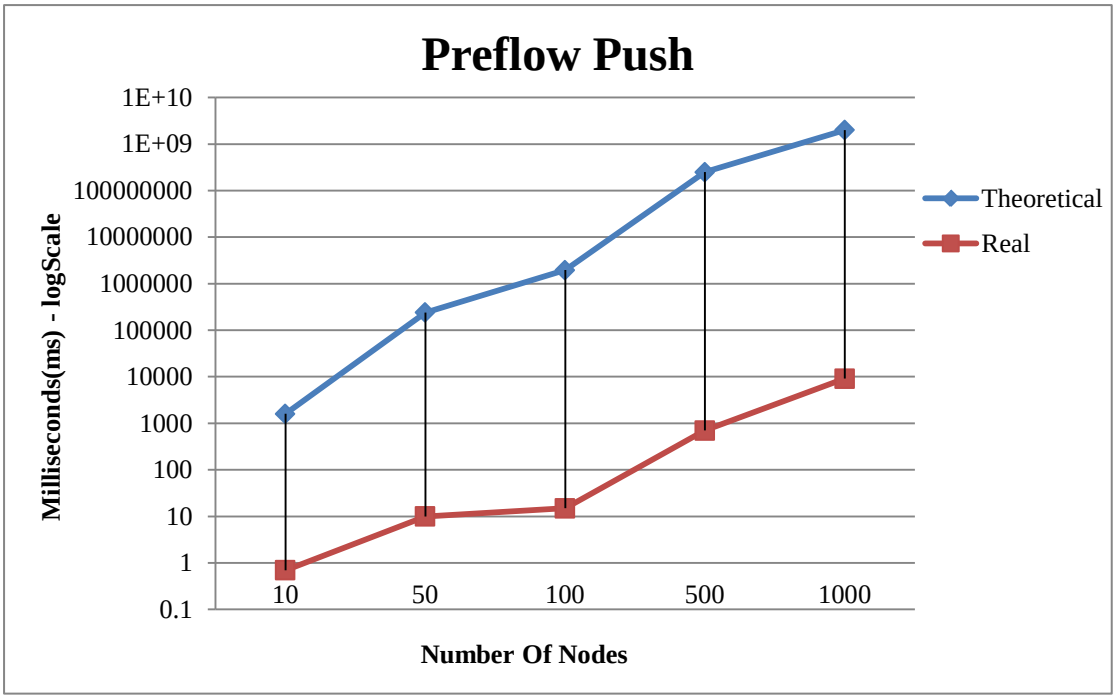


Όπως βλέπουμε η γραφική παράσταση είναι ακριβώς αυτό που αναμέναμε. Τα δύο γραφήματα είναι εντελώς ίδια μεταξύ τους. Αυτό σημαίνει ότι ο αλγόριθμος συμπεριφέρεται όπως ακριβώς ήταν αναμενόμενο.

Τοπολογία 6

Η παράμετροι που επηρεάζουν τον αλγόριθμο αυτό είναι όπως γνωρίζουμε ο αριθμός των κόμβων και ο αριθμός των ακμών που υπάρχουν στον γράφο. Στην Τοπολογία 6 ο αριθμός των ακμών είναι σχετικά μικρός διότι με βάση τον σχεδιασμό της τοπολογίας ισούται περίπου με το διπλάσιο του αριθμού των κόμβων που υπάρχουν σε ένα γράφο. Οπότε αυτό που αναμένουμε να δούμε το πραγματικό χρόνο να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και να προκύπτει ένας γράφημα παρόμοιας συμπεριφοράς με το θεωρητικό.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 6

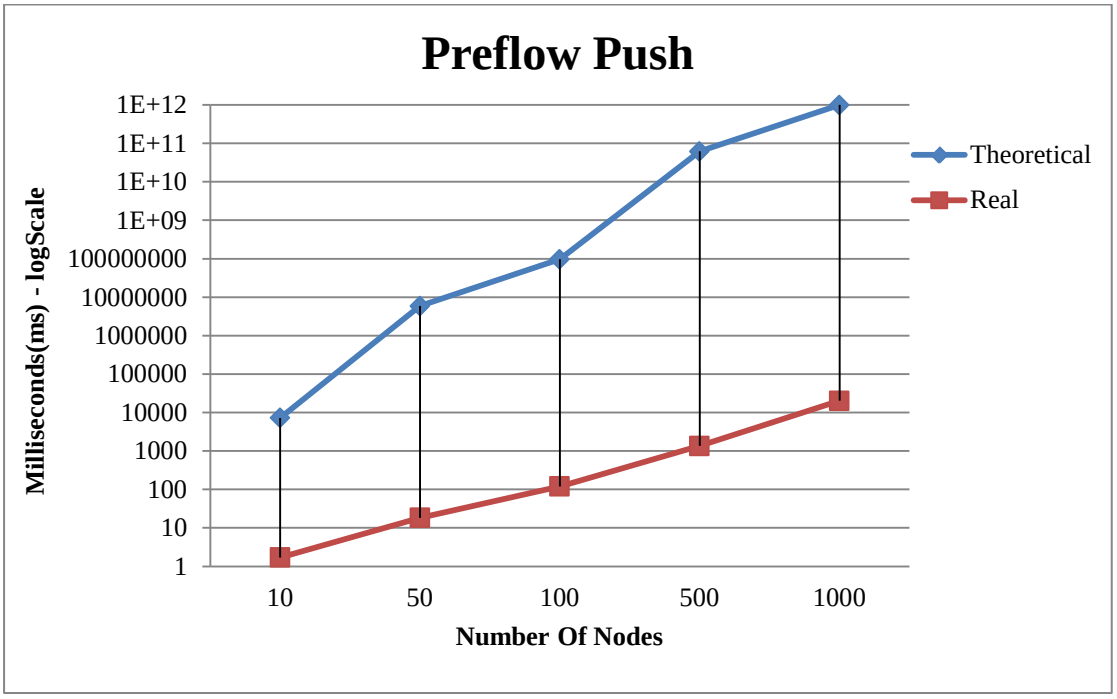


Στην πιο πάνω γραφική βλέπουμε να συμβαίνει αυτό ακριβώς που προβλέψαμε. Η συμπεριφορά του αλγόριθμου, βάση των αποτελεσμάτων που προέκυψαν από τα πειράματα μας, είναι ακριβώς ίδια με τη θεωρητική του ανάλυση.

Τοπολογία 7

Ο χρόνος του αλγορίθμου αυτού είναι άμεσα επηρεαζόμενος από τον αριθμό των κόμβων όπως επίσης και από το αριθμό των ακμών όπως ήδη ξέρουμε. Αυτό που αναμένουμε είναι να δούμε μια σταθερή αύξηση του χρόνου όσο μεγαλώνει ο αριθμός των κόμβων και κατ' επέκταση και ο αριθμός των ακμών. Επίσης υπολογίζουμε ότι η θεωρητική ανάλυση και η πραγματική ανάλυση του χρόνου του αλγορίθμου θα είναι όμοιας συμπεριφοράς.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 7

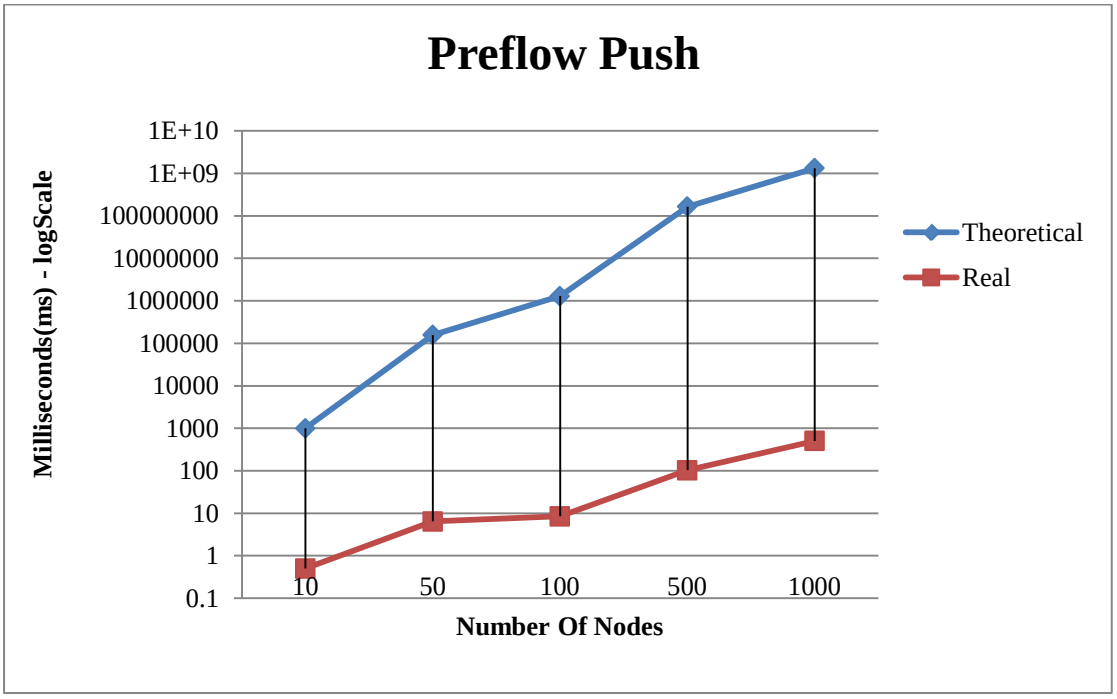


Η γραφική παράσταση που προέκυψε, βάση των αποτελεσμάτων που εξασφαλίσαμε από τα πειράματα μας αξιολογώντας τον αλγόριθμο Preflow Push[1], πάνω σε γράφους που προέκυψαν από την Τοπολογία 7, είναι ακριβώς αυτό που αναμέναμε. Το θεωρητικό γράφημα παρουσιάζει μια πιο έντονη συμπεριφορά αλλά αυτό είναι φυσικό.

Τοπολογία 8

Γνωρίζοντας ότι ο χρόνος του αλγορίθμου αυτού εξαρτάται κυρίως από τον αριθμό των κόμβων και λιγότερο από τον αριθμό των ακμών συμπεραίνουμε ότι αφού εδώ ο αριθμός των ακμών αυξάνεται σταθερά όσο αυξάνεται ο αριθμός των κόμβων θα δούμε ένα γράφημα υο οποίο θα παρουσιάζει μια σταθερή ανοδική πορεία. Επίσης αναμένουμε ότι τα δύο γραφήματα θα είναι όμοια μεταξύ τους.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 8



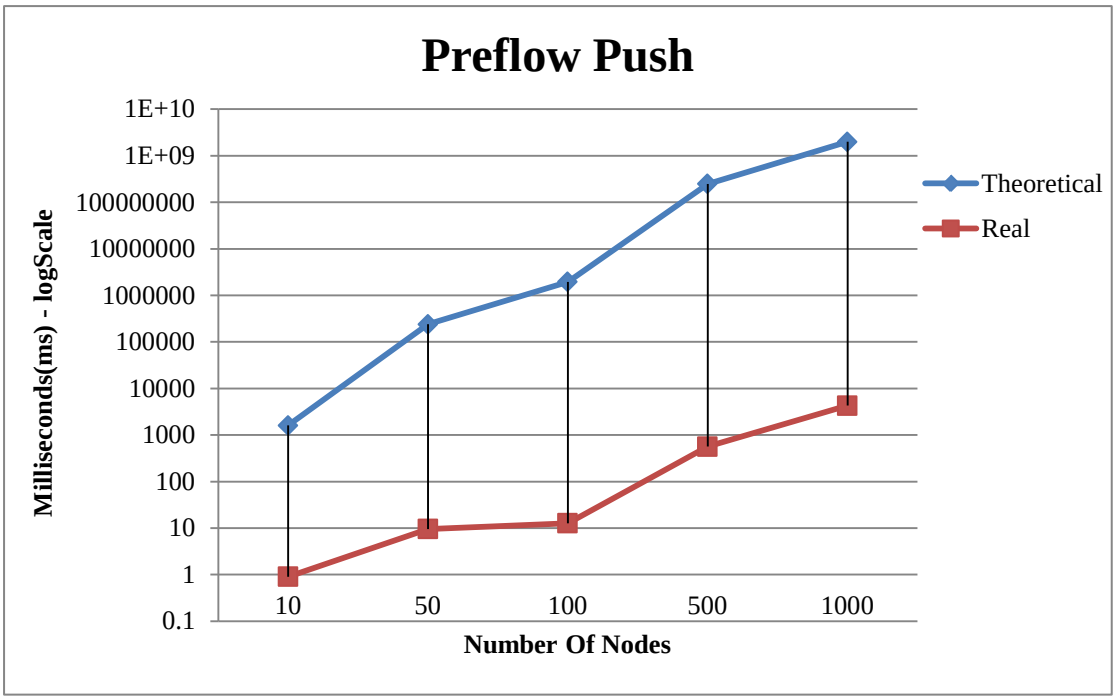
Η γραφική παράσταση που προέκυψε, βάση των αποτελεσμάτων που εξασφαλίσαμε από τα πειράματα μας αξιολογώντας τον αλγόριθμο Preflow Push[1], πάνω σε γράφους που προέκυψαν από την Τοπολογία 8, είναι ακριβώς αυτό που αναμέναμε. Μεταξύ των δύο γραφημάτων υπάρχει μια μικρή διαφορά όσο μεγαλώνει ο αριθμός των κόμβων.

Εικάζουμε ότι αυτό συμβαίνει λόγω του ότι οι γράφοι που δημιουργούνται είναι σχετικά απλοί δεν έχουν πολύπλοκα μονοπάτια τα οποία θα αύξαναν τον αριθμό των ελέγχων που θα έκανε ο αλγόριθμος.

Τοπολογία 9

Ξέροντας ότι ο χρόνος του αλγορίθμου αυτού εξαρτάται κυρίως από τον αριθμό των κόμβων και λιγότερο από τον αριθμό των ακμών συμπεραίνουμε ότι αφού εδώ ο αριθμός των ακμών αυξάνεται σταθερά όσο αυξάνεται ο αριθμός των κόμβων θα δούμε ένα γράφημα υο οποίο θα παρουσιάζει μια σταθερή ανοδική πορεία. Ξέρουμε εκ των προτέρων λόγω του σχεδιασμού της τοπολογίας ότι ο αριθμός των ακμών είναι ο διπλάσιος αριθμός των κόμβων που υπάρχουν στον γράφο. Επίσης αναμένουμε ότι τα δύο γραφήματα θα είναι όμοια μεταξύ τους.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Preflow Push[1] στην Τοπολογία 9



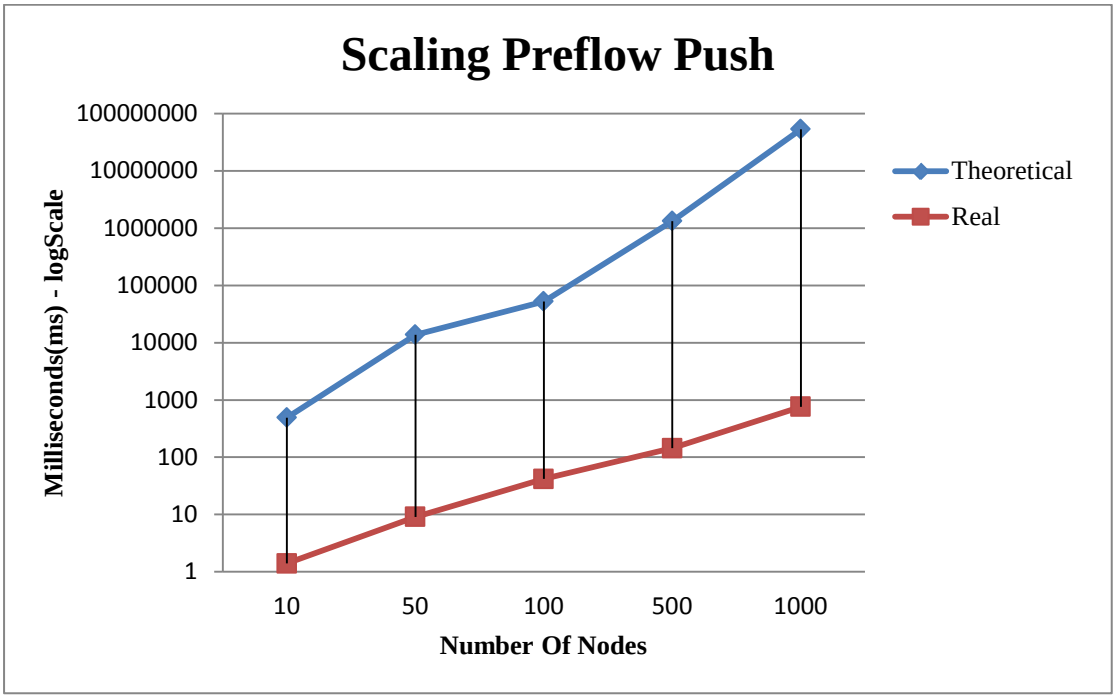
Η γραφική παράσταση είναι ακριβώς αυτό που αναμέναμε. Βλέπουμε δύο σχεδόν εντελώς ίδια γραφήματα με ανοδική πορεία.

5.4.1.5 Αλγόριθμος Scaling Preflow Push[2]

Τοπολογία 1

Αυτό που αναμένουμε είναι ο πραγματικός χρόνος του αλγορίθμου είναι να έχει ακριβώς ίδια συμπεριφορά με την θεωρητική ανάλυση του χρόνου που χρειάζεται ο αλγόριθμος Scaling Preflow Push[2] για να επιλύσει το πρόβλημα της Εύρεσης Μέγιστης Ροής σε ένα γράφο που δημιουργείται με βάση την Τοπολογία 1. Ο λόγος που αναμένουμε κάτι τέτοιο είναι επειδή ο χρόνος του αλγορίθμου επηρεάζεται άμεσα από τρείς παραμέτρους οι οποίες σε αυτήν τη τοπολογία δεν αποτελούν στόχο. Ο θεωρητικός λοιπόν χρόνος του αλγορίθμου εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι σχετικά μικρός. Επιπρόσθετα μικρός είναι και ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή εφόσον η μεγαλύτερη δύναμη που να είναι όμως και μικρότερη από την μέγιστη χωρητικότητα είναι οκτώ. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων και κατά συνέπεια και ο αριθμός των ακμών με τον ίδιο ρυθμό που θα αυξάνεται και στο θεωρητικό γράφημα.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 1.



