ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μελέτη και υλοποίηση μεθόδων επικαλύπτοντων δένδρων ελάχιστης έκτασης για προσομοίωση μεγάλης κλίμακας γραμμικών κυκλωμάτων

Design and development of a low-stretch spanning tree preconditioner for large-scale circuit simulation

Διπλωματική Εργασία

Καραφέρη Ελευθερία

Επιβλέποντες Καθηγητές: Ευμορφόπουλος Νέστωρ
Επίκουρος Καθηγητής

Σταμούλης Γεώργιος
Καθηγητής

Βόλος, Ιούνιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μελέτη και υλοποίηση μεθόδων επικαλύπτοντων δένδρων ελάχιστης έκτασης για προσομοίωση μεγάλης κλίμακας γραμμικών κυκλωμάτων

Διπλωματική Εργασία

Καραφέρη Ελευθερία

Επιβλέποντες Καθηγητές: Ευμορφόπουλος Νέστωρ
Επίκουρος Καθηγητής

Σταμούλης Γεώργιος
Καθηγητής

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την Ιουνίου 2014

……	……
Ν.Ευμορφόπουλος	Γ.Σταμούλης
Επίκουρος Καθηγητής	Καθηγητής

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

……..

Ελευθερία Καραφέρη
Διπλωματούχος Μηχανικών Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων
Πανεπιστημίου Θεσσαλίας

Σα βγεις στον πηγαιμό για την Ιθάκη,
να εύχεσαι νάναι μακρύς ο δρόμος,
γεμάτος περιπέτειες, γεμάτος γνώσεις.
Τους Λαιστρυγόνας και τους Κύκλωπας,
τον θυμωμένο Ποσειδώνα μη φοβάσαι,
τέτοια στον δρόμο σου ποτέ σου δεν θα βρεις,
αν μέν' η σκέψις σου υψηλή, αν εκλεκτή
συγκίνησις το πνεύμα και το σώμα σου αγγίζει.
…
Κι αν πτωχική την βρεις, η Ιθάκη δεν σε γέλασε.
Έτσι σοφός που έγινες, με τόση πείρα,
ήδη θα το κατάλαβες η Ιθάκες τι σημαίνουν.

Στην οικογένειά μου και στους φίλους μου.

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τον κ. Νέστορα Ευμορφόπουλο για την εμπιστοσύνη που επέδειξε στο πρόσωπό μου και την άψογη συνεργασία και συνεχή καθοδήγηση καθ όλη τη διάρκεια της διπλωματικής εργασίας. Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Γεώργιο Σταμούλη, χωρίς την ψυχολογική αλλά και έμπρακτη υποστήριξή του οποίου, δεν θα ήμουν εδώ που είμαι σήμερα.

Στην συνέχεια θα ήθελα να ευχαριστήσω τους φίλους και συνεργάτες του εργαστηρίου Ε5 που με την ευχάριστη διάθεσή τους συνέβαλαν στην ομαλή εξέλιξη της εργασίας. Ιδιαίτερα όμως οφείλω ένα μεγάλο ευχαριστώ στον διδακτορικό φοιτητή Κωνσταντή Νταλούκα για τις ουσιώδεις υποδείξεις και παρεμβάσεις που βοήθησαν σε μεγάλο βαθμό την εκπόνηση της παρούσας εργασίας. Ευχαριστώ επίσης τους φίλους μου, που όλα αυτά τα χρόνια ήτανε δίπλα μου και μου στάθηκαν σε όλες τις στιγμές.

Δεν θα μπορούσα όμως να ξεχάσω και τα άτομα χωρίς τα οποία δεν θα ολοκλήρωνα ούτε την διπλωματική μου εργασία αλλά ούτε και τις σπουδές μου. Την οικογένειά μου, η οποία απάλυνε τα όποια άγχη και ανασφάλειες μου, πίστεψε σε μένα και με στήριξε με όλους τους τρόπους στο να ολοκληρώσω αυτό που άρχισα. Αυτοί είναι και θα είναι η δύναμη που με παρακινεί στο να συνεχίζω.

Καραφέρη Ελευθερία

Βόλος, 2014

Περιεχόμενα

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος Εικόνων

Κατάλογος Συντομογραφιών

BE backward Euler

BiCG biconjugate gradient

CG conjugate gradients

DFS depth-first search

GMRES generalized minimal residual

MAST minimum average stretch tree

MMST minimum max stretch tree

MNA modified nodal analysis

PDN power delivery network

SDD symmetric diagonally dominant

SOR successive overrelaxation

SSOR symmetric successive overrelaxation

SPD symmetric positive definite

TR trapezoidal

Περίληψη

Η δραστικά αυξανόμενη πολυπλοκότητα στο σχεδιασμό και την προσομοίωση των δικτύων ηλεκτρικής ενέργειας πολύ μεγάλης κλίμακας, καθιστά σχεδόν αναγκαία πλέον την χρήση preconditioned επαναληπτικών αλγορίθμων για επίτευξη πιο γρήγορης ανάλυση του δικτύου ηλεκτρικής ενέργειας. Σε αυτή την διπλωματική εργασία θα αναλύσουμε την θεωρία γράφων και θα εφαρμόσουμε δυο πολλά υποσχόμενους support-graph preconditioned επαναληπτικούς αλγόριθμους, τον maximum spanning tree και τον low-stretch tree preconditioner, ανεξάρτητα τον έναν από τον άλλον. Αφού υλοποιήσουμε τους δυο αλγόριθμους, τους εφαρμόζουμε πάνω στον γράφο που αναπαριστά το ηλεκτρικό κύκλωμα και χρησιμοποιούμε τα αποτελέσματα τους για πιο αποδοτική λύση του γραμμικού συστήματος της μορφής $Ax = b$. Τα αποτελέσματα τους είναι πολύ ενθαρρυντικά, καθώς επιτυγχάνουν σημαντική επιτάχυνση στον χρόνο σύγκλισης της μεθόδου συζυγών κλίσεων (CG), στην οποία εφαρμόστηκαν.