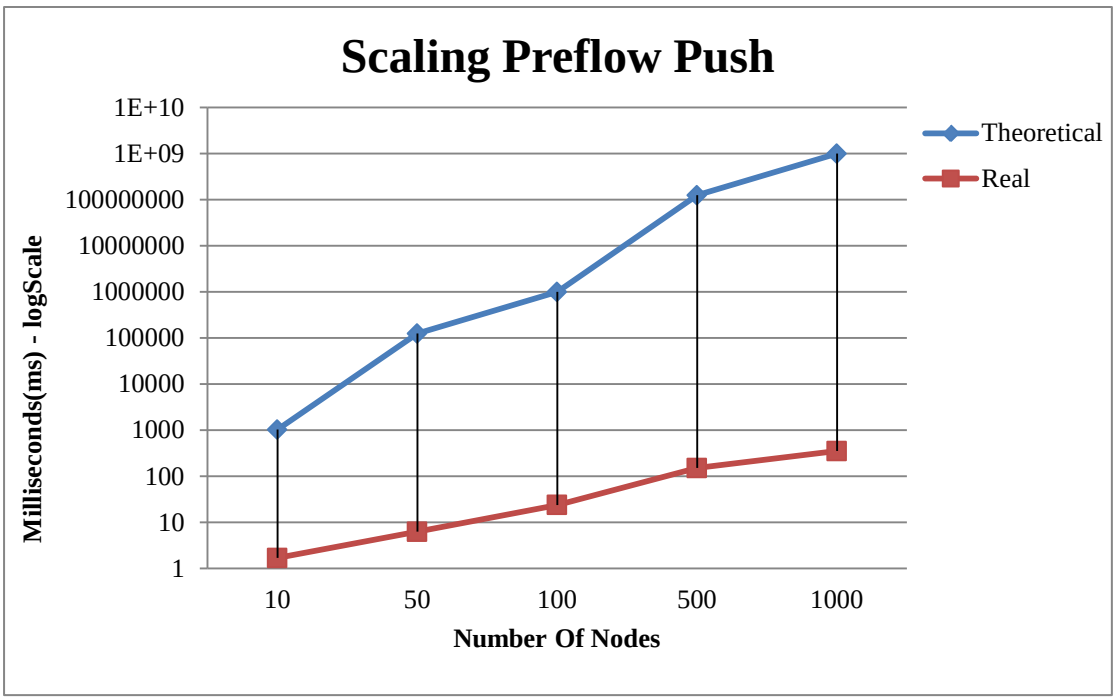
Όπως αναμέναμε η πραγματική ανάλυση είναι ακριβώς ίδιας συμπεριφοράς με την θεωρητική ανάλυση και ακόμα καλύτερη , κυρίως όσο ο αριθμός των κόμβων μεγαλώνει και αυτό

ίσως συμβαίνει λόγω του ότι η υλοποίηση του αλγορίθμου του δίνει το προβάδισμα της εξάλειψης ελέγχων και σπατάλης χρόνου.

Τοπολογία 2

Ο χρόνος του αλγορίθμου Scaling Preflow Push[2] επηρεάζεται άμεσα από τρείς παραμέτρους. Ο θεωρητικός λοιπόν χρόνος του αλγορίθμου εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι μεγάλος. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή εφόσον η μεγαλύτερη δύναμη που είναι όμως και μικρότερη από την μέγιστη χωρητικότητα είναι οκτώ. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων με αρκετά έντονο ρυθμό εφόσον ο αριθμός των ακμών είναι αρκετά μεγάλος για κάθε αριθμό κόμβων. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 2.



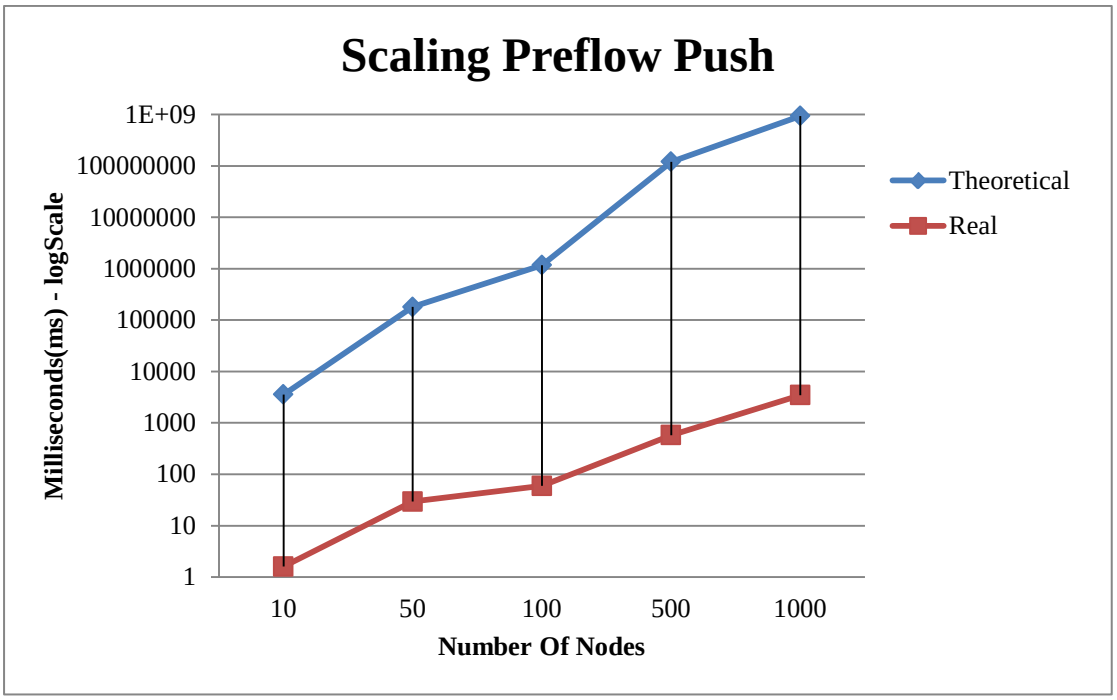
Παρατηρώντας τη γραφική παράσταση που πρόκυψε βλέπουμε ότι τα δύο γραφήματα είναι πολύ παρόμοια όμως δεν προέκυψε ακριβώς αυτό που αναμέναμε. Δηλαδή δεν παρατηρούμε έντονη συμπεριφορά στο γράφημα του πραγματικού χρόνου.

Εικάζουμε ότι αυτό είναι πιθανό να συμβαίνει για δύο λόγους. Ο ένας λόγος για το οποίο συμβαίνει αυτό είναι λόγω της τοπολογίας η οποία κατά κάποιο τρόπο λόγω των ευέλικτων σύντομων μονοπατιών διευκολύνει την δουλειά του αλγορίθμου στην πραγματικότητα αντί να επηρεάζει το χρόνο του λόγω του μεγάλου αριθμού ακμών. Ο άλλος λόγος είναι ίσως επειδή ο αλγόριθμος αυτός λόγω της υλοποίησης του εξαλείφει το προγραμματιστικό overhead και έτσι τα συστατικά που απαιτεί η υλοποίηση του δεν επιβαρύνουν το συνολικό πραγματικό του χρόνο.

Τοπολογία 3

Ο θεωρητικός χρόνος του αλγορίθμου εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι αρκετά μεγάλος. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή. Μπορεί η μεγαλύτερη δύναμη του δύο που είναι όμως και μικρότερη από την μέγιστη χωρητικότητα να είναι μεγάλος αριθμός όμως ο λογάριθμος είναι μια μικρή τιμή. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 3.



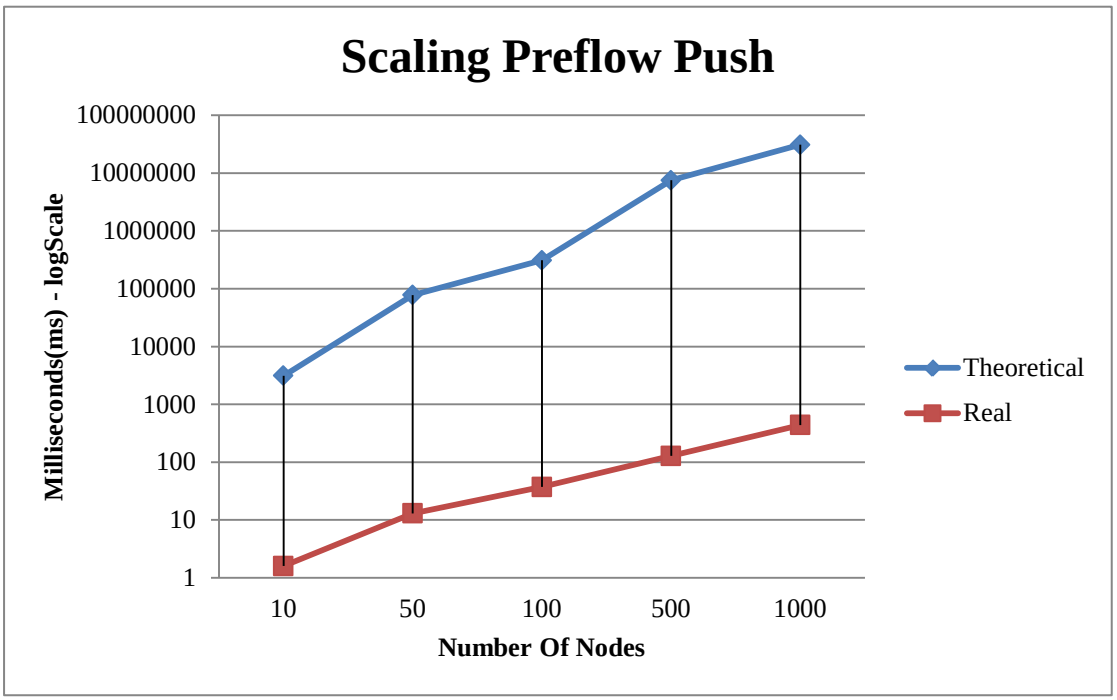
Παρατηρώντας τη γραφική παράσταση που πρόκυψε βλέπουμε ότι τα δύο γραφήματα είναι πολύ παρόμοια όμως δεν προέκυψε ακριβώς αυτό που αναμέναμε. Δηλαδή δεν παρατηρούμε έντονη συμπεριφορά στο γράφημα του πραγματικού χρόνου.

Εικάζουμε ότι αυτό είναι πιθανό να συμβαίνει για δύο λόγους. Ο ένας λόγος για το οποίο συμβαίνει αυτό είναι λόγω της τοπολογίας η οποία κατά κάποιο τρόπο λόγω των ευέλικτων σύντομων μονοπατιών διευκολύνει την δουλειά του αλγορίθμου στην πραγματικότητα αντί να επηρεάζει το χρόνο του λόγω του μεγάλου αριθμού ακμών. Ο άλλος λόγος είναι ίσως επειδή ο αλγόριθμος αυτός λόγω της υλοποίησης του εξαλείφει το προγραμματιστικό overhead και έτσι τα συστατικά που απαιτεί η υλοποίηση του δεν επιβαρύνουν το συνολικό πραγματικό του χρόνο.

Τοπολογία 4

Ο θεωρητικός χρόνος του αλγορίθμου εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι πολύ μικρός διότι οι γράφοι που δημιουργούνται στην τοπολογία αυτή είναι αραιοί. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή. Μπορεί η μεγαλύτερη δύναμη του δύο που είναι όμως και μικρότερη από την μέγιστη χωρητικότητα να είναι μεγάλος αριθμός όμως ο λογάριθμος είναι μια μικρή τιμή. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων με αρκετά έντονο ρυθμό εφόσον ο αριθμός των ακμών είναι αρκετά μεγάλος για κάθε αριθμό κόμβων. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 4

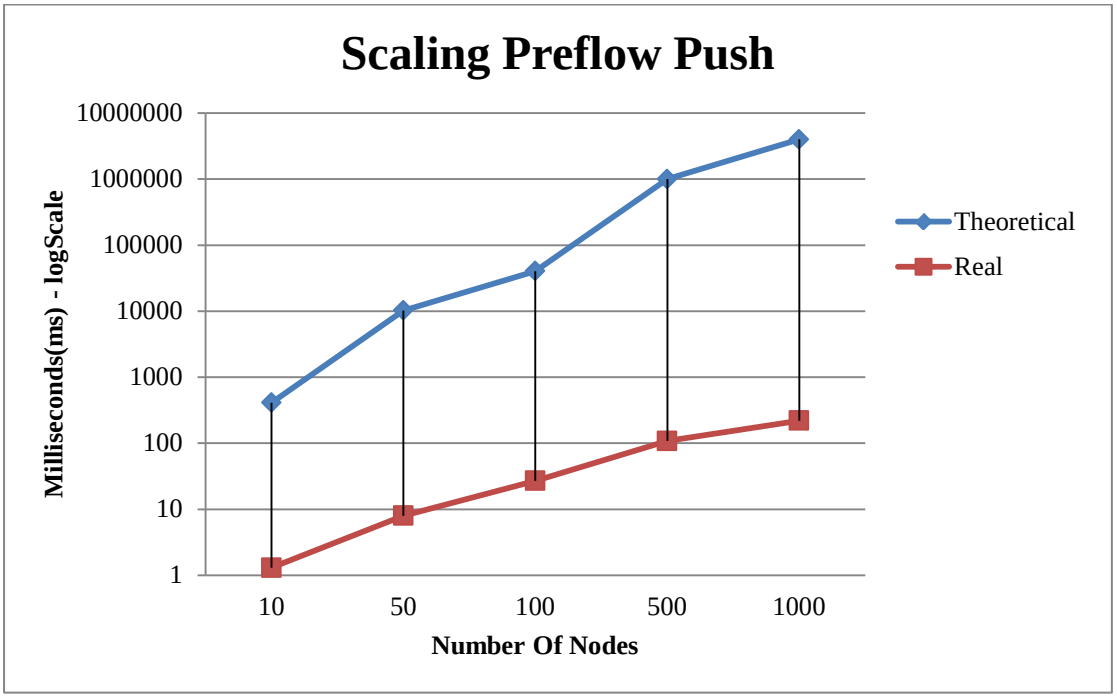


Παρατηρώντας την γραφική παράσταση βλέπουμε ότι είναι ακριβώς αυτό που αναμέναμε. Βλέπουμε να απεικονίζονται δύο πολύ παρόμοια μεταξύ τους γραφήματα ,όπως επίσης βλέπουμε ότι η συμπεριφορά του θεωρητικού χρόνου είναι πιο έντονή από τον χρόνο του πραγματικού χρόνου όμως αυτό είναι λογικό διότι το παρατηρούμε για όλους τους αριθμούς κόμβων.

Τοπολογία 5

Ο χρόνος του αλγορίθμου Scaling Preflow Push[2] επηρεάζεται άμεσα από τρείς παραμέτρους, τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι πολύ μικρός διότι οι γράφοι που δημιουργούνται στην τοπολογία αυτή είναι αραιοί. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή, επειδή οι τιμές των χωρητικοτήτων στους γράφους που δημιουργούνται με βάση την Τοπολογία 5 είναι πολύ μικρές. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 5

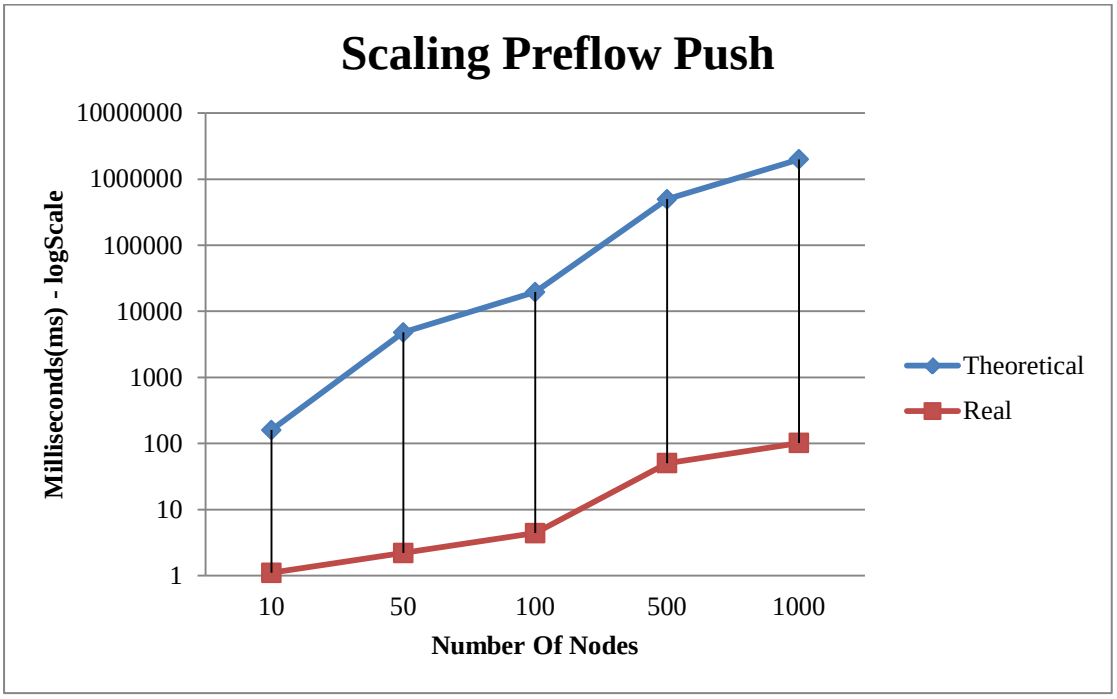


Παρατηρώντας την γραφική παράσταση βλέπουμε ότι είναι ακριβώς αυτό που αναμέναμε. Βλέπουμε να απεικονίζονται δύο πολύ παρόμοια μεταξύ τους γραφήματα ,όπως επίσης βλέπουμε ότι η συμπεριφορά του θεωρητικού χρόνου είναι πιο έντονή από τον χρόνο του πραγματικού χρόνου όμως αυτό είναι λογικό και το παρατηρούμε για όλους τους αριθμούς κόμβων.

Τοπολογία 6

Όπως ξέρουμε θεωρητικά ο χρόνος του αλγορίθμου Scaling Preflow Push[2] εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι σχετικά μικρός διότι ξέρουμε ότι ο αριθμός των ακμών ισούται με το διπλάσιο του αριθμού των κόμβων που υπάρχουν στο γράφο. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή επειδή όλες οι ακμές των γράφων που δημιουργούνται με βάση την Τοπολογία 6 ισούται με ένα. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 6

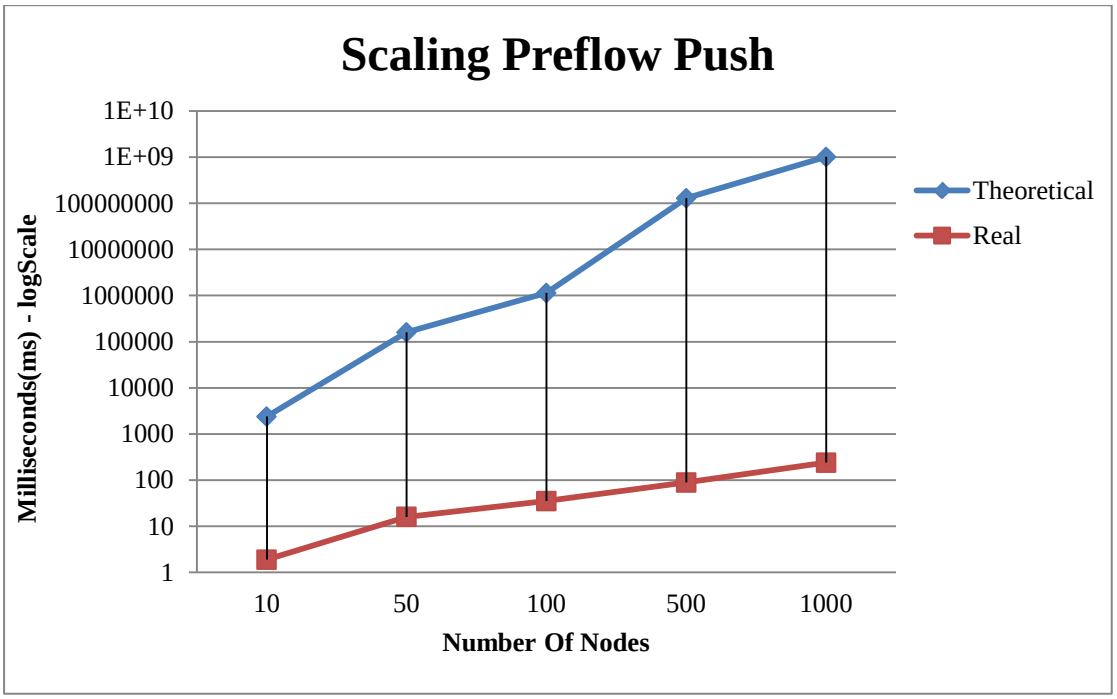


Όπως μπορούμε να παρατηρήσουμε πράγματι τα δύο γραφήματα είναι μεταξύ τους όμοια με το γράφημα του πραγματικού χρόνου να είναι σχετικά λίγο καλύτερο όσο μεγαλώνει ο αριθμός των κόμβων από ότι θα έπρεπε να είναι θεωρητικά. Όμως η διαφορά αυτή είναι αμελητέα.

Τοπολογία 7

Γνωρίζουμε ότι θεωρητικά ο χρόνος του αλγορίθμου Scaling Preflow Push[2] εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι μεγάλος διότι ξέρουμε ότι οι γράφοι που δημιουργούνται με βάση την Τοπολογία 7 είναι σχετικά πυκνοί. Όμως ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή, διότι μπορεί να έχουμε και πολύ μεγάλες χωρητικότητες λόγω του εύρους των τιμών που μπορούν να επιλεγούν σε αυτή την τοπολογία, αλλά ο λογάριθμος θα είναι πάλι μια πολύ μικρή τιμή. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων, κυρίως λόγω της πυκνότητας του γράφου. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 7



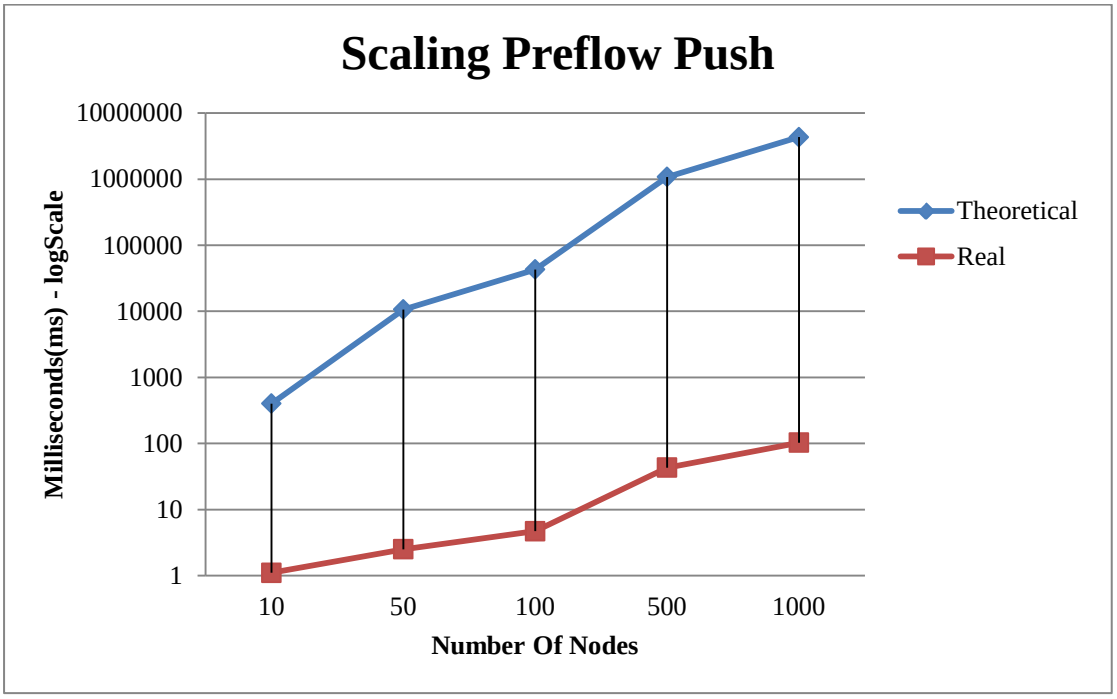
Η γραφική παράσταση που προέκυψε είναι αυτό που αναμέναμε. Βλέπουμε όμως μια διαφορά μεταξύ των δύο γραφημάτων όσο μεγαλώνει ο αριθμός των κόμβων.

Εικάζουμε ότι αυτό συμβαίνει επειδή ο αλγόριθμος αυτός όσο μεγαλώνει ο αριθμός των κόμβων σε αυτή την τοπολογία, αποβάλλει το προγραμματιστικό overhead και βελτιώνεται ο χρόνος του.

Τοπολογία 8

Όπως ξέρουμε θεωρητικά ο χρόνος του αλγορίθμου Scaling Preflow Push[2] εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι κανονικός , δηλαδή ούτε μεγάλος , ούτε μικρός και επίσης είναι προκαθορισμένος ανάλογα με τον αριθμό των κόμβων. Επίσης ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή. Εφόσον η μέγιστη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή, είναι μια μικρή τιμή και είναι ίδια για όλους αριθμούς κόμβων. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 8

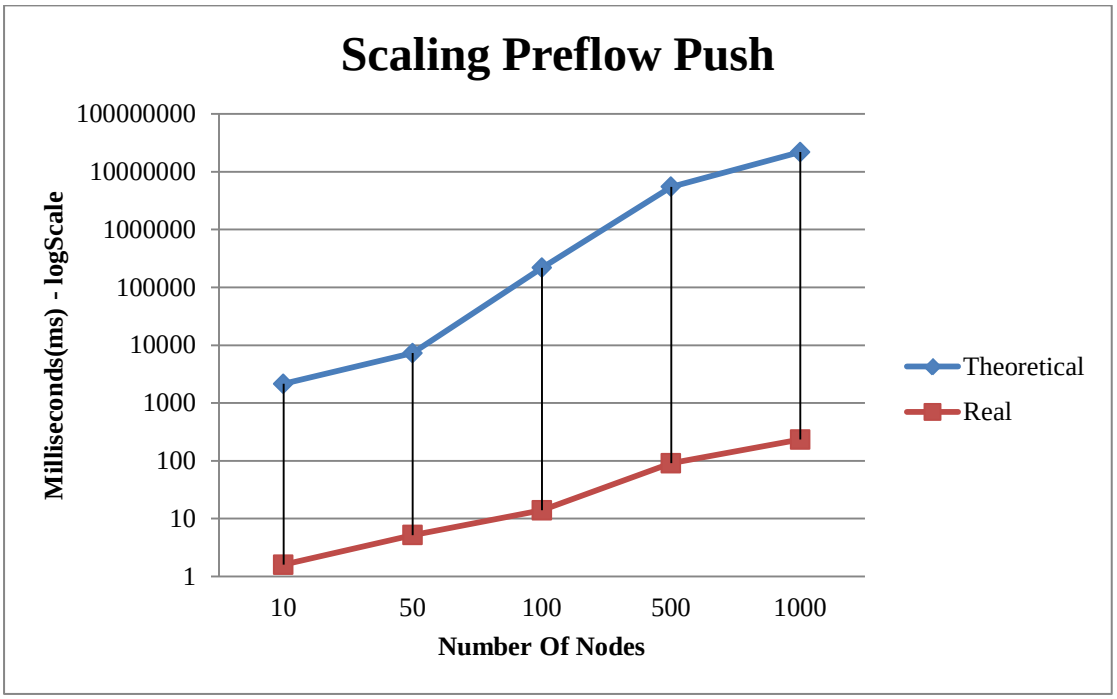


Όπως μπορούμε να παρατηρήσουμε πράγματι τα δύο γραφήματα είναι μεταξύ τους όμοια με το γράφημα του πραγματικού χρόνου να είναι σχετικά λίγο καλύτερο όσο μεγαλώνει ο αριθμός των κόμβων από ότι θα έπρεπε να είναι θεωρητικά. Όμως η διαφορά αυτή σταθεροποιείται και είναι αμελητέα.

Τοπολογία 9

Ξέρουμε ότι θεωρητικά ο χρόνος του αλγορίθμου Scaling Preflow Push[2] εξαρτάται από τον αριθμό των ακμών, των αριθμό των κόμβων και επίσης από το λογάριθμο της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο. Στην τοπολογία αυτή ο αριθμός των ακμών για κάθε αριθμό κόμβων είναι κανονικός , δηλαδή ούτε μεγάλος , ούτε μικρός και επίσης είναι προκαθορισμένος ανάλογα με τον αριθμό των κόμβων, διότι ακριβώς ο διπλάσιος. Επίσης ο λογάριθμος της μεγαλύτερης δύναμης του δύο είναι μικρή τιμή. Παρόλο που η μέγιστη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή, είναι μια πολύ μεγάλη τιμή και είναι ίδια για όλους αριθμούς κόμβων ο λογάριθμος αυτής της τιμής είναι πολύ μικρός αριθμός. Οπότε περιμένουμε ότι ο χρόνος θα μεγαλώνει όσο αυξάνεται ο αριθμός των κόμβων. Αναμένουμε λοιπόν ότι το γράφημα για τον πραγματικό χρόνο που χρειάζεται ο αλγόριθμος θα είναι πολύ παρόμοιο με το γράφημα που παρουσιάζει ο θεωρητικός χρόνος.

Η γραφική που προέκυψε με βάση τη αξιολόγηση του αλγορίθμου Scaling Preflow Push[2] στην Τοπολογία 9



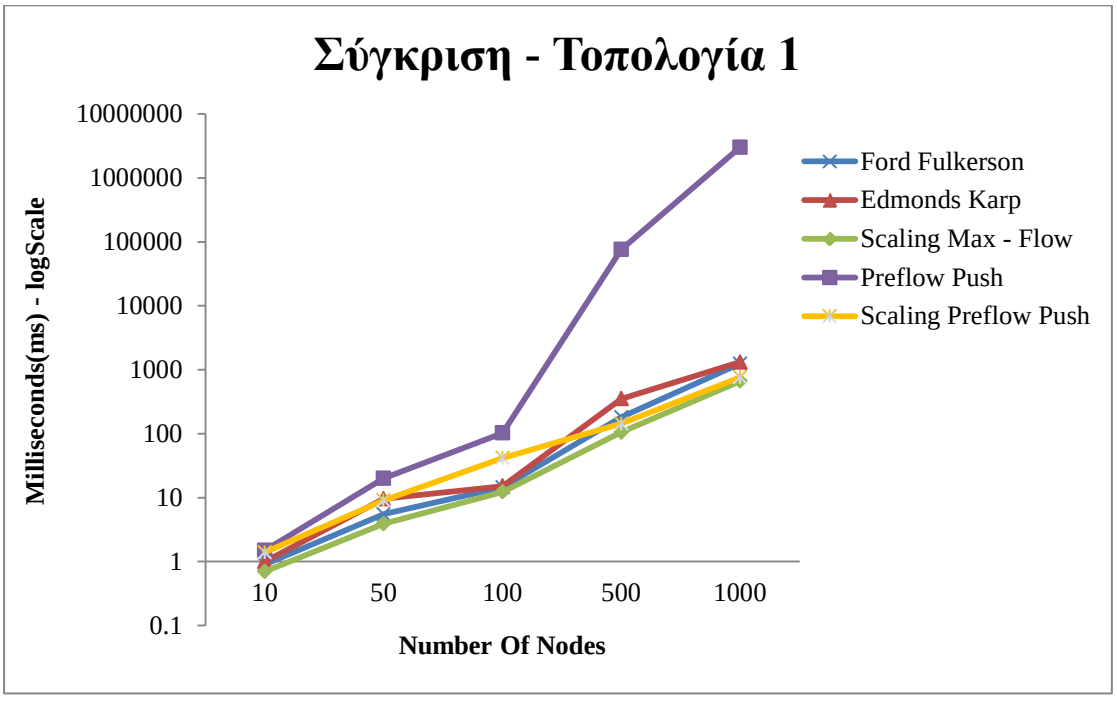
Όπως μπορούμε να παρατηρήσουμε πράγματι τα δύο γραφήματα είναι μεταξύ τους όμοια με το γράφημα του πραγματικού χρόνου να είναι σχετικά λίγο καλύτερο όσο μεγαλώνει ο αριθμός των κόμβων από ότι θα έπρεπε να είναι θεωρητικά. Όμως η διαφορά αυτή σταθεροποιείται και είναι αμελητέα.

5.4.2 Συγκριτικά Αποτελέσματα

Τοπολογία 1

Στην τοπολογία αυτή, στόχος είναι να αποδυναμώσουμε τον αλγόριθμο Edmonds Karp[3] με βάση τον τρόπο που επιλέγει τα μονοπάτια διαμέσου των οποίων θα αποστείλει ροή από την πηγή, στον προορισμό. Στοχεύουμε λοιπόν στο ότι ο αλγόριθμος Edmonds Karp[3] επιλέγει τα πιο σύντομα μονοπάτια πρώτα για να στείλει ροή και στην συνέχεια περνά ροή διαμέσου άλλων μονοπατιών οπότε κάνουμε τα σύντομα αυτά μονοπάτια στενά και τα πιο μακρινά μονοπάτια με κανονικές χωρητικότητες, τόσο όσο για να χωράει η ροή που προέρχεται από την πηγή. Άρα εφόσον ο αριθμός μέγιστης χωρητικότητας είναι μικρός και ο αριθμός των ακμών σταθερός, αναμένουμε ότι από ένα σημείο και μετά ο αλγόριθμος Edmonds Karp[3] θα είναι πιο αργός από τους υπόλοιπους διότι ο τρόπος επιλογής των μονοπατιών θα επηρεάσει πολύ τον χρόνο του σε συνδυασμό βέβαια και με τον αριθμό των ακμών. Κρατήσαμε τις παραμέτρους που δύναται να επηρεάσουν τους χρόνους των υπόλοιπων αλγορίθμων σταθερές ώστε να είναι φανερή η επιρροή του τρόπου επιλογής των μονοπατιών στον αλγόριθμο Edmonds Karp[3]. Αυτό που αναμένουμε να δούμε τελικά είναι τον πραγματικό χρόνο του αλγορίθμου Edmonds Karp[3] να αυξάνεται όσο μεγαλώνει ο αριθμός των κόμβων και να ξεπερνά τον χρόνο όλων των άλλων αλγορίθμων.

Γραφική Παράσταση:



Συμπερασματικά αυτό που παρατηρούμε στην γραφική παράσταση στην οποία προέκυψε είναι ότι όντως ο χρόνος του αλγορίθμου Edmonds Karp[3] ειδικά από ένα σημείο και μετά αρχίζει να ξεπερνά τους χρόνους των υπολοίπων αλγορίθμων εκτός από τον χρόνο του αλγορίθμου Preflow

Push[1]. Επίσης ένα άλλο σημείο το οποίο είναι άξιο σχολιασμού είναι η συμπεριφορά του αλγορίθμου Scaling Preflow Push[2] ο οποίος ενώ αρχικά ξεπερνά σχεδόν όλους τους άλλους στην συνέχεια όσο μεγαλώνει ο αριθμός των κόμβων γίνεται όλο και καλύτερος.

Στηριζόμενη λοιπόν στις γνώσεις μου και στην μελέτη που έκανα για τον κάθε αλγόριθμο ξεχωριστά υποθέτω ότι ο λόγος που αλγόριθμος Preflow Push[1] παρουσιάζει την συμπεριφορά που είδαμε πιο πάνω λόγω του προγραμματιστικού overhead που χρειάζεται ούτως ώστε να γίνονται σωστά οι έλεγχοι που απαιτούνται για να γίνει διοχέτευση ροής από την πηγή και να φτάσει στον προορισμό. Έτσι θεωρώ ότι ο λόγος που ο χρόνος εκτινάσσεται από ένα σημείο και μετά είναι ο μεγάλος αριθμός ελέγχων που γίνονται οι οποίοι θεωρητικά είναι μηδαμινού χρόνου όμως τελικά επηρεάζουν τον συνολικό χρόνο του αλγορίθμου.