Λέξεις Κλειδιά:

δίκτυο ηλεκτρικής ενέργειας, support-graph, preconditioner, θεωρία γράφων, maximum spanning tree, low-stretch tree, γραμμικό σύστημα, χρόνος σύγκλισης, μέθοδος συζυγών κλίσεων

Abstract

The drastically growing complexity in the very large-scale power grid design and simulation makes it almost necessary now using preconditioned iterative algorithms to achieve more rapid resolution of the power grid. In this thesis we will analyze the graph theory and apply two highly promising support-graph preconditioned iterative algorithms, the maximum spanning tree and low-stretch tree preconditioner, independently of one another. Once we have implemented the two algorithms, we apply them on the graph that represents the power grid and use the results for a more efficient solution of the linear system of the form Ax = b. The results are very encouraging, and achieve significant acceleration in the convergence time of the conjugate gradient method (CG), to which were applied.

Keywords:

power grid, support-graph, preconditioner, graph theory, maximum spanning tree, low-stretch tree, linear system, convergence time, conjugate gradient method

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή του Προβλήματος

Στην σύγχρονη IC σχεδίαση, τα δίκτυα παροχής ηλεκτρικής ενέργειας (PDNs) έχουν γίνει ιδιαιτέρως σημαντικά, καθώς οι επιδόσεις των κυκλωμάτων όπως η καθυστέρηση και η κατανάλωση ισχύος μπορεί να σχετίζονται άμεσα με τα επίπεδα τροφοδοσίας τάσης. Ενώ η υπάρχουσες εξελίξεις στην τεχνολογία ημιαγωγών επιτρέπουν την ενσωμάτωση πολύ πιο γρήγορων τρανζίστορ σε chip, η αντίσταση της on-chip διασύνδεσης δεν κλιμακώθηκε αντίστοιχα καλά, δημιουργώντας έναν εξαιρετικά κατασταλτικό παράγοντα επίδοσης. Καθώς η σχεδίαση του οικονομικότερου αλλά και αξιόπιστου δικτύου ηλεκτρικής ενέργειας που να πληροί όλες της προδιαγραφές σχεδίασης και ταυτόχρονα να αποφεύγει προβλήματα κατά την διάρκεια του σχεδιασμού δεν είναι πάντα εύκολο, έχουν γίνει πολύ σημαντικές οι γρήγορες προσομοιώσεις και σταδιακές αναλύσεις των δικτύων ηλεκτρικής ενέργειας, έτσι ώστε να εφαρμόζονται πιο εύκολα διορθώσεις και βελτιστοποιήσεις.

Η πλειοψηφία των αλγορίθμων προσομοίωσης εντάσσονται σε δυο κατηγορίες: άμεσες μέθοδοι και επαναληπτικές μέθοδοι. Οι άμεσες μέθοδοι λύνουν το πρόβλημα της προσομοίωσης αλλά απαιτούν πολύ περισσότερη μνήμη για να αποθηκεύσουν τους παράγοντες πινάκων, ενώ οι επαναληπτικές μέθοδοι συνήθως έχουν πιο προσιτές απαιτήσεις μνήμης αλλά υστερούν στο θέμα σύγκλισης. Για τα δίκτυα ηλεκτρικής ενέργειας με κανονικότητα και πλήρης δομή γνωστή εκ των προτέρων, ο preconditioned CG επιλυτής μπορεί να συγκλίνει πολύ γρήγορα και αξιόπιστα όταν είναι καλά σχεδιασμένος και βελτιστοποιημένος.

Προκειμένου να επιτευχθεί λοιπόν μεγαλύτερη αποδοτικότητα εκτέλεσης και μνήμης, μια ποικιλία preconditioned επαναληπτικών αλγορίθμων έχει ερευνηθεί τις τελευταίες δεκαετίες με πολλά υποσχόμενες επιδόσεις. Παρόλο που οι υπάρχοντες preconditioned solvers συνήθως παρουσιάζουν υψηλή αποδοτικότητα στη χρήση της μνήμης, οι συμπεριφορές σύγκλισης τους δεν είναι πάντα ικανοποιητικές.

1.2 Συμβολή της Εργασίας

Σε αυτό το ειδικό θέμα, γίνεται μελέτη, υλοποίηση αλλά και εφαρμογή δυο support-graph preconditioned επαναληπτικών αλγορίθμων, των *maximum spanning tree* preconditiner και *low stretch tree* preconditiner, σε γράφο που αναπαριστά το δίκτυο ηλεκτρικής ενέργειας, με σκοπό την βελτίωση της αποδοτικότητας εκτέλεσης του επιλυτή CG. Η υλοποίηση των αλγορίθμων πραγματοποιήθηκε σε γλώσσα προγραμματισμού C και βασίστηκε στους ψευδοκώδικες τους.

Μετά την εφαρμογή των δύο preconditioners σε netlist ηλεκτρικού κυκλώματος αποτελούμενο από αντιστάσεις και πηγές ρεύματος, παρατηρήθηκαν εξαιρετικά αποτελέσματα όσον αφορά την μείωση του αριθμού επαναλήψεων της μεθόδου επίλυσης *συζυγών κλίσεων* (CG) καθώς και του χρόνου εύρεσης της λύσης.

Πιο συγκεκριμένα, σε σύγκριση με τον Jacobi preconditioner, με χρήση του *maximum spanning tree* preconditioner σημειώθηκε 5 % μείωση των μη μηδενικών στοιχείων του πίνακα αγωγιμοτήτων, 13x speedup στον αριθμό επαναλήψεων της CG καθώς και 3x speedup στον χρόνο σύγκλισης της μεθόδου.