Όσον αφορά τώρα την συμπεριφορά του αλγορίθμου Scaling Preflow Push[2] εικάζω ότι οφείλεται στο ότι ο αλγόριθμος αυτός εκμεταλλεύεται την τιμή της μέγιστης χωρητικότητας που υπάρχει στον γράφο και ανήκει σε ακμή που προέρχεται από την πηγή. Έτσι επειδή όσο μεγαλώνει ο αριθμός των κόμβων μεγαλώνει και η τιμή αυτή, ο αλγόριθμος αυτός βελτιώνει τον χρόνο του σε σχέση με τους υπόλοιπους. Οπότε εξαλείφοντας ένα ικανοποιητικό μέρος του overhead του εκμεταλλευόμενος την παράμετρο αυτή, μειώνει τους ελέγχους που χρειάζεται να κάνει και καθίσταται καλύτερος από τους άλλους αλγορίθμους.

Παρατηρούμε επίσης ότι ο αλγόριθμος Scaling Max Flow[1] είναι συνεχώς καλύτερος από τον Ford Fulkerson[1]και αυτό εικάζω ότι συμβαίνει επειδή ο Scaling Max Flow[1][1] αξιοποιεί την παράμετρο που εκμεταλλεύεται και ο Scaling Preflow Push[2] και έτσι είναι καλύτερος από τον Ford Fulkerson. Όμως δεν γίνεται καλύτερος από τον Scaling Preflow Push[2] διότι συνεχίζει να ακολουθεί μια τεχνική με αρκετούς ελέγχους.

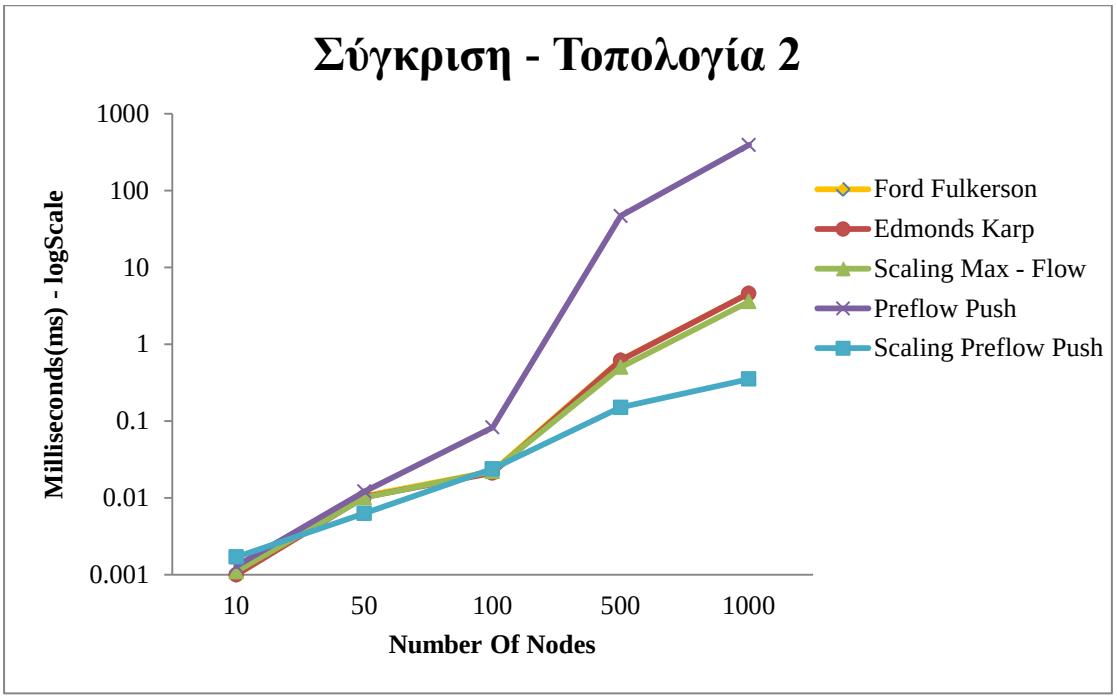
Ακόμα βλέποντας τη γραφική παράσταση που προέκυψε είναι εμφανές ότι ο χρόνος του αλγορίθμου Edmonds Karp[3] είναι για οποιοδήποτε αριθμό μεγαλύτερος από τον χρόνο των αλγορίθμων Ford Fulkerson[1]και Scaling Max Flow[1][1] και αυτό ήταν αναμενόμενο.

Τοπολογία 2

Στη τοπολογία αυτή γίνονται στόχος οι αλγόριθμοι που ο θεωρητικός τους χρόνος εξαρτάται από τον αριθμό των ακμών. Ο λόγος είναι επειδή, οι γράφοι που δημιουργούνται με βάση τον σχεδιασμό της τοπολογίας αυτής είναι πολύ πυκνοί σχεδόν όλοι με όλους ειδικά για μικρό αριθμό κόμβων αναμένουμε ότι ο χρόνος των αλγορίθμων κυρίως της οικογένειας του Ford Fulkerson[1]θα είναι μεγαλύτερος. Η μέγιστη χωρητικότητα που μπορεί να δοθεί σε μια ακμή παρόλο που η τοποθέτηση χωρητικοτήτων στις ακμές είναι τυχαία θα είναι μια μικρή τιμή. Άρα αφού δεν έχουμε άλλες παραμέτρους που μπορεί να επηρεάσουν τον χρόνο των αλγορίθμων αναμένουμε ότι ο πιο

αργός από όλους ,θεωρητικά μιλώντας πρέπει να είναι ο χρόνος του αλγορίθμου Edmonds Karp[3] βασιζόμενοι στο ότι ο χρόνος του εξαρτάται από το τετράγωνο του αριθμού των ακμών.

Γραφική Παράσταση:



Βλέποντας την γραφική παράσταση αυτό που παρατηρούμε είναι ότι ο χρόνος του αλγορίθμου Preflow Push[1] ξεπερνά κατά πολύ τον χρόνο των υπόλοιπων αλγορίθμων. Υποθέτω ότι αυτό συμβαίνει επειδή ο αλγόριθμος Preflow Push[1] λόγω του μεγάλου προγραμματιστικού overhead που χρειάζεται ούτως ώστε να γίνονται σωστά οι απαραίτητοι έλεγχοι για να γίνει διοχέτευση ροής από την πηγή και να φτάσει στον προορισμό στο τελικά καταστρέφει τα πλεονεκτήματα που έχει. Έτσι θεωρώ ότι ο λόγος που ο χρόνος εκτινάσσεται από ένα σημείο και μετά είναι ο μεγάλος αριθμός ελέγχων που γίνονται οι οποίοι θεωρητικά είναι μηδαμινού χρόνου και αυτό ισχύει εν μέρη όταν ο γράφος είναι μικρός, όμως τελικά επηρεάζουν τον συνολικό χρόνο του αλγορίθμου.

Ακόμα παρατηρούμε ξεκάθαρα ότι πράγματι ο χρόνος του αλγορίθμου Edmonds Karp[3] ξεπερνά σε κάποιο σημείο τον χρόνο κάποιων από τους υπόλοιπους αλγορίθμους όμως αυτό συμβαίνει όταν έχουμε πολύ μεγάλους γράφους και κατά συνέπεια και "τεράστιο" αριθμό ακμών. Ενώ όταν ο γράφος είναι μικρός παρατηρούμε ότι ο χρόνος του αλγορίθμου Edmonds Karp[3] δεν είναι μεγαλύτερος αλλά είναι σχεδόν ίδιου χρόνου από τον χρόνο τους αλγόριθμους Scaling Max Flow[1][1] και Ford Fulkerson[1]. Εικάζω ότι αυτό προέκυψε λόγω του ότι μπορεί ο γράφος να είναι πυκνός και άρα να έχουμε μεγάλο αριθμό ακμών και αυτό προσδίδει χρόνο στον αλγόριθμο Edmonds Karp[3] όμως το ότι περιέχει πολλά μικρά μονοπάτια (δύο μόνο βήματα το καθένα), ευνοούν τον

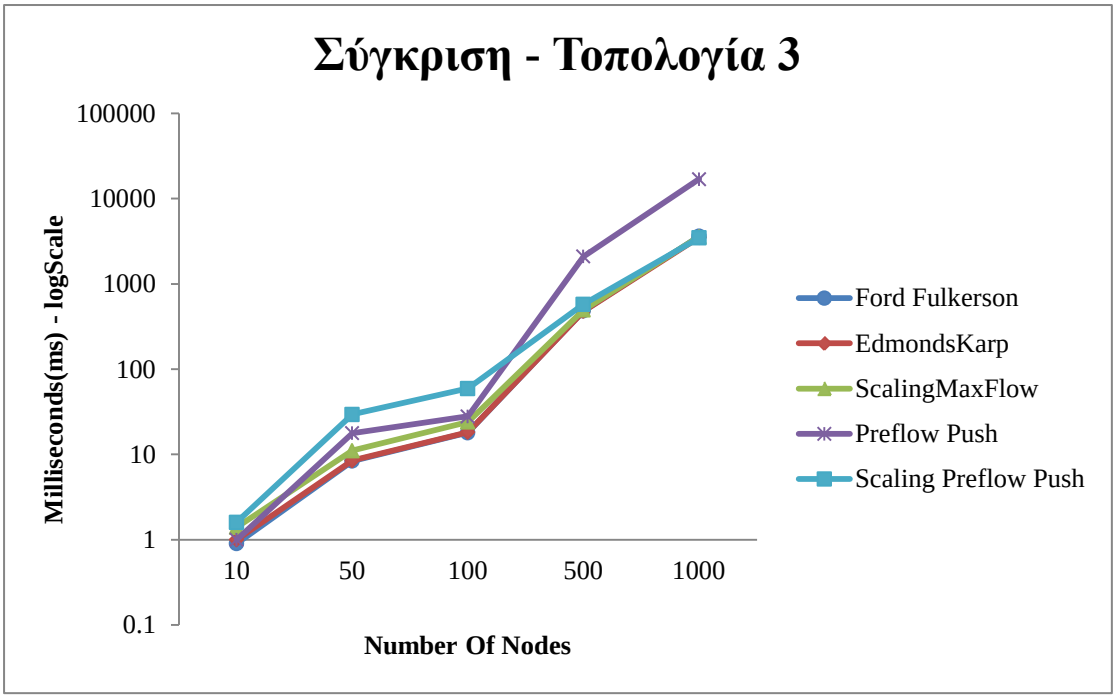
αλγόριθμο διότι επιλέγει πρώτα τα πιο σύντομα μονοπάτια για να στείλει ροή από την πηγή στον προορισμό κα έτσι εξοικονομεί χρόνο πλησιάζοντας έτσι τον χρόνο των άλλων αλγορίθμων.

Όσον αφορά τώρα την συμπεριφορά του αλγορίθμου Scaling Preflow Push[2] εικάζω ότι όσο μεγαλώνει ο γράφος που έχει στην διάθεση του να επεξεργαστεί ο αλγόριθμος ενώ οι άλλοι αλγόριθμοι λόγω της τεχνικής επιλογής των μονοπατιών, ή και λόγω του αριθμού των ακμών αυξάνουν με πιο γρήγορο ρυθμό τον χρόνο τους ο αλγόριθμος αυτός με την δική του τεχνική είναι καλύτερος από τους υπόλοιπους.

Τοπολογία 3

Στην Τοπολογία 3 οι γράφοι είναι τυχαίοι και όπως ξέρουμε έχουν ως στόχο να αποδυναμώσουν τους αλγορίθμους τους οποίους ο χρόνος εξαρτάται από τις χωρητικότητες των ακμών που προέρχονται από την πηγή και καταλήγουν σε ένα οποιοδήποτε άλλο κόμβο. Οι αλγόριθμοι που ο θεωρητικός τους χρόνος είναι άμεσα επηρεαζόμενος από την παράμετρο αυτή είναι οι αλγόριθμοι Scaling Max Flow[1][1], Ford Fulkerson[1]και Scaling Preflow Push[2]. Από τους τρείς προηγούμενους που αναφέρθηκαν πιο πάνω, πιο πολύ επηρεάζεται από την παράμετρο αυτή ο χρόνος του Ford Fulkerson[1]διότι ο χρόνος εξαρτάται άμεσα από τον άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή και στην συγκεκριμένη τοπολογία οι χωρητικότητες αυτών των ακμών είναι τεράστιες. Οπότε αυτό που αναμένουμε είναι να δούμε τον χρόνο του αλγορίθμου Ford Fulkerson, κυρίως για πυκνούς γράφους να είναι ο χειρότερος από όλους τους υπόλοιπους.

Γραφική Παράσταση:



Η γραφική παράσταση που προέκυψε έχει αρκετό ενδιαφέρον διότι καταλύει την υπόθεση που κάναμε πιο πάνω. Παρατηρούμε ότι τελικά ο αλγόριθμος Ford Fulkerson[1]είναι ο καλύτερος από όλους ενώ εμείς περιμέναμε να δούμε εντελώς το αντίθετο. Ο λόγος που συνέβηκε αυτό πιστεύω είναι επειδή οι αλγόριθμοι επηρεάστηκαν από τις μεγάλες χωρητικότητες των ακμών που ξεκινούν από την πηγή διαφορετικά από ότι υπολογίσαμε.

Ο αλγόριθμος Scaling Preflow Push[2] όπως ήδη γνωρίζουμε χρησιμοποιεί την τιμή της μεγαλύτερης δύναμης του δύο, που είναι μικρότερη όμως από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο για να κάνει επαναλήψεις στα πλαίσια της τεχνικής που χρησιμοποιεί. Οπότε επειδή στην τοπολογία αυτή μονό οι ακμές που ξεκινούν από την πηγή έχουν μεγάλη χωρητικότητα και οι υπόλοιπες ακμές είναι πολύ στενές οι επαναλήψεις αυτές γίνονται άδικα και προσδίδουν χρόνο και overhead στον συνολικό αποτέλεσμα του αλγορίθμου.

Ο αλγόριθμος Scaling Max Flow[1][1] κάνει επίσης κάποιες άχρηστες επαναλήψεις για αυτό και είναι σχεδόν συνέχεια πιο αργός από τους άλλους αλγορίθμους της ομάδας του.

Επίσης βλέπουμε ότι οι αλγόριθμοι Ford Fulkerson[1]και Edmonds Karp[3] είναι συνεχώς πολύ κοντά. Ο αλγόριθμος Edmonds Karp[3] εξαρτάται κυρίως από τον αριθμό των ακμών και εδώ ειδικά για μικρούς γράφους έχουμε σχεδόν ένα πλήρες συνδεδεμένο γράφο. Τώρα ο αλγόριθμος Ford Fulkerson[1]εξαρτάται από τον αριθμό των ακμών και από το άθροισμα των χωρητικοτήτων των ακμών που ξεκινούν από την πηγή. Ο λόγος που έχουν παρόμοιους χρόνους είναι ίσως επειδή οι γράφοι είναι πολύ πυκνοί και τελικά δουλεύουν με παρόμοιο τρόπο επειδή έχουμε μικρά μονοπάτια οπότε έχει να κάνει με την τεχνική που επιλέγουν τα μονοπάτια και όχι με τις παραμέτρους του θεωρητικού τους χρόνου οι οποίες δεν συσχετίζονται ξεκάθαρα.

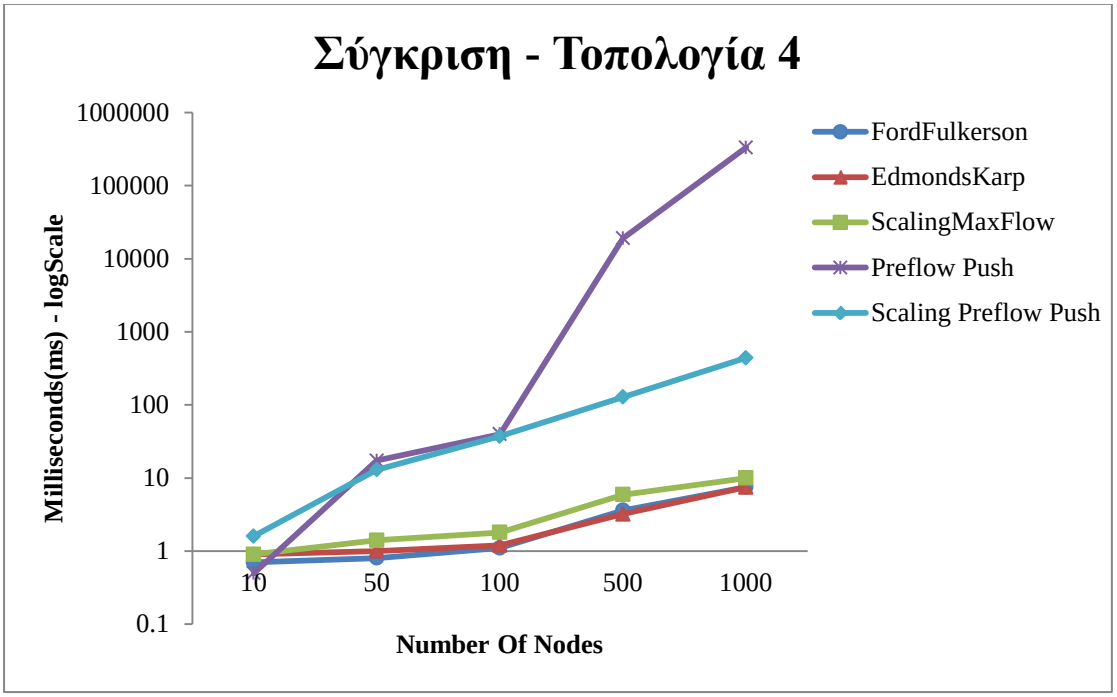
Ο αλγόριθμος Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για μεγάλο διάστημα ήταν καλύτερος από τον αλγόριθμο Scaling Preflow Push[2]. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] υπερνικά ακόμα και τον άσχημο χρόνο που χρειάζεται ο Scaling Preflow Push[2] λόγω των άχρηστων επαναλήψεων.

Τοπολογία 4

Με βάση την Τοπολογία 4 γνωρίζουμε ότι οι γράφοι που δημιουργούνται είναι αραιοί και επίσης ότι οι χωρητικότητες των ακμών που προέρχονται από την πηγή είναι πολύ μεγάλες. Οπότε λαμβάνοντας υπόψη τα χαρακτηριστικά της τοπολογίας αυτής ξέρουμε ότι δεν θα υπάρξουν περισσότερα από δύο μονοπάτια στον γράφο, από την πηγή στον προορισμό. Άρα οι αλγόριθμοι Ford Fulkerson[1]και Edmonds Karp[3] δεν αναμένουμε ότι θα χρειαστούν πολύ χρόνο για να λύσουν το πρόβλημα Εύρεσης Μέγιστης Ροής και επίσης περιμένουμε ότι θα είναι κοντά σχετικά οι χρόνο τους

διότι το γεγονός ότι ο γράφος είναι πολύ αραιός θα του κάνει να λειτουργούν σχεδόν το ίδιο διότι η διαφορετικότητα στη τεχνική επιλογής μονοπατιού δεν θα έχει καμιά σημασία κυρίως για μικρό αριθμό κόμβων. Ακόμα πιστεύω ότι οι αλγόριθμοι Scaling Max Flow[1] και Scaling Preflow Push[2] θα κάνουν σχετικά περισσότερο χρόνο από τους υπόλοιπους αλγόριθμους για το λόγο ότι θα κάνουν άχρηστες επαναλήψεις που θα τους καθιστούν αργούς. Ο αλγόριθμος Preflow Push[1] επειδή εξαρτάται από τον αριθμό των κόμβων κυρίως ο χρόνος του αναμένω ότι θα είναι αρκετά αργός σε σχέση με τους υπόλοιπους διότι οι παράμετροι που τον επηρεάζουν θα είναι πιο αισθητοί στον χρόνο από ότι τις παραμέτρους που επηρεάζουν τους άλλους αλγορίθμους που μελετούμε.

Γραφική Παράσταση:



Όπως ήταν αναμενόμενο βλέπουμε ότι οι αλγόριθμοι Edmonds Karp[3] και Ford Fulkerson[1]παρουσιάζουν αρκετά καλό χρόνο και ειδικά για μεγάλο αριθμό κόμβων έχουν πολύ παρόμοια συμπεριφορά. Αρχικά βέβαια ενώ θα αναμέναμε ο Ford Fulkerson[1]να είναι αυτός θα είναι ο πιο αργός μεταξύ των δύο βλέπουμε ότι συμβαίνει ακριβώς το αντίθετο και πιστεύω ότι αυτό συμβαίνει επειδή η παράμετρος της μεγάλης χωρητικότητας των ακμών δεν παίζει τόσο μεγάλο ρόλο όταν ο γράφος είναι αραιός και έχει μικρό αριθμό κόμβων.

Επίσης παρατηρούμε ότι όντως ο αλγόριθμος Scaling Max Flow[1] είναι χειρότερος από τους αλγορίθμους της ομάδας του, οπότε συμπεραίνουμε ότι αυτό προκύπτει διότι κάνει όπως υπολογίσαμε άχρηστες επαναλήψεις οι οποίες προσδίδουν χρόνο στο συνολικό αποτέλεσμα του αλγορίθμου.

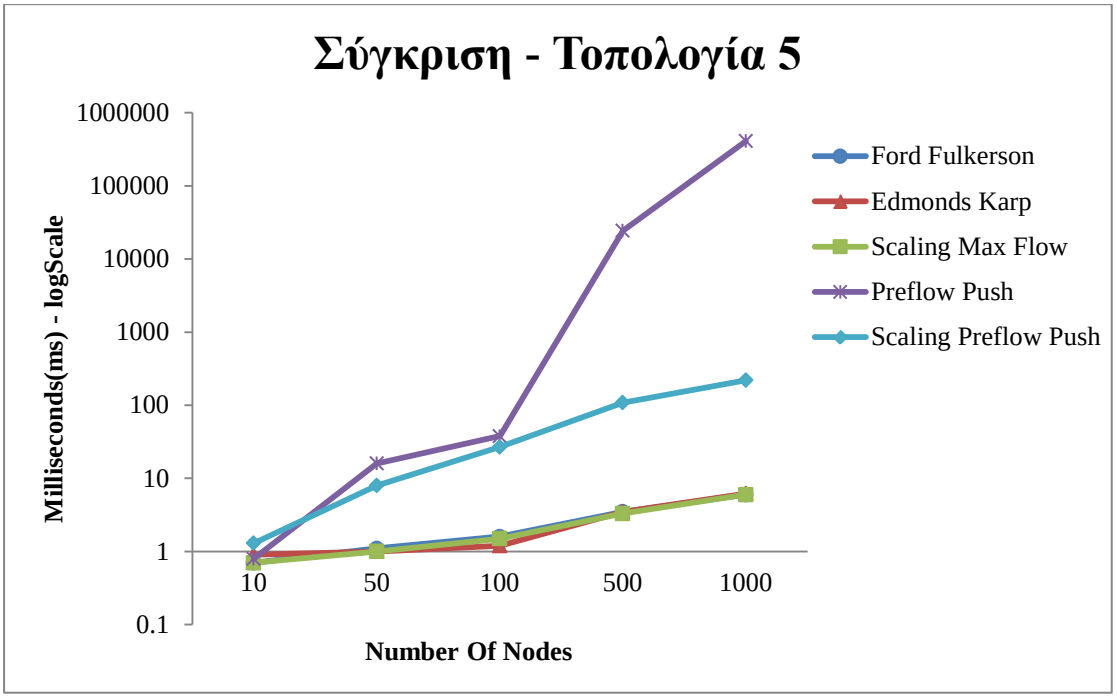
Όπως προβλέψαμε συμπεριφέρθηκε και ο αλγόριθμος Scaling Preflow Push[2] ο οποίος όχι μόνο έκανε άχρηστες επαναλήψεις αλλά με αυτό τον τρόπο εκτόξευσε τον χρόνο του λόγω του overhead που προσδίδει η κάθε μια από τις επαναλήψεις ξεχωριστά.

Ο αλγόριθμος Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για κάποιο διάστημα ήταν καλύτερος από τον αλγόριθμο Scaling Preflow Push[2]. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] υπερνικά ακόμα και τον άσχημο χρόνο που χρειάζεται ο Scaling Preflow Push[2] λόγω των άχρηστων επαναλήψεων που κάνει.

Τοπολογία 5

Με βάση τον σχεδιασμό της τοπολογίας αυτής επειδή δημιουργούνται πολύ αραιοί γράφοι με μικρές χωρητικότητες αναμένουμε ότι τον χειρότερο χρόνο θα τον έχει ο αλγόριθμος Preflow Push[1] διότι είναι ο χρόνος του εξαρτάται κυρίως από τον αριθμό των κόμβων. Επίσης άσχημο χρόνο αναμένουμε να δούμε και για τον αλγόριθμο Scaling Preflow Push[2] διότι και αυτού ο χρόνος του εξαρτάται από τον αριθμό των κόμβων σε πιο μικρό βέβαια βαθμό από ότι εξαρτάται ο χρόνος του αλγορίθμου Preflow Push[1]. Οπότε αναμένουμε να δούμε παρόμοιους χρόνους για τους αλγορίθμους της ομάδας Ford Fulkerson[1]και μια διαφορετική και παράλληλα χειρότερη συμπεριφορά για τους αλγορίθμους της ομάδας Preflow Push[1].

Γραφική Παράσταση:



Η γραφική παράσταση που προέκυψε μέσα από τα πειράματα μου για την Τοπολογία 5 είναι ακριβώς αυτό που ανέμενα. Οι αλγόριθμοι που ανήκουν στην ομάδα Ford Fulkerson[1]παρουσιάζουν παρόμοιο χρόνο ο οποίος είναι πολύ καλύτερος από τον χρόνο που παρουσιάζουν οι αλγόριθμοι της ομάδας Preflow Push[1]. Φαίνεται να είναι λίγο καλύτερος ο αλγόριθμος Edmonds Karp[3] σε

αρκετά σημεία. Εικάζω ότι αυτό συμβαίνει λόγω της τεχνικής που χρησιμοποιεί, βάση της οποίας επειδή ο γράφος είναι αραιός εξοικονομεί κάποιο χρόνο.

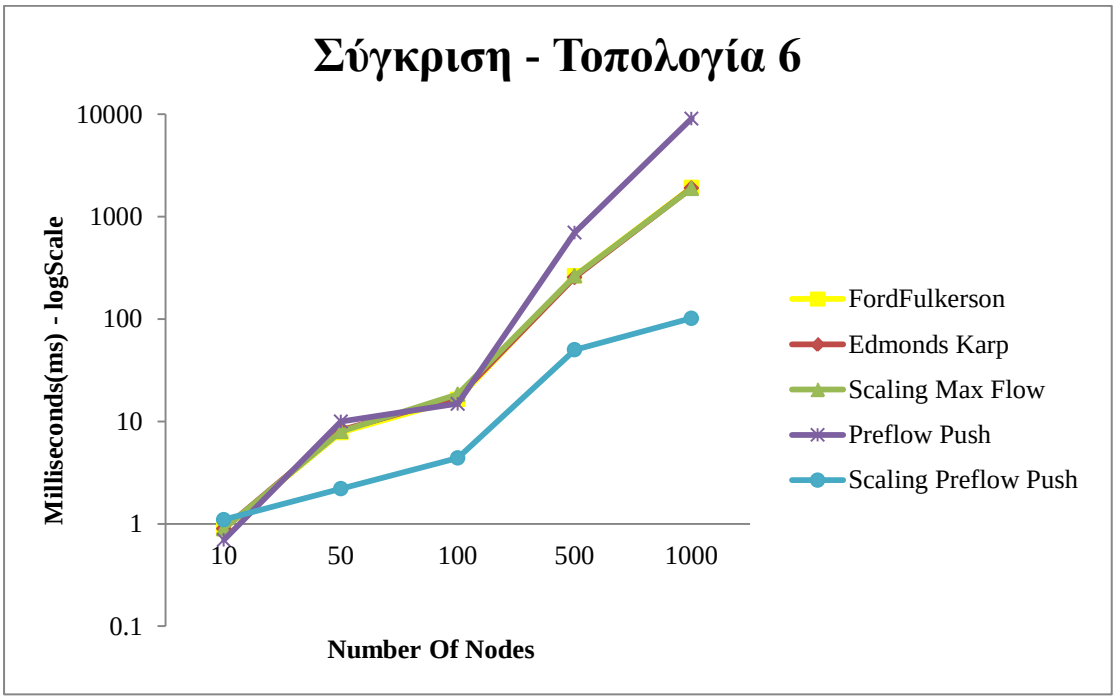
Επίσης παρατηρούμε ότι ο αλγόριθμος Scaling Preflow Push[2] χρειάζεται αρκετό χρόνο για να υπολογίσει τη Μέγιστη Ροή σε ένα αραιό γράφο με μικρές τιμές στις χωρητικότητες των ακμών και αυτό εικάζουμε ότι συμβαίνει γιατί ο χρόνος τους εξαρτάται από τον αριθμό των κόμβων ενώ ο χρόνος των αλγόριθμων που ανήκουν στην άλλη ομάδα δεν εξαρτάται από αυτή την παράμετρο. Ακόμα μπορεί να παρουσιάζει άσχημο χρόνο επειδή λόγω του ότι ο γράφος είναι αραιός ίσως κάνει κάποιες άχρηστες επαναλήψεις που του προσδίδουν χρόνο και overhead.

Ο αλγόριθμος Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για κάποιο διάστημα ο χρόνος του είναι αρκετά κοντά στον χρόνο του αλγόριθμου Scaling Preflow Push[2]. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] κάνει τον χρόνο του να εκτοξεύεται.

Τοπολογία 6

Η τοπολογία αυτή λόγω του ότι δημιουργεί απλούς και με μικρές χωρητικότητες γράφους αναμένουμε ότι οι αλγόριθμοι που ανήκουν στην ομάδα Ford Fulkerson[1]θα έχουν και παρόμοιους χρόνους. Ο λόγος που αναμένουμε να δούμε κάτι τέτοιο είναι επειδή οι τεχνικές επιλογής μονοπατιών που ακολουθούν οι αλγόριθμοι που ανήκουν στη ομάδα Ford Fulkerson[1]δεν παίζουν ρόλο σε αυτού του είδους τους γράφους που δημιουργούνται με βάση την Τοπολογία 6. Επίσης οι αλγόριθμοι της ομάδας Preflow Push[1] πιστεύουμε βάση και την θεωρητική τους ανάλυση, ότι θα διαφέρουν οι χρόνοι τους αισθητά, με τον αλγόριθμο Scaling Preflow Push[2] να είναι πολύ καλύτερος.

Γραφική Παράσταση:



Η γραφική παράσταση που προέκυψε έρχεται και επιβεβαιώνει τις υποθέσεις που κάναμε πιο πάνω. Αυτό που παρατηρούμε λοιπόν είναι ότι οι αλγόριθμοι που ανήκουν στην ομάδα Ford Fulkerson[1]έχουν παρόμοιο χρόνο διότι όπως είπαμε και πιο πάνω λόγω του ότι ο γράφος είναι πολύ συγκεκριμένος και με ίδιες, μικρές χωρητικότητες για όλες τις ακμές οι τεχνικές που χρησιμοποιούν δεν παίζουν και τόσο μεγάλο ρόλο.

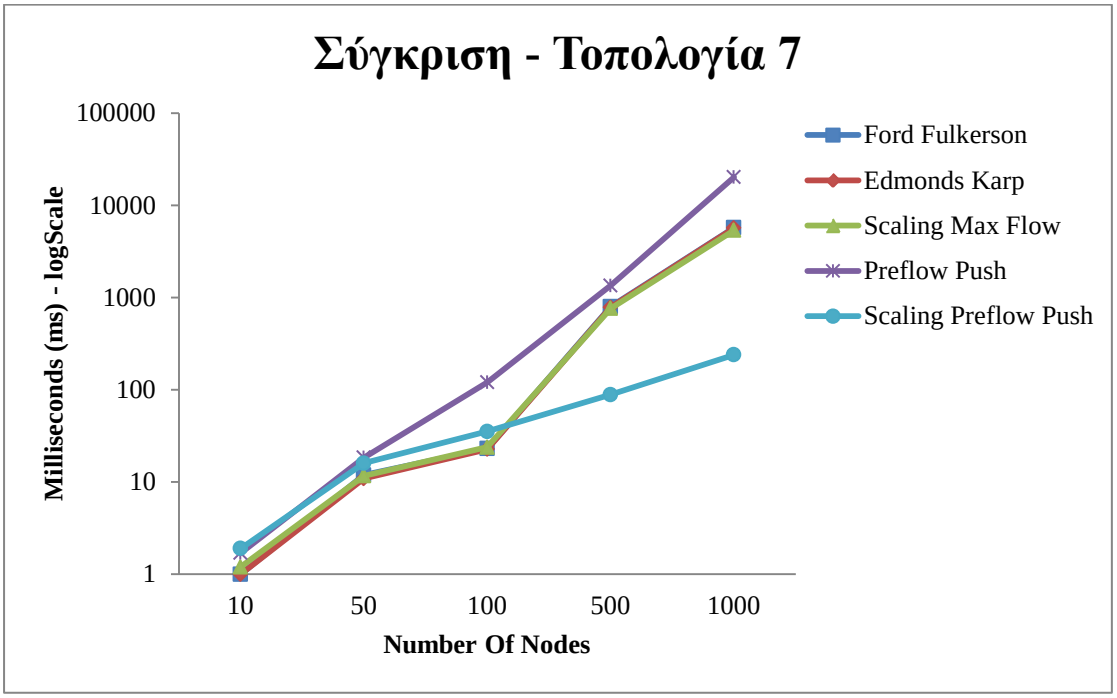
Όσο για τον αλγόριθμο Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για κάποιο διάστημα ο χρόνος του είναι αρκετά κοντά στον χρόνο των αλγορίθμων που ανήκουν στην ομάδα Ford Fulkerson. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] κάνει τον χρόνο του να μεγαλώνει ενώ ήταν αρκετά καλός για μικρότερους γράφους.

Παρατηρούμε επίσης ότι ο χρόνος του αλγορίθμου Scaling Preflow Push[2] είναι μακράν καλύτερος από τον χρόνο των υπόλοιπων αλγορίθμων και εικάζω ότι αυτό συμβαίνει επειδή οι παράμετροι που επηρεάζουν τον θεωρητικό του χρόνο διατηρούνται πολύ μικρές. Συγκεκριμένα ο θεωρητικός χρόνος του αλγορίθμου Scaling Preflow Push[2] είναι ο εξής: $|V||E|+|V|^2 \log_2 U$ (όπου U= μεγαλύτερη δύναμη του δύο, που είναι μικρότερη από την μεγαλύτερη χωρητικότητα που μπορεί να έχει μια ακμή που ξεκινά από την πηγή προς οποιοδήποτε άλλο κόμβο, V = αριθμός των κόμβων που υπάρχουν στον γράφο , E =αριθμός ακμών που υπάρχουν στον γράφο). Όπως είναι λοιπόν φανερό το δεύτερο σκέλος της πιο πάνω πράξης ($|V|^2 \log_2 U$) μηδενίζεται σε αυτή την τοπολογία διότι $\log_2(1)=0$. Οπότε μένει μόνο το πρώτο σκέλος. Επίσης ο αλγόριθμος κάνει μόνο μια επανάληψη (βλ. ψευδοκώδικα) για γράφους που δημιουργούνται με βάση την Τοπολογία 6.