Όσον αφορά την εφαρμογή του low stretch preconditioner, παρατηρήθηκαν εξίσου αισιόδοξα αποτελέσματα με 20 % μείωση των μη μηδενικών στοιχείων του πίνακα αγωγιμοτήτων, 4x speedup στον αριθμό επαναλήψεων της CG αλλά και 3x speedup στον χρόνο σύγκλισης της μεθόδου.

1.3 Διάρθρωση της Διπλωματικής Εργασίας

Η διάρθρωση της παρούσας διπλωματικής εργασίας έχει ως εξής.

Στο Κεφάλαιο 2 παραθέτεται το απαραίτητο υπόβραθο για την κατανόηση της μοντελοποίησης και της ανάλυσης των δικτύων ηλεκτρικής ενέργειας καθώς και των υπαρχόντων μεθόδων επίλυσης των γραμμικών τους εξισώσεων.

Στο Κεφάλαιο 3 αναλύεται η έννοια του preconditioning, η χρήση του αλλά και η σημαντικότητά του στην αποδοτικότερη λύση των γραμμικών συστημάτων.

Στο Κεφάλαιο 4 γίνεται μια επισκόπηση στις βασικές αρχές της θεωρίας των γράφων όσον αφορά τους preconditioners και τα ηλεκτρικά κυκλώματα.

Στο Κεφάλαιο 5 πραγματοποιείται η μελέτη του *maximum spanning tree* preconditioner και αναλύεται ο τρόπος υλοποίησής και χρήσης του.

Στο Κεφάλαιο 6 αναλύεται ο *low-stretch tree* preconditioner και παραθέτεται ο τρόπος υλοποίησης και χρήσης του.

Τέλος, στο Κεφάλαιο 7 γίνεται η παρουσίαση των αποτελεσμάτων εκτέλεσης των δυο preconditioners σε σύγκριση με τον ήδη υπάρχων preconditioner και πραγματοποιείται μια σύντομη ανάλυση μελλοντικής επέκτασης της παρούσας εργασίας.

Κεφάλαιο 2

Μέθοδοι Ανάλυσης και Επίλυσης Δικτύων Ηλεκτρικής Ενέργειας

2.1 Γραμμικό Σύστημα $Ax = b$

Ανάλυση δικτύων ηλεκτρικής ενέργειας

Η ανάλυση δικτύων ηλεκτρικής ενέργειας εντάσσεται συνήθως στις εξής κατηγορίες: DC ή ανάλυση σταθερής κατάστασης, και transient ή μεταβατικές προσομοιώσεις.

Στην DC ανάλυση σταθερής κατάστασης το δίκτυο ηλεκτρικής ενέργειας μοντελοποιείται σαν ένα δίκτυο που αποτελείται καθαρά από αντιστάσεις με πηγές διέγερσης, όπως πηγές ρεύματος και τάσης.

Μπορεί να αποδειχτεί ότι η transient ανάλυση μπορεί να διεξαχθεί εάν γίνει αντικατάσταση των στοιχείων αποθήκευσης ενέργειας, όπως οι πυκνωτές και τα πηνία, με αντίστοιχα μοντέλα που συμπεριλαμβάνουν αντιστάσεις και πηγές τάσης/ρεύματος μέσω των backward Euler (BE) ή Trapezoidal (TR) προσεγγίσεων, και έπειτα λύση του αντίστοιχου DC προβλήματος.

Πιο συγκεκριμένα, στην μεταβατική (transient) ανάλυση το σύστημα ενός γραμμικού κυκλώματος είναι το

$$Gx(t) + \frac{G'\,dx(t)}{dt} = e(t) \qquad (0)$$

όπου x(t) διάνυσμα αγνώστων συναρτήσεων του χρόνου t, $\frac{dx(t)}{dt}$ διάνυσμα παραγώγων αγνώστων εξισώσεων και $e(t)$διάνυσμα γνωστών διεγέρσεων (συναρτήσει του χρόνου). Αν $h_k = t_k - t_{k-1}$ είναι το βήμα δειγματοληψίας, τότε ο υπολογισμός της προσέγγισης $x(t_0)$ $\forall$ t_k γίνεται μέσω μιας από τις ακόλουθες προσεγγίσεις της $\frac{dx(t)}{dt}$ στο (0) :

- Προσέγγιση Backward Euler

  Αν $\frac{dx(t)}{dt} = \frac{1}{h}[x(t_k) - x(t_{k-1})]$, τότε το διαφορικό σύστημα (0) μετατρέπεται στο ακόλουθο

  $$\left(G + \frac{1}{h}C\right)x(t_k) = e(t_k) + \frac{1}{h}Cx(t_{k+1}) \qquad \forall k$$

  όπου $G + \frac{1}{h}C$ ο πίνακας του συστήματος, $e(t_k)$ οι διεγέρσεις και $x(t_{k+1})$ η προηγούμενη λύση του προηγούμενου βήματος.

- Προσέγγιση Trapezoidal

  Αν $\frac{1}{2}\left[\frac{dx(t_k)}{dt} + \frac{dx(t_{k-1})}{dt}\right] = \frac{1}{h}[x(t_k) - x(t_{k-1})]$, τότε το διαφορικό σύστημα (0) μετατρέπεται στο ακόλουθο γραμμικό σύστημα

  $$\left(G + \frac{2}{h}C\right)x(t_k) = e(t_k) + e(t_{k-1}) - \left(G + \frac{2}{h}C\right)x(t_{k-1}) \qquad \forall k$$

  ομοίως με την προηγούμενη μέθοδο.