Τοπολογία 7

Στην τοπολογία αυτή επειδή οι γράφοι που δημιουργούνται είναι αρκετά πυκνοί με μεγάλο εύρος τιμών στις χωρητικότητες αναμένουμε ότι οι αλγόριθμοι που θα επηρεαστούν είναι αυτοί, των οποίων ο χρόνος εξαρτάται από τον αριθμό των ακμών και από τις χωρητικότητες των ακμών. Άρα υποθέτουμε ότι οι αλγόριθμοι της ομάδας Ford Fulkerson[1]θα παρουσιάζουν παρόμοιο χρόνο διότι εξαρτούνται από αυτές τις παραμέτρους και λόγω του ότι οι γράφοι είναι πολύ πυκνοί θα κάνουν σχεδόν ίδιο χρόνο. Επίσης είναι πιθανό να δούμε τον αλγόριθμο Scaling Max Flow[1] να είναι λίγο καλύτερος από τους άλλους δύο, διότι θα αξιοποιεί τις πιθανώς μεγάλες χωρητικότητες που θα δοθούν στις ακμές που ξεκινούν από την πηγή, για να εξαλείψει κάποιους ελέγχους. Επίσης αναμένουμε ότι οι αλγόριθμοι της ομάδας Preflow Push[1] θα έχουν παρόμοιο χρόνο μεταξύ τους λαμβάνοντας υπόψη την θεωρητική τους ανάλυση και την τοπολογία που μελετούμε. Όταν λέμε παρόμοιους χρόνους εννοούμε παρόμοια συμπεριφορά στην γραφική παράσταση.

Γραφική Παράσταση:



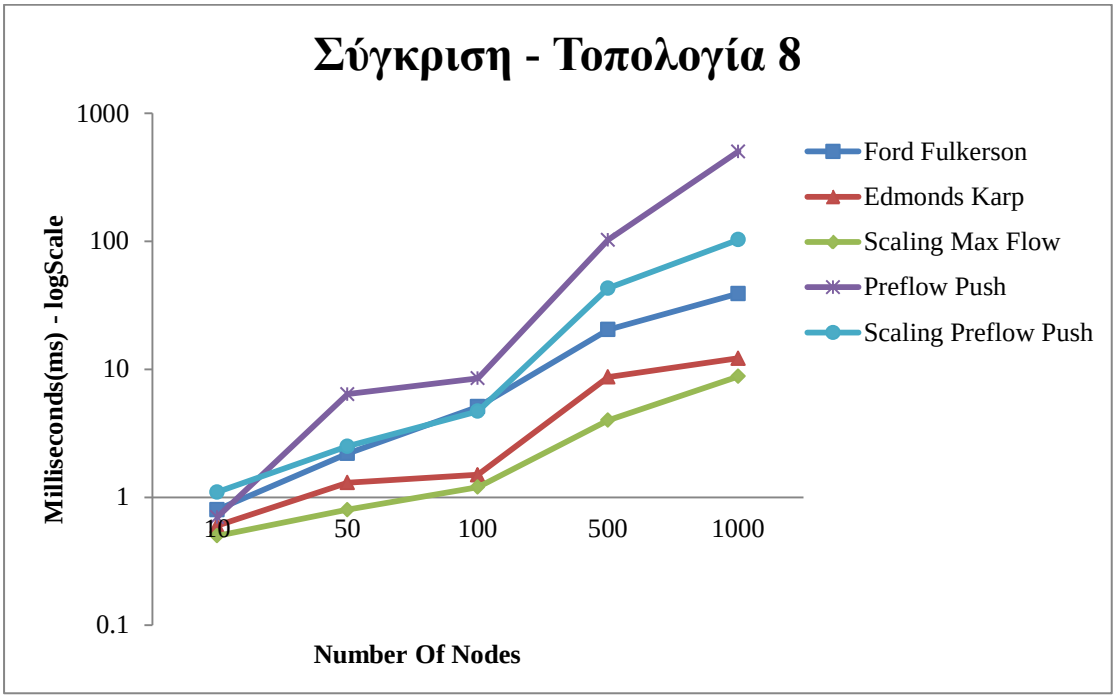
Όπως μπορούμε να παρατηρήσουμε στην γραφική παράσταση αυτά που αναμέναμε συνέβησαν σε κάποιο βαθμό. Οι αλγόριθμοι που ανήκουν στην ομάδα Ford Fulkerson[1]παρουσιάζουν όντως παρόμοια συμπεριφορά, με τον Edmonds Karp[3] να είναι αρχικά λίγο καλύτερος από τους άλλους δύο αντί όπως προβλέψαμε να είναι καλύτερος ο Scaling Max Flow[1]. Αυτό ίσως συνέβηκε επειδή για μικρό κυρίως αριθμό κόμβων οι γράφοι που δημιουργούνται με βάση την Τοπολογία 7 είναι σχεδόν όλοι προς όλους και αυτό τελικά ευνοεί τον Edmonds Karp[3], διότι υπάρχουν πολλά μονοπάτια το πολύ δύο βήματα διαμέσου των οποίων μπορεί να στείλει ροή, τα οποία είναι αυτά που θα επιλέξει πρώτα. Ακόμα αυτό που παρατηρούμε είναι ότι ο χρόνος των αλγορίθμων της ομάδας Ford Fulkerson[1]αυξάνεται πιο απότομα από τον χρόνο του αλγορίθμου Scaling Preflow Push[2].

Ο αλγόριθμος Scaling Preflow Push[2] ενώ αρχικά έχει παρόμοιο χρόνο με τον Preflow Push[1] στην συνέχεια καταλήγει να είναι πολύ καλύτερος του. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] κάνει τον χρόνο του να μεγαλώνει. Επιπλέον βλέπουμε ότι ενώ αρχικά οι αλγόριθμοι της ομάδας Ford Fulkerson[1]είναι καλύτεροι από τον Scaling Preflow Push[2] στην συνέχεια γίνεται πολύ καλύτερος τους όσο μεγαλώνει ο αριθμός των κόμβων. Υπολογίζω ότι αυτό συμβαίνει διότι αρχικά οι γράφοι που δημιουργούνταν ήταν μικροί και ο Scaling Preflow Push[2] έκανε κάποιες επιπλέον άχρηστες επαναλήψεις λόγω των χωρητικοτήτων και αυτό πρόσδιδε κάποιο επιπρόσθετο χρόνο στο συνολικό αποτέλεσμα του. Στην συνέχεια όμως αυτές οι επαναλήψεις απέκτησαν νόημα στους πιο μεγάλους γράφους και βελτίωσαν την συμπεριφορά του σε σχέση με τους υπόλοιπους.

Τοπολογία 8

Στην Τοπολογία 8 στόχος είναι να αποδυναμωθεί ο αλγόριθμος Ford Fulkerson[1]όσον αφορά την τεχνική επιλογής μονοπατιών που ακολουθεί. Παράμετροι όπως χωρητικότητες και πυκνότητα γράφου έχουν κρατηθεί χαμηλά διότι δεν θέλουμε ο αλγόριθμος να επηρεαστεί από κάτι άλλο εκτός από την τεχνική του και επίσης λόγω του ότι οι παράμετροι αυτοί επηρεάζουν τον χρόνο και άλλων αλγορίθμων. Επομένως αυτό που αναμένουμε να δούμε είναι τον αλγόριθμο Ford Fulkerson[1]να παρουσιάζει τον χειρότερο χρόνο από τους αλγορίθμους που είναι στην ομάδα του λόγω της τεχνικής του. Ακόμα αναμένουμε ότι ο αλγόριθμος Scaling Preflow Push[2] θα έχει καλύτερο χρόνο από τον Preflow Push[1] διότι δεν θα κάνει άχρηστες επαναλήψεις και θα αξιοποιήσει αποτελεσματικά την δική του τακτική.

Γραφική Παράσταση:



Όπως αναμέναμε ο αλγόριθμος Ford Fulkerson[1]είναι πολύ χειρότερος από τους αλγορίθμους της ομάδας του οπότε έχει εν μέρη επιτευχθεί ο στόχος ου θέσαμε για αυτήν την τοπολογία. Παρατηρούμε όμως ότι ο χρόνος του δεν είναι χειρότερος από τους αλγορίθμους της ομάδας Preflow Push[1]. Εικάζω ότι αυτό συμβαίνει επειδή το overhead που έχουν οι αλγόριθμοι της ομάδας Preflow Push[1] δεν αξίζει τον κόπο όταν οι γράφοι δημιουργούνται με βάση αυτήν την τοπολογία.

Ακόμα βλέπουμε ότι ο καλύτερος χρόνος ανήκει στον αλγόριθμο Scaling Max Flow[1] και αυτό είναι αναμενόμενο διότι με βάση την τεχνική του αλλά και τον θεωρητικό του χρόνο αυτή ακριβώς θα έπρεπε να είναι η συμπεριφορά του. Να είναι καλύτερος από τον Ford Fulkerson[1]διότι

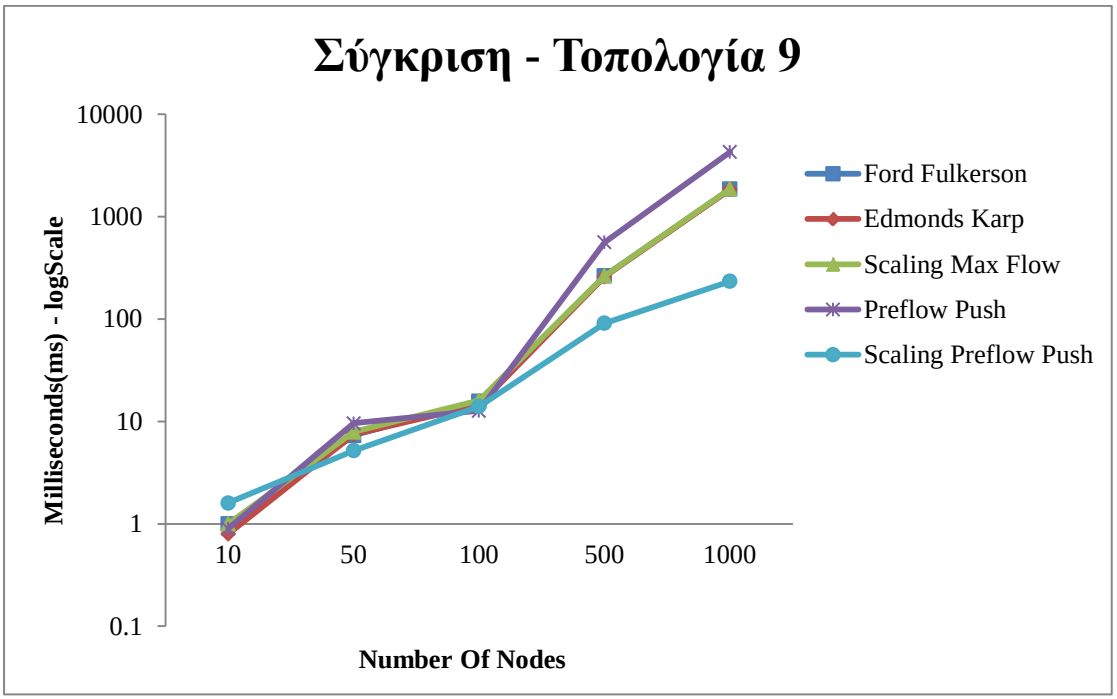
δεν υπάρχει παράμετρος που να τον κάνει θεωρητικά χειρότερο. Όπως επίσης καλύτερος και από τον Edmonds Karp[3] διότι με βάση την θεωρητική τους ανάλυση και τους γράφους που δημιουργούνται είναι αναμενόμενο.

Ο αλγόριθμος Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για κάποιο διάστημα ο χρόνος του είναι αρκετά κοντά στον χρόνο του αλγόριθμου Scaling Preflow Push[2]. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] κάνει τον χρόνο του να μεγαλώνει χωρίς όμως να αλλάζει συμπεριφορά σε σχέση με τον Scaling Preflow Push[2]

Τοπολογία 9

Στην Τοπολογία 9 τα αποτελέσματα που αναμένουμε δεν είναι εύκολο να προσδιοριστούν. Ο λόγος είναι ότι οι γράφοι που δημιουργούνται με βάση την τοπολογία αυτή είναι απλοί, όχι τόσο πυκνοί, έχουν όμως μεγάλες χωρητικότητες στις ακμές που είναι όμως και ίδιες. Επίσης ξέρουμε ότι τα μονοπάτια που δημιουργούνται είναι το πολύ δύο βήματα και αυτό περιπλέκει ακόμα περισσότερο τα πράγματα.

Γραφική Παράσταση:



Βλέποντας την γραφική παράσταση που προέκυψε παρατηρούμε ότι οι χρόνοι που παρουσιάζουν οι αλγόριθμοι είναι αρκετά κοντινοί. Ο αλγόριθμος Scaling Preflow Push[2] είναι λίγο

καλύτερος όλους τους άλλους σε κάποια σημεία το οποίο δεν μπορούμε να εξηγήσουμε όμως είναι κάτι που προέκυψε από τα πειράματα μου. Το ίδιο συμβαίνει και για τον αλγόριθμο Preflow Push[1] του οποίου ο χρόνος μετά γίνεται χειρότερος από όλους.

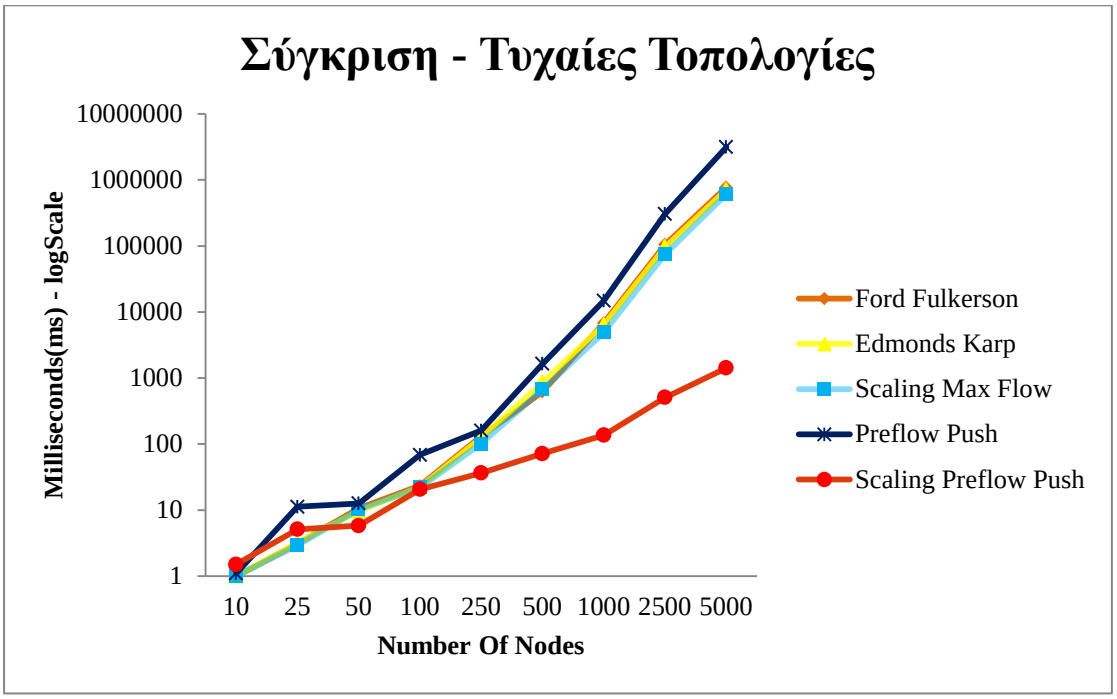
Συμπερασματικά καταλήγουμε στο ότι στην Τοπολογία 9 οι χρόνοι που παίρνουμε που παίρνουμε από τους αλγορίθμους είναι ανάλογοι του αριθμού των κόμβων και επίσης δεν μπορούμε να προσδιορίσουμε τον καλύτερο αλγόριθμο με βάση αυτήν την τοπολογία

Τυχαίες Τοπολογίες

Στα πειράματα που πραγματοποιήθηκαν για τις τυχαίες τοπολογίες, όπως ήδη γνωρίζουμε οι τιμές των χωρητικοτήτων ήταν μικρές, διότι δεν θέλαμε κάποιοι αλγόριθμοι οι οποίοι επηρεάζονται από την παράμετρο να παρουσιάσουν ένα μη αντιπροσωπευτικό χρόνο. Επίσης γνωρίζουμε ότι με βάση το προγραμματιστικό κομμάτι για την παραγωγή αυτών των τυχαίων τοπολογιών, οι γράφοι που δημιουργούνται για μικρό κυρίως αριθμό κόμβων είναι πολύ πυκνοί. Λόγω του ότι λοιπόν οι τοπολογίες που δημιουργούνται είναι εντελώς τυχαίες δεν μπορούμε να αναμένουμε συγκεκριμένα αποτελέσματα. Όμως υπάρχουν κάποια συμπεράσματα τα οποία μπορούν να προκύψουν με βάση τις λεπτομέρειες που αναφέραμε πιο πάνω. Αναμένουμε λοιπόν ότι λόγω του ότι οι γράφοι είναι πολύ πυκνοί και άρα έχουμε μεγάλο αριθμό ακμών, ο αλγόριθμος Edmonds Karp[3] του οποίου ως γνωστόν ο χρόνος εξαρτάται άμεσα από την παράμετρο αυτή, θα είναι μεγαλύτερος από τον χρόνο των αλγορίθμων της ομάδας του. Ακόμα ξέροντας ότι οι τιμές των χωρητικοτήτων είναι πολύ μικρές, αναμένουμε ότι θεωρητικά ο χρόνος του αλγορίθμου Scaling Preflow Push[2] θα είναι καλύτερος από τους χρόνους των άλλων αλγορίθμων.

Γραφική Παράσταση:

Η γραφική παράσταση που θα δούμε παρακάτω μας δίνει κατά μέσο όρο την συμπεριφορά του κάθε αλγορίθμου σε σχέση με τους υπόλοιπους.



Η γραφική παράσταση που προέκυψε επιβεβαιώνει κάποιες από τις υποθέσεις μου. Παρατηρούμε ότι ο αλγόριθμος Edmonds Karp[3] παρουσιάζει πράγματι χειρότερο χρόνο από τους

αλγορίθμους της ομάδας του για μεγάλο διάστημα. Η συμπεριφορά αυτή αλλάζει μόνο για πολύ μεγάλο αριθμό κόμβων όπου ο αλγόριθμος Ford Fulkerson[1]γίνεται ο χειρότερος από τους αλγορίθμους της ομάδας του ξεπερνώντας ακόμα και τον Edmonds Karp[3]. Μία άλλη παρατήρηση είναι ότι ο χρόνος του αλγόριθμου Scaling Max Flow[1] είναι λίγο καλύτερος από τους υπόλοιπους αλγορίθμους της ομάδας του. Ίσως αυτό συμβαίνει επειδή αξιοποιεί τις τιμές των χωρητικοτήτων των ακμών που προέρχονται από την πηγή.

Η διαφορά όμως μεταξύ των χρόνων των αλγορίθμων που ανήκουν στην ομάδα Ford Fulkerson[1]δεν είναι πολύ μεγάλη. Εικάζω ότι αυτό συμβαίνει επειδή οι παράμετροι που τους επηρεάζουν είναι ελεγχόμενοι εκτός από τον αριθμό των ακμών, ο οποίος μπορεί να μην είναι ελεγχόμενος αλλά είναι περίπου στα ίδια επίπεδα για κάθε τοπολογία που δημιουργείται. Οπότε υποθέτω ότι παρουσιάζουν παρόμοιος χρόνους διότι η τεχνική που χρησιμοποιεί ο καθένας δεν παίζει τόσο μεγάλο ρόλο λόγω της τυχαιότητας που υπάρχει στις τοπολογίες και επίσης λόγω του ότι υπολογίζεται ένας μέσος όρος ο οποίος εξομαλύνει τα τελικά αποτελέσματα.

Παρατηρούμε επίσης ότι ο αλγόριθμος Scaling Preflow Push[2] παρουσιάζει τον καλύτερο χρόνο κυρίως για μεγάλο αριθμό κόμβων. Αναμέναμε ότι θα συνέβαινε κάτι τέτοιο διότι έχοντας υπόψη τον θεωρητικό χρόνο του αλγορίθμου και τις παραμέτρους που επηρεάζονταν ήταν εμφανές ότι θα ήταν ο καλύτερος από όλους όσο μεγάλωνε ο αριθμός των κόμβων.

Ακόμα βλέπουμε ότι ο αλγόριθμος Preflow Push[1] από ένα σημείο και μετά είναι χειρότερος από όλους ενώ για κάποιο διάστημα ο χρόνος του είναι αρκετά κοντά στους χρόνους των υπόλοιπων αλγορίθμων με κάποια σκαμπανεβάσματα τα οποία τα αποδίδουμε την τυχαιότητα των τοπολογιών. Εικάζω ότι αυτό συμβαίνει επειδή από ένα σημείο και μετά για μεγάλο αριθμό κόμβων το overhead του αλγορίθμου Preflow Push[1] κάνει τον χρόνο του να μεγαλώνει κατά πολύ σε σχέση με τους υπόλοιπους χωρίς όμως να αλλάζει συμπεριφορά σε σύγκριση με την συμπεριφορά των αλγορίθμων της ομάδας Ford Fulkerson

Κεφάλαιο 6

Συμπεράσματα

6.1 Συμπεράσματα

Η τριβή που είχα με το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής με βοήθησε να καταλάβω σε μεγαλύτερο βαθμό την σημαντικότητα αυτού του προβλήματος. Τα Δίκτυα Ροής βρίσκονται σε πολλά σημεία στην καθημερινότητα μας και απεικονίζουν πολλά διαφορετικά προβλήματα στα οποία είναι σημαντικό να δώσουμε μια γρήγορη και σωστή λύση. Για αυτό εξάλλου το πρόβλημα αυτό μελετάτε συνέχεια και κατασκευάζονται αλγόριθμοι οι οποίοι στοχεύουν στην σωστή και γρήγορη επίλυση του. Το ιδανικό θα ήταν μέσα από αυτήν την μελέτη να μπορούσα να διακρίνω ξεκάθαρα ποιος από τους πέντε αλγορίθμους που μελέτησα υπερτερεί πάντοτε των υπολοίπων.

Συμπερασματικά όμως κατέληξα στο ότι κανένας από τους αλγορίθμους δεν υπερτερεί πάντοτε των υπολοίπων. Αυτό σημαίνει πως κανένας δεν ήταν ο καλύτερος από όλους σε όσες τοπολογίες μελετήσαμε. Για τον κάθε αλγόριθμο υπήρξε μια τοπολογία που τον αποδυνάμωσε. Επίσης για τον κάθε αλγόριθμο υπήρξε μια τοπολογία στην οποία ήταν ο καλύτερος από όλους τους άλλους. Καταλήγουμε λοιπόν στο συμπέρασμα ότι ο κάθε αλγόριθμος από τους πέντε που μελετήσαμε έχει ένα τρωτό σημείο. Αν θέλαμε να ορίσουμε εντελώς χαλαρά ποιος από τους πέντε αλγορίθμους είναι ο καλύτερος θα επιλέγαμε αυτόν που ήταν ο καλύτερος στις περισσότερες τοπολογίες που μελετήσαμε. Στις περισσότερες λοιπόν τοπολογίες που μελετήθηκαν συμπεριλαμβανομένης και της τυχαίας τοπολογίας, ο καλύτερος αλγόριθμος ήταν ο Scaling Preflow Push[2]. Δεν ήταν βέβαια πάντοτε καλύτερος από όλους διότι σε άλλες τοπολογίες παρουσίασε πολύ μεγάλο χρόνο σε σχέση με τους υπόλοιπους αλγορίθμους αλλά κρίνοντας με βάση τις τοπολογίες που κατασκεύασα ήταν καλύτερος από τους υπόλοιπους σχεδόν σε αρκετές από αυτές.

6.2 Δυσκολίες που αντιμετώπισα

Μια από τις πιο βασικές δυσκολίες που αντιμετώπισα ήταν το γεγονός ότι έπρεπε να γράψω τον κώδικα για τον κάθε αλγόριθμο, χωρίς να τον αδικώ σε σχέση με τους υπόλοιπους όσο αφορά το προγραμματιστικό κομμάτι. Επίσης το γεγονός ότι ο αλγόριθμος Preflow Push[1] είχε πολύ προγραμματιστικό overhead έκανε ακόμα πιο δύσκολή την υλοποίηση του. Επιπρόσθετα ένα άλλο πρόβλημα που προέκυψε όταν προσπαθούσα να κάνω τις αξιολογήσεις των αλγορίθμων, ήταν το ότι η μηχανή μου δεν διέθετε αρκετό heap space ώστε να μπορώ να τρέξω τους αλγορίθμους μου και με μεγάλους γράφους οπότε αυτό με εμπόδιζε να κάνω την μελέτη μου. Ευτυχώς μετά από παρέμβαση του επιβλέποντα καθηγητή μου κατάφερα να εξασφαλίσω ένα εικονικό χώρο στο Nephelae Cloud Computing και έτσι μπόρεσα να ολοκληρώσω την μελέτη μου.

6.3 Μελλοντικά Σχέδια

Αν είχα στην διάθεση μου περισσότερο χρόνο ώστε να εργαστώ περισσότερο πάνω σε αυτή την μελέτη θα προσπαθούσα να υλοποιήσω όσους πιο πολλούς αλγόριθμους είναι δυνατόν που να λύνουν αυτό το πρόβλημα ώστε να μπορώ αξιολογώντας τους με τον ίδιο τρόπο, να τους συγκρίνω μεταξύ τους και να έχω μια πιο εμπεριστατωμένη εικόνα όσον αφορά το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής.

Μεγάλο επίτευγμα βέβαια για εμένα θα ήταν να χρησιμεύσει η μελέτη μου σε μια πραγματική εφαρμογή όπως είναι για παράδειγμα η αποστολή νερού από την πηγή(εκεί όπου φυλάγεται το νερό) σε ολόκληρη την χώρα. Αν δηλαδή χρησιμοποιώντας την μελέτη μου να, με βάση τα δεδομένα και την τοπολογία που θα έχουμε στην διάθεση μας να γίνεται αποστολή όσο το δυνατό περισσότερου νερού με τον πιο εύκολο και σύντομο τρόπο, επιλέγοντας ένα από τους αλγορίθμους που μελέτησα.

Θα ήταν βέβαια το ιδανικό αν μέσα από μια πιο μακροχρόνια μελέτη μπορούσα να προσδιορίσω με σιγουριά ποιος από τους αλγορίθμους που γνωρίζουμε οι οποίοι λύνουν το πρόβλημα Εύρεσης Μέγιστης Ροής σε ένα Δίκτυο Ροής είναι ο πιο αποδοτικός χωρίς να μου δίνονται στοιχειά ούτε ως προς τις χωρητικότητες και κυρίως ούτε ως προς την τοπολογία.

Βιβλιογραφία

[1] KLEINBRG, JON and TARDOS, EVA,Algorithm Design, 1st Edition, ISBN 0321295358,

[2] Excess Scaling ή Scaling Preflow Push, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.4191&rep=rep1&type=pdf.

[3] Χρύσης Γεωργίου, Σημειώσεις ,Edmonds Karp, Μαθήματος ΕΠΛ336

Παράρτημα Α

Στο παράρτημα αυτό περιλαμβάνονται οι υλοποιήσεις των αλγορίθμων Ford Fulkerson[1], Edmonds Karp[3], Scaling Max Flow[1], Preflow Push[1] και Scaling Preflow Push[2]. Επίσης υπάρχει υλοποίηση μιας από τις στοχευόμενες τοπολογίες.

Ford Fulkerson[1]

```
public class FordFulkerson {
	static void readFromFile(int numberOfNodes, int[][] netflow)
				throws FileNotFoundException {
		File file = new File("graph.txt");
		Scanner scanner = new Scanner(file);
		String line = null;
		line = scanner.nextLine();
		String[] file1tags;

		String delimiter = ",";
		file1tags = line.split(delimiter);
		int p = 0;

		for (int i = 1; i < numberOfNodes + 1; i++) {

			for (int j = 1; j < numberOfNodes + 1; j++) {

				int current = Integer.parseInt(file1tags[p]);
				netflow[i][j] = current;

				p++;
			}
		}

	}

	// FordFulkerson

	public static boolean isPathFF(int s, int t, int netflow[][], int[] parent,
			Queue<Integer> queue, boolean[] visited, int numOfNodes)

	{
```

```
int counter, item;
boolean pathFound = false;

// initializations

Arrays.fill(parent, 1, numOfNodes + 1, -1);
Arrays.fill(visited, 1, numOfNodes + 1, false);
((LinkedList<Integer>) queue).push(s);

parent[s] = -1;
visited[s] = true;

while (!queue.isEmpty())

{

        counter = 1;
        item = ((LinkedList<Integer>) queue).pop();

        while (counter <= numOfNodes)

        {
                if (netflow[item][counter] > 0 && !visited[counter])

                {
                        parent[counter] = item;

                        ((LinkedList<Integer>) queue).push(counter);

                        visited[counter] = true;
                }

                counter++;
        }
}
if (visited[t])
        pathFound = true;

return pathFound;
```

```
}

static int augmentFF(int s, int t, int[][] residualnet, int[] parent) {

        ArrayList<Integer> list = new ArrayList<Integer>();
        int pathFlow = Integer.MAX_VALUE, u, v = t;

        // find the bottleneck of the path
        while (v != s) {

                u = parent[v];
                pathFlow = Math.min(pathFlow, residualnet[u][v]);
                v = parent[v];

        }

        v = t;
        while (v != s) {

                u = parent[v];
                list.add(u);
                residualnet[u][v] -= pathFlow;// forward
                residualnet[v][u] += pathFlow;// backward
                v = parent[v];
        }

        return pathFlow;

}

static int AlgorithmFF(int source, int destination, int numberOfNodes,
                int[][] netflow) throws FileNotFoundException {

        int[] parent = new int[numberOfNodes + 1];
        Queue<Integer> queue = new LinkedList<Integer>();
        boolean[] visited = new boolean[numberOfNodes + 1];

        int maxFlow = 0;
```

```
        boolean flag = false;
        flag = isPathFF(source, destination, netflow, parent, queue, visited,
                        numberOfNodes);

        while (flag)

        {
            maxFlow += augmentFF(source, destination, netflow, parent);

            flag = isPathFF(source, destination, netflow, parent, queue,
                            visited, numberOfNodes);
        }

        return maxFlow;

    }

    public static void main(String[] args) throws FileNotFoundException

    {
        int numberOfNodes = 6;
        int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
        readFromFile(numberOfNodes, netflow);

        long start = System.currentTimeMillis();

        int source, sink, maxFlow = 0;
        source = 1;
        sink = numberOfNodes;

        maxFlow = AlgorithmFF(source, sink, numberOfNodes, netflow);
        System.out.println(maxFlow);

        long end = System.currentTimeMillis();

        System.out.println(((end - start)));

    }
}
```

Edmonds Karp[3]

```
public class EdmondsKarp {

	static void readFromFile(int numberOfNodes, int[][] netflow)
				throws FileNotFoundException {
		File file = new File("graph.txt");
		Scanner scanner = new Scanner(file);
		String line = null;
		line = scanner.nextLine();
		String[] file1tags;

		String delimiter = ",";
		file1tags = line.split(delimiter);
		int p = 0;

		for (int i = 1; i < numberOfNodes + 1; i++) {

			for (int j = 1; j < numberOfNodes + 1; j++) {

				int current = Integer.parseInt(file1tags[p]);
				netflow[i][j] = current;

				p++;
			}
		}

	}

	// Edmonds Karp
	public static boolean isPathEK(int s, int t, int netflow[][], int[] parent,
			int numberOfNodes, Queue<Integer> queue, boolean[] visited)

	{
		int counter = 0, item;
		boolean pathFound = false;

		// initializations
```

```
        Arrays.fill(parent, 1, numberOfNodes + 1, -1);
        Arrays.fill(visited, 1, numberOfNodes + 1, false);
        queue.offer(s);

        parent[s] = -1;
        visited[s] = true;

        while (!queue.isEmpty())

        {
                counter = 1;
                item = queue.poll();
                while (counter <= numberOfNodes)

                {
                        if (netflow[item][counter] > 0 && !visited[counter])

                        {
                                parent[counter] = item;
                                queue.offer(counter);

                                visited[counter] = true;

                        }

                        counter++;
                }
        }
        if (visited[t])
                pathFound = true;

        return pathFound;

}

static int augmentEK(int s, int t, int[][] residualnet, int[] parent) {

        ArrayList<Integer> list = new ArrayList<Integer>();
        int pathFlow = Integer.MAX_VALUE, u, v = t;
```

```
            // find the bottleneck of the path
            while (v != s) {

                        u = parent[v];
                        pathFlow = Math.min(pathFlow, residualnet[u][v]);
                        v = parent[v];

            }

            v = t;
            while (v != s) {

                        u = parent[v];
                        list.add(u);
                        residualnet[u][v] -= pathFlow;// forward
                        residualnet[v][u] += pathFlow;// backward
                        v = parent[v];
            }

            return pathFlow;

}

static int AlgorithmEK(int source, int destination, int numberOfNodes,
                        int[][] netflow) throws FileNotFoundException {

            int maxFlow = 0;
            boolean flag = false;
            int[] parent = new int[numberOfNodes + 1];
            ;
            Queue<Integer> queue = new LinkedList<Integer>();
            boolean[] visited = new boolean[numberOfNodes + 1];

            flag = isPathEK(source, destination, netflow, parent, numberOfNodes,
                                    queue, visited);

            while (flag)
```

```
		{
			maxFlow += augmentEK(source, destination, netflow, parent);

			flag = isPathEK(source, destination, netflow, parent,
					numberOfNodes, queue, visited);
		}

		return maxFlow;
	}

	public static void main(String[] args) throws FileNotFoundException

	{
		int numberOfNodes = 6;
		int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
		readFromFile(numberOfNodes, netflow);

		long start = System.currentTimeMillis();

		int source, sink, maxFlow = 0;
		source = 1;
		sink = numberOfNodes;

		maxFlow = AlgorithmEK(source, sink, numberOfNodes, netflow);
		System.out.println(maxFlow);
		long end = System.currentTimeMillis();

		System.out.println(((end - start)));
	}

}
```

Scaling Max Flow[1]

```
public class ScalingMaxFlow {

	static void readFromFile(int numberOfNodes, int[][] netflow)
			throws FileNotFoundException {
		File file = new File("graph.txt");
		Scanner scanner = new Scanner(file);
```

```
            String line = null;
            line = scanner.nextLine();
            String[] file1tags;

            String delimiter = ",";
            file1tags = line.split(delimiter);
            int p = 0;

            for (int i = 1; i < numberOfNodes + 1; i++) {

                  for (int j = 1; j < numberOfNodes + 1; j++) {

                        int current = Integer.parseInt(file1tags[p]);
                        netflow[i][j] = current;

                        p++;
                  }
            }

}

// Scaling Max Flow
public static boolean isPathSMF(int s, int t, int netflow[][], int D,
                  int[] parent, boolean[] visited, int numberOfNodes,
                  Queue<Integer> queue)

{
            int counter, item;
            boolean pathFound = false;

            // initializations

            Arrays.fill(parent, 1, numberOfNodes + 1, -1);
            Arrays.fill(visited, 1, numberOfNodes + 1, false);

            ((LinkedList<Integer>) queue).push(s);
            parent[s] = -1;
            visited[s] = true;
```

```
            while (!queue.isEmpty())

            {
                counter = 1;
                item = ((LinkedList<Integer>) queue).pop();
                while (counter <= numberOfNodes)

                {
                    if (netflow[item][counter] > 0 && !visited[counter]
                                        && netflow[item][counter] >= D)

                    {
                        parent[counter] = item;
                        ((LinkedList<Integer>) queue).push(counter);
                        visited[counter] = true;
                    }

                    counter++;
                }
            }
            if (visited[t])
                pathFound = true;

            return pathFound;

}

static int augmentSMF(int s, int t, int[][] residualnet, int[] parent) {

            ArrayList<Integer> list = new ArrayList<Integer>();
            int pathFlow = Integer.MAX_VALUE, u, v = t;

            // find the bottleneck of the path
            while (v != s) {

                u = parent[v];
                pathFlow = Math.min(pathFlow, residualnet[u][v]);
                v = parent[v];
```

```
		}

		v = t;
		while (v != s) {

			u = parent[v];
			list.add(u);
			residualnet[u][v] -= pathFlow;// forward
			residualnet[v][u] += pathFlow;// backward
			v = parent[v];
		}

		return pathFlow;

}

static int AlgorithmSMF(int source, int destination, int numberOfNodes,
			int[][] netflow) throws FileNotFoundException {

		int[] parent = new int[numberOfNodes + 1];
		Queue<Integer> queue = new LinkedList<Integer>();
		boolean[] visited = new boolean[numberOfNodes + 1];

		int maxFlow = 0;
		boolean flag = false;
		int D = 1;
		int maxCe = -1;
		int counter = 1;
		int pre = 0;

		for (int i = 1; i < numberOfNodes + 1; i++) {

			if (netflow[1][i] > maxCe) {
				maxCe = netflow[1][i];
			}
		}

		while (D <= maxCe) {
```

```
                pre = D;
                counter++;
                D = (int) Math.pow(2, counter);

            }

            D = pre;

            while (D >= 1) {
                flag = isPathSMF(source, destination, netflow, D, parent, visited,
                                numberOfNodes, queue);

                while (flag)

                {
                    maxFlow += augmentSMF(source, destination, netflow, parent);

                    flag = isPathSMF(source, destination, netflow, D, parent,
                                    visited, numberOfNodes, queue);
                }

                D = D / 2;
            }
            return maxFlow;

    }

    public static void main(String[] args) throws FileNotFoundException

    {
            int numberOfNodes = 6;
            int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
            readFromFile(numberOfNodes, netflow);

            long start = System.currentTimeMillis();

            int source, sink, maxFlow = 0;
            source = 1;
            sink = numberOfNodes;
```

```
                maxFlow = AlgorithmSMF(source, sink, numberOfNodes, netflow);

                System.out.println(maxFlow);
                long end = System.currentTimeMillis();

                System.out.println(((end - start)));

        }

}
```