Συνεπώς, σε αυτή την εργασία θα γίνει εστίαση κυρίως στις μεθόδους DC επίλυσης, αφού η transient ανάλυση μπορεί να επιτευχθεί με παρόμοιο τρόπο.

Γραμμικό σύστημα για DC ανάλυση

Το DC πρόβλημα για ένα τέτοιο δίκτυο ηλεκτρικής ενέργειας αντιστάσεων με *n* κόμβους μπορεί να αναπαρασταθεί χρησιμοποιώντας το παρακάτω γραμμικό σύστημα εξισώσεων, με βάση την ανάλυση κόμβων (NA) [1], [6]:

$$Ax = b \tag{1}$$

όπου το *A* είναι ένας *n x n* συμμετρικός θετικά ορισμένος (SPD) πίνακας αγωγιμότητας, που αντιπροσωπεύει όλες τις διασυνδεδεμένες αντιστάσεις, *x* είναι ένα *n x 1* διάνυσμα συμπεριλαμβανομένων όλων των άγνωστων κόμβων τάσεων και *b* είναι ένα *n x 1* διάνυσμα μοντελοποίησης πηγών διέγερσης, όπως πηγών ρεύματος ή τάσης.

Πιο συγκεκριμένα, οι προσθήκες στον πίνακα A και στο διάνυσμα b γίνονται ως εξής:

- Αντιστάσεις

  Αν η αντίσταση Rx συνδέεται με τους κόμβους x1 και x2 με αντίσταση r, τότε έχουμε συνεισφορές στον πίνακα A στην θέση (x1,x1) και (x2,x2) την τιμή 1/r και στις θέσεις (x1,x2) και (x2,x1) την τιμή -1/r.

- Πηγές ρεύματος

  Αν η πηγή ρεύματος Ix συνδέεται με τους κόμβους x1 και x2 με ρεύμα i, τότε έχουμε συνεισφορές στον διάνυσμα b στην θέση (x1,1) την τιμή –i και στην θέση (x2,1) την τιμή i.

- Πηγές τάσης

  Αν η πηγή τάσης Vx συνδέεται με τους κόμβους x1 και x2 με τάση v, τότε έχουμε συνεισφορές στον πίνακα A στην θέση (x1, (n-1+m2)) και ((n-1+m2) , x1) την τιμή 1, όπου n το πλήθος των κόμβων και m2 οι πηγές τάσης και επαγωγές, και στην θέση (x2, (n-1+m2)) και ((n-1+m2) , x2) την τιμή -1. Επίσης έχουμε συνεισφορά στον διάνυσμα b στην θέση ((n-1+m2), 1) την τιμή v.

Στην περίπτωσή μας, το n είναι πολύ μεγάλο και ο πίνακας A είναι αρκετά αραιός. Αυτό ισχύει γιατί συνήθως το πλήθος των στοιχείων ενός δικτύου ηλεκτρικής ενέργειας μεγάλης κλίμακας είναι μόνο 2-4 φορές το πλήθος των κόμβων. Δεδομένου και του ότι τα στοιχεία συνήθως έχουν 2-4 ακροδέκτες, τότε ο βαθμός κορυφής στους γράφους δικτύου θα είναι μικρός.

Πράγματι, δεδομένου των παραπάνω παρατηρήσεων μπορεί να αποδειχτεί ότι ο μέσος βαθμός κορυφής σε ένα μεγάλο γράφο δικτύου πρέπει να είναι μικρότερος από κάποιο άνω όριο το οποίο είναι της τάξης 6-12. Έτσι, παρόλο που ο αριθμός των ακμών σε ένα γράφο είναι θεωρητικά $O(n^2)$, όταν το n είναι το πλήθος των κόμβων, στην πραγματικότητα είναι μόνο $O(n)$ για κυκλώματα μεγάλης κλίμακας.

Κατά συνέπεια, οι πίνακες κυκλωμάτων είναι σε μεγάλο βαθμό αραιοί και αυτό μπορούμε να το εκμεταλλευτούμε χρησιμοποιώντας τις *τεχνικές αραιών πινάκων* έτσι ώστε να επιταχύνουμε την προσομοίωση του κυκλώματος.

2.2 Τεχνικές Αραιών Πινάκων

Με τις *τεχνικές αραιών πινάκων* οι εισαγωγές πίνακα μηδενικής τιμής δεν αποθηκεύονται και χρησιμοποιούνται στον μικρότερο δυνατό βαθμό.

Οι σύγχρονες τεχνικές κάνουν χρήση δυο δομών δεδομένων:

1. Η μορφή triplet είναι εύκολη στην δημιουργία αλλά όχι και η πιο αποδοτική στην χρήση.
2. Η compressed-column μορφή είναι πιο εύχρηστη και χρησιμοποιείται στους περισσότερους αραιούς πίνακες, αλλά πιο δύσκολο να δημιουργηθεί.

Μια συνήθης ροή είναι να προσδιοριστεί ο πίνακας σε triplex μορφή και έπειτα να μετατραπεί σε compressed-column μορφή για την μετέπειτα επεξεργασία του.