Preflow Push[1]

```
public class PreflowPush {

        public static List<Integer> exNodes = new ArrayList<Integer>();

        static void readFromFile(int numberOfNodes, int[][] netflow)
                        throws FileNotFoundException {
                File file = new File("graph.txt");
                Scanner scanner = new Scanner(file);
                String line = null;
                line = scanner.nextLine();
                String[] file1tags;
                String delimiter = ",";
                file1tags = line.split(delimiter);
                int p = 0;

                for (int i = 1; i < numberOfNodes + 1; i++) {

                        for (int j = 1; j < numberOfNodes + 1; j++) {

                                int current = Integer.parseInt(file1tags[p]);
                                netflow[i][j] = current;

                                p++;
                        }
                }
```

```
}

// Preflow Push

static void isExcessPP(int numberOfNodes, int f[][], int excess[],
                        int source, int sink, List<Integer> exNodes) {
                int in = 0, out = 0;
                for (int i = 1; i < numberOfNodes + 1; i++) {
                        for (int j = 1; j < numberOfNodes + 1; j++) {

                                in = in + f[j][i];
                                out = out + f[i][j];

                        }
                        if (in > out && i != source && i != sink && !exNodes.contains(i)) {
                                excess[i] = in - out;
                                exNodes.add(i);
                        }
                        in = 0;
                        out = 0;
                }
}

static void PushRelabelPP(List<Integer> exNodes, int numberOfNodes,
                        int[][] netflow, int[][] residualnet, int[] excess, int[][] f,
                        int source, int sink, int[] h) {
                int exNode = exNodes.get(0);
                boolean flag = false;
                for (int i = numberOfNodes; i >= 1; i--) {

                        if (h[exNode] > h[i] && excess[exNode] > 0
                                        && residualnet[exNode][i] > 0) {// is in Gf
                                flag = true;
                                if (netflow[exNode][i] - f[exNode][i] > 0) {// forward

                                        int d = Math.min(excess[exNode], netflow[exNode][i]
                                                        - f[exNode][i]);
                                        f[exNode][i] = f[exNode][i] + d;
                                        excess[exNode] = excess[exNode] - d;
```

```
                                        excess[i] = excess[i] + d;
                                        residualnet[exNode][i] = residualnet[exNode][i] - d;
                                        residualnet[i][exNode] = residualnet[i][exNode] + d;

                                } else {// backward

                                        int d = Math.min(excess[exNode], f[i][exNode]);
                                        f[i][exNode] = f[i][exNode] - d;

                                        residualnet[exNode][i] = residualnet[exNode][i] - d;
                                        residualnet[i][exNode] = residualnet[i][exNode] + d;
                                        excess[exNode] = excess[exNode] - d;
                                        excess[i] = excess[i] + d;

                                }

                        }

                }

                if (excess[exNode] == 0) {

                        exNodes.remove(0);
                        isExcessPP(numberOfNodes, f, excess, source, sink, exNodes);

                }
                if (flag == false && excess[exNode] > 0) {

                        h[exNode] = h[exNode] + 1;
                }

        }

        static int AlgorithmPP(int source, int sink, int numberOfNodes,
                        int[][] netflow) throws FileNotFoundException {

                List<Integer> exNodes = new ArrayList<Integer>();

                int residualnet[][] = new int[numberOfNodes + 1][numberOfNodes + 1];
```

```
int f[][] = new int[numberOfNodes + 1][numberOfNodes + 1];
int h[] = new int[numberOfNodes + 1];
int excess[] = new int[numberOfNodes + 1];

for (int i = 0; i < numberOfNodes + 1; i++) {
        for (int j = 0; j < numberOfNodes + 1; j++) {
                residualnet[i][j] = netflow[i][j];
        }
}
// initializations
h[source] = numberOfNodes;
for (int i = 2; i <= numberOfNodes; i++) {
        h[i] = 0;
}
for (int i = 1; i <= numberOfNodes; i++) {
        for (int j = 0; j <= numberOfNodes; j++) {
                if (i == source) {
                        f[i][j] = netflow[i][j];
                        residualnet[i][j] = 0;
                        residualnet[j][i] = f[i][j];

                } else {
                        f[i][j] = 0;
                }
        }
}

isExcessPP(numberOfNodes, f, excess, source, sink, exNodes);

while (!exNodes.isEmpty()) {
        PushRelabelPP(exNodes, numberOfNodes, netflow, residualnet, excess,
                        f, source, sink, h);

}

int maxflow = 0;
for (int i = 1; i < numberOfNodes + 1; i++) {
        maxflow = maxflow + f[i][numberOfNodes];
}
```

```
		return maxflow;
	}

	public static void main(String[] args) throws FileNotFoundException

	{
		int numberOfNodes = 4;
		int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
		readFromFile(numberOfNodes, netflow);

		long start = System.currentTimeMillis();

		int source, sink, maxFlow = 0;
		source = 1;
		sink = numberOfNodes;

		maxFlow = AlgorithmPP(source, sink, numberOfNodes, netflow);
		System.out.println(maxFlow);
		long end = System.currentTimeMillis();

		System.out.println(((end - start)));

	}
}
```

Scaling Preflow Push[2]

```
public class ScalingPreflowPush {
	static class lists {
		List<Integer> distance;

		public lists() {
			distance = new ArrayList<Integer>();
		}
	}

	static void readFromFile(int numberOfNodes, int[][] netflow)
				throws FileNotFoundException {
		File file = new File("graph.txt");
		Scanner scanner = new Scanner(file);
```

```
            String line = null;
            line = scanner.nextLine();
            String[] file1tags;
            String delimiter = ",";
            file1tags = line.split(delimiter);
            int p = 0;

            for (int i = 1; i < numberOfNodes + 1; i++) {

                    for (int j = 1; j < numberOfNodes + 1; j++) {

                            int current = Integer.parseInt(file1tags[p]);
                            netflow[i][j] = current;

                            p++;
                    }
            }

}

// Scaling Preflow Push

static void isExcessSPP(int numberOfNodes, int f[][], int excess[],
                    int source, int sink, int D, lists dists[], int d[]) {
            int in = 0, out = 0;
            for (int i = 1; i < numberOfNodes + 1; i++) {
                    for (int j = 1; j < numberOfNodes + 1; j++) {

                            in = in + f[j][i];
                            out = out + f[i][j];

                    }
                    if (in > out && i != source && i != sink
                                    && !dists[d[i]].distance.contains(i) && (in - out) > D / 2) {
                            excess[i] = in - out;
                            dists[d[i]].distance.add(i);

                    }
                    in = 0;
```

```
                out = 0;
        }

}

static int PushRelabelSPP(int numberOfNodes, int[][] residualnet,
                lists dists[], int excess[], int[][] f, int source, int sink,
                int[] d, int level, int D) {
        int exNode = dists[level].distance.get(0);
        boolean flag = false;// found
        int arc = 0;
        for (int z = 1; z < numberOfNodes + 1; z++) {

                if (d[exNode] == d[z] + 1 && residualnet[exNode][z] > 0
                                && z != exNode) {
                        arc = z;
                        flag = true;
                        break;
                }

        }

        if (flag == true) {// is in Gf

                if (residualnet[exNode][arc] > 0) {

                        int delta1 = Math.min(excess[exNode], residualnet[exNode][arc]);
                        int delta = Math.min(delta1, D - excess[arc]);
                        f[exNode][arc] = f[exNode][arc] + delta;
                        excess[exNode] = excess[exNode] - delta;

                        if (arc != source && arc != sink) {
                                excess[arc] = excess[arc] + delta;
                        } else {
                                excess[arc] = 0;
                        }
                        residualnet[exNode][arc] = residualnet[exNode][arc] - delta;
                        residualnet[arc][exNode] = residualnet[arc][exNode] + delta;
```

```
                }
                if (excess[exNode] <= D / 2) {

                        dists[d[exNode]].distance.remove(0);
                }
                if (arc != source && arc != sink && excess[arc] > D / 2
                                && !dists[d[exNode]].distance.contains(arc)) {
                        dists[d[arc]].distance.add(dists[d[arc]].distance.size(), arc);// add
                                                                                // to
                                                                                // tail
                        level--;
                }

        }

        if (flag == false) {

                dists[d[exNode]].distance.remove(0);
                int min = Integer.MAX_VALUE;
                for (int j = 1; j < numberOfNodes + 1; j++) {

                        if (residualnet[exNode][j] > 0 && d[j] < min) {
                                min = d[j];

                        }
                }

                d[exNode] = min + 1;
                dists[d[exNode]].distance.add(dists[d[exNode]].distance.size(),
                                exNode);// add
                                                        // to
                                                        // tail

        }
        return level;
}

    static int AlgorithmSPP(int source, int sink, int numberOfNodes,
```

```
        int netflow[][]) throws FileNotFoundException {

int residualnet[][] = new int[numberOfNodes + 1][numberOfNodes + 1];
int f[][] = new int[numberOfNodes + 1][numberOfNodes + 1];
int d[] = new int[numberOfNodes + 1];
int excess[] = new int[numberOfNodes + 1];
int D = 1;
int maxCe = -1;
int counter = 1;
int pre = 0;
int level = 1;
lists dists[] = new lists[2 * numberOfNodes];

for (int i = 0; i < 2 * numberOfNodes; i++) {
        dists[i] = new lists();
}

for (int i = 1; i < numberOfNodes + 1; i++) {

        if (netflow[1][i] > maxCe) {
                maxCe = netflow[1][i];
        }
}

while (D <= maxCe) {

        pre = D;
        counter++;
        D = (int) Math.pow(2, counter);

}

D = pre;

for (int i = 0; i < numberOfNodes + 1; i++) {
        for (int j = 0; j < numberOfNodes + 1; j++) {
                residualnet[i][j] = netflow[i][j];
        }
```

```
}
// initializations
d[source] = numberOfNodes;
d[numberOfNodes] = 0;
for (int i = 2; i < numberOfNodes; i++) {
        d[i] = 1;
}
for (int i = 1; i <= numberOfNodes; i++) {
        for (int j = 0; j <= numberOfNodes; j++) {
                if (i == source) {
                        // send flow
                        f[i][j] = netflow[i][j];
                        // reversion
                        residualnet[i][j] = 0;
                        residualnet[j][i] = f[i][j];

                } else {
                        f[i][j] = 0;
                }
        }
}

while (D >= 1) {
        isExcessSPP(numberOfNodes, f, excess, source, sink, D, dists, d);

        level = 1;
        while (level < 2 * numberOfNodes) {

                if (dists[level].distance.isEmpty()) {
                        level++;

                } else {
                        level = PushRelabelSPP(numberOfNodes, residualnet, dists,
                                        excess, f, source, sink, d, level, D);

                }

        }
        D = D / 2;
```

```
                }// while D

                int maxflow = 0;
                for (int i = 1; i < numberOfNodes + 1; i++) {
                        maxflow = maxflow + f[i][numberOfNodes];
                }
                return maxflow;
        }

        public static void main(String[] args) throws FileNotFoundException

        {

                int numberOfNodes = 4;
                int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
                readFromFile(numberOfNodes, netflow);

                long start = System.currentTimeMillis();

                int source, sink, maxFlow = 0;
                source = 1;
                sink = numberOfNodes;
                maxFlow = AlgorithmSPP(source, sink, numberOfNodes, netflow);
                System.out.println(maxFlow);

                long end = System.currentTimeMillis();

                System.out.println(((end - start)));

        }

}
```

Τοπολογία

```
public class Topology1 {
	public static void main(String[] args) throws IOException {
		int numberOfNodes = 20;

		int source = 1, sink = numberOfNodes;
		int[][] netflow = new int[numberOfNodes + 1][numberOfNodes + 1];
		int[][] visited = new int[numberOfNodes + 1][numberOfNodes + 1];
		// calculate the rows

		int fullRows = 0, notFullRow = 0, Nodes = 0;

		Nodes = numberOfNodes - 2;// except source & sink

		fullRows = Nodes / 3;// not too far paths -> numberOfFull Rows
		notFullRow = Nodes % 3;// #nodes at last row

		int totalRows = 0;
		if (notFullRow != 0) {
			totalRows = source + fullRows + 1;
		} else {
			totalRows = source + fullRows;
		}

		for (int i = 1; i < fullRows + 1; i++) {
			//System.out.println();
			int pre = source;
			int flag = 0;

			for (int j = source + i; j < source + i + 2 * totalRows + 1; j = j
					+ totalRows - 1) {
				if (j == 2) {
					flag = 1;
				}
				visited[pre][j] = 2;

				pre = j;
				if (pre - 1 != source && flag != 1) {
```

```
                                visited[pre][pre - 1] = 3;
                        }


                }
                if (pre != sink) {
                        visited[pre][sink] = 2;
                }
        }

        int preRow = fullRows + 2;
        int pre = source;
        for (int j = preRow; j < preRow + (notFullRow * 3); j = j + 3) {

                visited[pre][j] = 2;
                //System.out.print(j + " -> ");
                pre = j;
                visited[pre][pre - 1] = 3;

        }
        if (pre != sink) {
                visited[pre][sink] = 2;
        }

        for (int i = 2; i < fullRows + 1; i++) {
                int current = source + i + 6;

                visited[current][current - 1] = 3;

        }




        for (int i = 2; i < numberOfNodes + 1; i++) {
                for (int j = 2; j < numberOfNodes + 1; j++) {
                        if (visited[i][j] == 2) {
                                netflow[i][j] =10 ;
                        }
                }
```

```
}
netflow[1][2] = 1;
for (int i = 2; i < numberOfNodes + 1; i++) {
        if (visited[i][sink] == 2) {
                netflow[i][sink] = 1;

        }
}

for (int j = 3; j < totalRows + 1; j++) {
        if (visited[1][j] == 2) {
                netflow[1][j] = 10;
        }
}
for (int i = 1; i < numberOfNodes + 1; i++) {
        for (int j = 1; j < numberOfNodes + 1; j++) {
                if (visited[i][j] == 3) {
                        netflow[i][j] = 10;
                }
        }
}

for (int i = 1; i < numberOfNodes + 1; i++) {
        if (visited[i][sink] == 2 && i != totalRows * 2) {
                netflow[i][i - 1] = 10;
        }

}

// for the pre row nodes first column
int to = 10;
for (int i = totalRows; i >=3; i--) {
        //if (visited[i][i - 1] == 3) {

                netflow[i][i - 1] = 1;
                to = to + 10;

        //}
}
```

```
                // for the first Row
                pre = source + 1;
                for (int j = pre + totalRows - 1; j < source + 2 * totalRows + 2; j = j
                                + totalRows - 1) {

                        netflow[pre][j] = to-10  + 1;
                        pre = j;

                }
                netflow[pre][sink] = to-10 + 1;
                netflow[2][2+totalRows-1]=totalRows-1;

                // write the capacity of network to graph.txt
                File file = new File("graph.txt");

                // if file doesn't exists, then create it
                if (!file.exists()) {
                        file.createNewFile();
                }

                FileWriter writer = new FileWriter(file);
                BufferedWriter writenet = new BufferedWriter(writer);

                for (int p = 1; p < numberOfNodes + 1; p++) {
                        for (int j = 1; j < numberOfNodes + 1; j++) {
                                writenet.write(String.valueOf(netflow[p][j]));
                                writenet.write(",");

                        }
                }

                writenet.close();

        }

}
```

Παράρτημα Β

Στο παράρτημα αυτό υπάρχει το batch file μέσω του οποίου καλούνταν οι αλγόριθμοι και η τοπολογία στην οποία δούλευαν. Στις εντολές αυτές εμπεριέχεται και η εντολή μέσω της οποίας μπορούσα να εκμεταλλεύομαι όλο το heap space της μηχανής στην οποία δούλευα.

```
#!/bin/bash

#now start to run the program

javac /home/kyriakic/test/GenerateNetworks.java

java -Xms4g -Xmx6g -d64 GenerateNetworks

javac /home/kyriakic/test/FordFulkerson.java

java -Xms4g -Xmx6g -d64 FordFulkerson

javac /home/kyriakic/test/EdmondsKarp.java

java -Xms4g -Xmx6g -d64 EdmondsKarp

javac /home/kyriakic/test/ScalingMaxFlow.java

java -Xms4g -Xmx6g -d64 ScalingMaxFlow

javac /home/kyriakic/test/PreflowPush.java

java -Xms4g -Xmx6g -d64 PreflowPush

javac /home/kyriakic/test/ScalingPreflowPush.java

java -Xms4g -Xmx6g -d64 ScalingPreflowPush
```