Triplet μορφή

Μια δομή δεδομένων triplet μορφής για έναν m x n πίνακα πραγματικών αριθμών A, με n_z μη μηδενικά στοιχεία, αποτελείται από δυο μονοδιάστατους πίνακες ακεραίων και έναν μονοδιάστατο πίνακα πραγματικών αριθμών, όπως φαίνεται παρακάτω(αν υποθέσουμε ότι ο δείκτης θέσης ξεκινά απ το 1):

- Πίνακας ακεραίων: $r[1], r[2], ..., r[n_z] \in \{1, 2, ..., m\}$
- Πίνακας ακεραίων: $c[1], c[2], ..., c[n_z] \in \{1, 2, ..., n\}$
- Πίνακας πραγματικών: $x[1], x[2], ..., x[n_z] \in \mathbb{R}$ *όπου* $x[i]=A(r[i], r[c])$

Οι πίνακες r και c δείχνουν την θέση ενός στοιχείου στην γραμμή και στήλη του πίνακα αντίστοιχα και ο x περιέχει την τιμή του εκάστοτε στοιχείου, όπως φαίνεται στο παρακάτω παράδειγμα:

Ο πίνακας $A = \begin{bmatrix} 5.6 & 0 & 1.8 & 0 \\ 2.4 & 1.5 & 0 & -0.2 \\ 0 & 1.6 & 4.0 & 0 \\ 2.8 & -3.0 & 0 & 1.7 \end{bmatrix}$ (2)

έχει την παρακάτω δομή triplet μορφής:

$r = [3\ \ 2\ \ 4\ \ 1\ \ 2\ \ 4\ \ 4\ \ 2\ \ 1\ \ 3]$
$c = [3\ \ 1\ \ 4\ \ 3\ \ 2\ \ 1\ \ 2\ \ 4\ \ 1\ \ 2]$
$x = [4.0\ \ 2.4\ \ 1.7\ \ 1.8\ \ 1.5\ \ 2.8\ \ -3.0\ \ -0.2\ \ 5.6\ \ 1.6]$

Compressed-Column μορφή

Μια δομή δεδομένων compressed-column μορφής για έναν $m\ x\ n$ πίνακα πραγματικών αριθμών A, που μπορεί να περιέχει μέχρι και $n_{z,max}$ μη μηδενικές εισαγωγές, αποτελείται από τρεις μονοδιάστατους πίνακες (αν υποθέσουμε ότι ο δείκτης θέσης ξεκινά απ το 1):

- Πίνακας ακεραίων: $p[1], p[2], ..., p[n+1] \in \{1, 2, ..., n_{z,max}+1\}$
- Πίνακας ακεραίων: $i[1], i[2], ..., i[n_{z,max}] \in \{1, 2, ..., m\}$
- Πίνακας πραγματικών: $x[1], x[2], ..., x[n_{z,max}] \in \mathbb{R}$ όπου $p[1] = 1, p[n+1] = n_z + 1$, και όπου $n_z < n_{z,max}$ είναι ο πραγματικός αριθμός των μη μηδενικών, έτσι ώστε οι δείκτες σειράς των μη μηδενικών εισαγωγών στην στήλη j του A να αποθηκεύονται σε οποιαδήποτε σειρά, στον:

  $r[p[j]], r[p[j]+1], ..., r[p[j+1]-1]$

  και οι αντίστοιχες τιμές εισόδου του A αποθηκεύονται στον:

  $x[p[j]], x[p[j]+1], ..., x[p[j+1]-1]$

  και, αν η στήλη j έχει μόνο μηδενικές εισόδους, τότε $p[j] = p[j+1]$.

Το κάθε στοιχείο του διανύσματος *p* περιέχει την θέση στο διάνυσμα *i* στην οποία αρχίζει η αναφορά για την εκάστοτε στήλη, το διάνυσμα *i* περιέχει την θέση γραμμής των μη μηδενικών στοιχείων, σταδιακά για την κάθε στήλη και τέλος το διάνυσμα x αποτελείται από τις τιμές των μη μηδενικών στοιχείων.

Για παράδειγμα, ο πίνακας (2) έχει την παρακάτω compressed-column μορφή:

p = [1 4 7 9 11]
i = [1 2 4 2 3 4 1 3 2 4]
x = [5.6 2.4 2.8 1.5 1.6 -3.0 1.8 4.0 -0.2 1.7]

2.3 Μέθοδοι Επίλυσης Γραμμικού Συστήματος

Για την λύση του συστήματος (1) χρησιμοποιούνται οι παρακάτω δυο ομάδες μεθόδων:

1. Άμεσες μέθοδοι: λύνουν το σύστημα σε έναν σταθερό και προκαθορισμένο αριθμό βημάτων. Αυτή η ομάδα περιλαμβάνει την απαλοιφή του Gauss και τις πολλές παραλλαγές της, όπως η παραγοντοποίηση *LU* και *Cholesky*.

2. Επαναληπτικές μέθοδοι: είναι επαναληπτικές τεχνικές που προσεγγίζουν την λύση με σταδιακά βελτιωμένη ακρίβεια. Παραδείγματα αυτής της ομάδας είναι η *Gauss-Seidel*, η *Gauss-Jacobi*, η *Conjugate Gradient*, κ.λ.π.

Οι άμεσες μέθοδοι διάσπασης πίνακα όπως η *LU* και *Cholesky* αλγόριθμοι παραγοντοποίησης μπορούν να χρησιμοποιηθούν στην λύση του παραπάνω γραμμικού συστήματος με μεγάλη ακρίβεια αλλά ελλιπή αποδοτικότητα μνήμης και χρόνου εκτέλεσης, εξαιτίας στον μεγάλο αριθμό καινούριων fill-ins που δημιουργούνται κατά την διάρκεια της διαδικασίας παραγοντοποίησης, ειδικά για την ανάλυση δικτύων ηλεκτρικής ενέργειας μεγάλης κλίμακας.

Απ την άλλη μεριά, οι επαναληπτικές μέθοδοι [4][6] επιδεικνύουν πολύ καλύτερη αποδοτικότητα μνήμης, η οποία εμπλέκει μόνο τον πίνακα sparse και μερικά διανύσματα κατά την διάρκεια των υπολογισμών. Αξίζει λοιπόν να γίνει μια σύντομη επισκόπηση αυτών, μιας και σε επαναληπτική μέθοδο θα εφαρμόσουμε τους preconditioners στην συνέχεια της εργασίας.

2.4 Επαναληπτικές Μέθοδοι

Οι επαναληπτικές μέθοδοι επίλυσης συστημάτων χωρίζονται στις στάσιμες και τις μη στάσιμες.

Στάσιμες επαναληπτικές μέθοδοι

Οι επαναληπτικές μέθοδοι που μπορούν να εκφραστούν με την μορφή

$$x^{(k)} = \mathrm{B}\, x^{(k-1)} + \mathrm{c}$$

(όπου ούτε το B ούτε το c εξαρτώνται από τον επαναληπτικό μετρητή k) ονομάζονται στάσιμες επαναληπτικές μέθοδοι.

Οι κυριότερες από αυτές είναι οι:

- *Jacobi.*

  Η μέθοδος *Jacobi* βασίζεται στην λύση για κάθε μεταβλητή τοπικά με σεβασμό στις άλλες μεταβλητές. Μια επανάληψη της μεθόδου αντιστοιχεί στην επίλυση για κάθε μεταβλητή μία φορά. Η προκύπτουσα μέθοδος είναι εύκολη στην κατανόηση και την εφαρμογή, αλλά η σύγκλιση είναι αργή.

- *Gauss-Seidel.*

  Η μέθοδος *Gauss-Seidel* είναι σαν την μέθοδο *Jacobi,* μόνο που χρησιμοποιεί ενημερωμένες τιμές εφόσον είναι διαθέσιμες. Σε γενικές γραμμές, αν η μέθοδος *Jacobi* συγκλίνει, η μέθοδος *Gauss-Seidel* θα συγκλίνει γρηγορότερα από ότι η *Jacobi* μέθοδος, παρότι πάλι αργά.

- SOR

  Η *Successive Overrelaxation* (SOR) μπορεί να προκύψει από την μέθοδο *Gauss-Seidel* με την εισαγωγή μιας παραμέτρου προέκτασης ω. Για την βέλτιστη επιλογή του ω, η SOR μπορεί να συγκλίνει ταχύτερα από την *Gauss-Seidel* κατά μια τάξη μεγέθους.

- SSOR

  Η *Symmetric Successive Overrelaxation* (SSOR) δεν έχει κανένα πλεονέκτημα έναντι της SOR ως αυτόνομη επαναληπτική μέθοδος. Ωστόσο, είναι χρήσιμη ως ένας preconditioner για μη στάσιμες μεθόδους.

Μη στάσιμες επαναληπτικές μέθοδοι

Οι μη στάσιμες επαναληπτικές μέθοδοι διαφέρουν απ τις στάσιμες μεθόδους στο ότι οι υπολογισμοί εμπλέκουν πληροφορίες που αλλάζουν σε κάθε επανάληψη. Συνήθως, οι σταθερές υπολογίζονται λαμβάνοντας εσωτερικά παράγωγα υπολοίπων ή άλλων διανυσμάτων που προκύπτουν από την επαναληπτική μέθοδο.

Κάποιες από τις σημαντικότερες μεθόδους τις κατηγορίας είναι οι:

- *Conjugate Gradient* (CG).

  Η μέθοδος των *συζυγών κλίσεων* αντλεί το όνομά της από το γεγονός ότι παράγει μια σειρά από συζυγή (ή ορθογώνια) διανύσματα. Αυτά τα διανύσματα είναι τα κατάλοιπα των επαναλήψεων. Επίσης, είναι οι βαθμίδες ενός τετραγωνικού λειτουργικού, η ελαχιστοποίηση του οποίου είναι ισοδύναμη με την επίλυση του γραμμικού συστήματος. Η CG είναι μια εξαιρετικά αποτελεσματική μέθοδος όταν ο πίνακας συντελεστών είναι συμμετρικός θετικά ορισμένος, αφού απαιτείται η αποθήκευση για μόνο ένα περιορισμένο αριθμό διανυσμάτων.

- *Minimum Residual* (MINRES) and *Symmetric LQ* (SYMMLQ).

  Αυτές οι μέθοδοι είναι υπολογιστικά εναλλακτικές λύσεις για την CG για πίνακες συντελεστών που είναι συμμετρικοί αλλά ενδεχομένως αόριστοι. Η SYMMLQ θα δημιουργήσει τις ίδιες επαναλήψεις λύσης όπως η CG αν ο πίνακας συντελεστών είναι συμμετρικός θετικά ορισμένος.

- *Conjugate Gradient* on the *Normal Equations*: CGNE and CGNR.

  Αυτές οι μέθοδοι βασίζονται στην εφαρμογή της μεθόδου CG σε μία από τις δύο μορφές των *κανονικών εξισώσεων* για Ax = b. Η CGNE λύνει το σύστημα $(AA^T)y = b$ ως προς y και στη συνέχεια υπολογίζει τη λύση $x = A^T y$. Η CGNR λύνει $(A^T A)x = b$ για το διάνυσμα λύσης x όπου $b = A^T b$. Όταν ο πίνακας συντελεστών A είναι αντιστρέψιμος και μη συμμετρικός, πίνακες κανονικών εξισώσεων AA^T και $A^T A$ θα είναι συμμετρικοί και θετικά ορισμένοι, και ως εκ τούτου η CG μπορεί να εφαρμοστεί. Η σύγκλιση μπορεί να είναι αργή, δεδομένου ότι το φάσμα των πινάκων κανονικών εξισώσεων θα είναι λιγότερο ευνοϊκές από ό, τι το φάσμα του *A.*

- *Generalized Minimal Residual* (GMRES).

  Η μέθοδος *Generalized Minimal Residual* υπολογίζει μια σειρά από ορθογώνια διανύσματα (όπως η MINRES), και τα συνδυάζει μέσω μιας λύσης και ενημέρωσης ελαχίστων τετραγώνων. Ωστόσο, αντίθετα με την MINRES (και την CG) απαιτεί την αποθήκευση ολόκληρης της αλληλουχίας, απαιτώντας ένα μεγάλο μέγεθος αποθήκευσης. Για το λόγο αυτό, χρησιμοποιούνται καινούριες εκδόσεις αυτής της μεθόδου. Στις νέες εκδόσεις, οι υπολογισμοί και το κόστος αποθήκευσης περιορίζονται από τον καθορισμό ενός σταθερού αριθμού διανυσμάτων που πρόκειται να παραχθεί. Αυτή η μέθοδος είναι χρήσιμη για γενικούς μη συμμετρικούς πίνακες

- *BiConjugate Gradient* (BiCG).

  Η μέθοδος *Biconjugate Gradient* παράγει δύο παρόμοιες με την CG αλληλουχίες διανυσμάτων, μία βασισμένη σε ένα σύστημα με τον αρχικό πίνακα συντελεστών A, και μία για τον A^T. Αντί να ορθογωνιοποιήσει κάθε αλληλουχία, γίνονται αμοιβαία ορθογώνιες, ή "διορθογώνιες". Αυτή η μέθοδος, όπως η CG, χρησιμοποιεί περιορισμένη αποθήκευση. Είναι χρήσιμο, όταν πίνακας είναι μη συμμετρικός και αντιστρέψιμος και nonsingular. Ωστόσο, η σύγκλιση μπορεί να είναι ακανόνιστη, και υπάρχει η πιθανότητα να καταρρεύσει η μέθοδος. Η BiCG απαιτεί πολλαπλασιασμό με τον συντελεστή πίνακα και με ανάστροφο του σε κάθε επανάληψη.

Στην συνέχεια ακολουθεί μια αναλυτικότερη επεξήγηση της μεθόδου *συζυγών κλίσεων* (CG), μιας και είναι η μέθοδος στην οποία θα εφαρμόσουμε τους preconditioner τους οποίους μελετά αυτή η διπλωματική εργασία και η κατανόησή της θα βοηθήσει στις καλύτερη εκτίμηση των τελικών αποτελεσμάτων.

2.5 Μέθοδος *συζυγών κλίσεων* (CG)

Η μέθοδος *συζυγών κλίσεων* είναι μια αποτελεσματική μέθοδος για συμμετρικά θετικά ορισμένα συστήματα. Είναι η παλαιότερη και πιο γνωστή από τις μη στάσιμες μεθόδους που συζητήθηκαν προηγουμένως.

Η μέθοδος προχωρά δημιουργώντας αλληλουχίες διανυσμάτων επαναλήψεων (δηλαδή, διαδοχικές προσεγγίσεις της λύσης), υπόλοιπα που αντιστοιχούν στις αναπροσεγγίσεις, και τις κατευθύνσεις αναζήτησης που χρησιμοποιούνται για την επικαιροποίηση των αναπροσεγγίσεων και υπολοίπων. Αν και το μήκος αυτών των αλληλουχιών μπορεί να γίνει μεγάλο, μόνο ένας μικρός αριθμός διανυσμάτων χρειάζεται να διατηρηθεί στη μνήμη.

Σε κάθε επανάληψη της μεθόδου, εκτελούνται δύο εσωτερικά γινόμενα προκειμένου να υπολογιστούν τα ενημερωμένα αυξανόμενα επίπεδα που έχουν οριστεί για να κάνουν τις ακολουθίες να πληρούν ορισμένες συνθήκες ορθογωνιότητας. Σε ένα συμμετρικό θετικά ορισμένο γραμμικό σύστημα αυτές οι συνθήκες υποδηλώνουν ότι η απόσταση προς την πραγματική λύση ελαχιστοποιείται σε κάποια νόρμα.

Ο αλγόριθμος της CG σε ψευδοκώδικα είναι ο παρακάτω:

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$

for *i = 1, 2, . . .*

solve $Mz^{(i-1)} = r^{(i-1)}$

$\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$

if *i = 1*

$p^{(1)} = z^{(0)}$

else

$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$

$p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$

endif

$q^{(i)} = Ap^{(i)}$

$\alpha_i = \rho_{i-1} / p^{(i)^T} q^{(i)}$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence; continue if necessary

end

Στον παραπάνω ψευδοκώδικα χρησιμοποιείται ο preconditioner *M,* ο οποίος πρέπει να είναι συμμετρικός και θετικά ορισμένος. Για *M=I* προκύπτει η unpreconditioned εκδοχή του αλγόριθμου *Conjugate Gradient*. Μια συνήθης μορφή του preconditioner *M*, που χρησιμοποιείται και με την οποία θα συγκρίνουμε τους preconditioner της εργασίας, είναι αυτή στην οποία περιέχει μόνο τα στοιχεία της διαγωνίου του πίνακα *A* (*Jacob*i preconditioner).

Κεφάλαιο 3

Preconditioned Επαναληπτικοί Επιλυτές

3.1 Η Έννοια του Preconditioning

Το preconditioning είναι μια βασική τεχνική για την επιτάχυνση των επαναληπτικών επιλυτών, όπως της μεθόδου συζυγών κλίσεων (CG) και της Generalized Minimal Residual (GMRES). Η βασική ιδέα είναι ότι αντί να λύσουμε απευθείας το δοθέν γραμμικό σύστημα (1), γίνεται κάποια προσπάθεια για να μειωθεί ο αριθμός κατάστασης του συστήματος.

$$k(A) = \|A\|_2 \|A^{-1}\|_2 = \lambda_{max}(A) \,/\, \lambda_{min}(A)$$

Στον preconditioned επαναληπτικό επιλυτή, προσπαθούμε να βρούμε έναν πίνακα P που ονομάζεται preconditioner, έτσι ώστε να πρέπει να λύσουμε το παρακάτω εναλλακτικό γραμμικό σύστημα :

$$P^{-1}A \cdot x = P^{-1}b$$

Αν ο αριθμός κατάστασης του νέου συστήματος ,

$$k(A,P) = \frac{\lambda_{max}(A,P)}{\lambda \min(A,P)}$$

είναι πολύ μικρότερος απ ότι του αρχικού, τότε η λύση του γραμμικού συστήματος μπορεί να βρεθεί με πολύ πιο γρήγορο τρόπο (μικρότερος αριθμός υπολογισμών). Η επιλογή του preconditioner έχει δραματική επίδραση στην σύγκλιση.

3.2 Ο *Jacobi* Preconditioner

Ένας δημοφιλής preconditioner στην βιβλιογραφία και απλός στην δημιουργία είναι ο *Jacobi* preconditioner, με τον οποίο θα συγκρίνουμε τους νέους preconditioner της παρούσας εργασίας, ο οποίος διαμορφώνεται μέσα από τη λήψη των διαγώνιων στοιχείων του πίνακα *A.*

$$m_{i,j} = \begin{cases} \alpha_{i,j} & if\ i=j \\ 0 & \text{otherwise} \end{cases}$$

Για το πρότυπο πρόβλημα , $k(B^{-1}A) = O(n) = k(A)$, έτσι ώστε ο ασυμπτωτικός ρυθμός σύγκλισης να μην βελτιώνεται με διαγώνια κλιμάκωση. Ο B σε αυτή την περίπτωση δεν χρειάζεται να συνυπολογιστεί. Η μνήμη που απαιτείται για τον preconditioner είναι $O(n)$ μιας και είναι αραιός πίνακας. Επίσης, ο συγκεκριμένος preconditioner είναι πολύ εύκολος στην λύση, αφού απαιτείται μόνο η διαίρεση του κάθε διανύσματος με την αντίστοιχη διαγώνια είσοδο του *B*.

Ο *Jacobi* preconditioner χρησιμοποιείται ευρέως στην πράξη, λόγω της απλότητας του. Ωστόσο, ένας τέτοιος preconditioner δεν μπορεί να είναι πολύ χρήσιμος για προσομοιώσεις δικτύου ηλεκτρικής ενέργειας, δεδομένου ότι *ill-conditioned* γραμμικά συστήματα συνήθως εμφανίζονται σε προβλήματα ανάλυσης δικτύου ηλεκτρικής ενέργειας που σχετίζονται με σχετικά μεγάλους αριθμούς κατάστασης.

3.3 Ο SSOR Preconditioner

Ένα άλλο παράδειγμα preconditioner είναι ο SSOR preconditioner, ο οποίος, σαν τον *Jacobi*, μπορεί εύκολα να παραχθεί από τον πίνακα A χωρίς καθόλου προεργασία.

Αν υποθέσουμε ότι έχουμε έναν συμμετρικό πίνακα A που μπορεί να αποσυνδεθεί ως

$$A = D + L + L^T$$

στην διαγώνιό του, κάτω και άνω τριγωνικό μέρος, ο SSOR πίνακας ορίζεται ως

$$M = (D + L)D^{-1}(D + L)^T$$

Ο πίνακας SSOR δίνεται σε *factored* μορφή, και έτσι ο preconditioner μοιράζεται πολλές ιδιότητες από άλλες βασισμένες σε παραγοντοποίηση μεθόδους. Για παράδειγμα, η καταλληλότητά του για επεξεργαστές διανύσματος ή παράλληλες αρχιτεκτονικές εξαρτάται άμεσα από την στοίχιση των μεταβλητών του.

3.4 Τα Προβλήματα των Υπαρχόντων Preconditioner

Επαναληπτικές μέθοδοι όπως οι μέθοδοι *συζυγών κλίσεων* (CG) απαιτούν υψηλής ποιότητας preconditioners για να βελτιώσουν το ποσοστό σύγκλισης[1] . Ένας καλός preconditioner P θα πρέπει να είναι αποδοτικό στον υπολογισμό (το γραμμικό σύστημα $Px = y$ πρέπει να είναι εύκολο να λυθεί), και αποτελεσματικό στην σημαντική μείωση του αριθμού κατάστασης (το $k(P^{-1}A)$ είναι μικρό).

Ο απλούστερος διαγώνιος preconditioner (*Jacobi* preconditioner) είναι αποδοτικός στον υπολογισμό αλλά μπορεί να μην είναι αποτελεσματικός στην μείωση του αριθμού των επαναλήψεων ή του αριθμού κατάστασης, ενώ οι ατελής preconditioners με πίνακας *Cholesky* βασισμένοι στην παραγοντοποίηση μπορεί να επιτύχουν πολύ καλύτερη σύγκλιση, αλλά προϋποθέτουν πολύ υψηλότερη κατανάλωση μνήμης και υπολογιστικό κόστος κατά τη διάρκεια της διαδικασίας preconditioning.

Ένας στοχαστικός preconditioner έχει παρουσιαστεί στο [12] για να επιτευχθεί καλός ρυθμός σύγκλισης, αλλά η διαδικασία ενημέρωσης μπορεί ακόμα να είναι πολύ δαπανηρή.

Δεν πρέπει όμως να αγνοηθεί και το γεγονός ότι στους preconditioner υπάρχει πάντα ένα trade-off μεταξύ του κόστους κατασκευής και εφαρμογής του preconditioner και του κέρδους στην ταχύτητα σύγκλισης δεδομένου ότι η χρήση ενός preconditioner σε μια επαναληπτική μέθοδο συνεπάγεται κάποιο επιπλέον κόστος, τόσο αρχικά για την δημιουργία όσο και ανά επανάληψη για την εφαρμογή του.

1. Η CG χρειάζεται $O\left(k^{1/2}(A)\right)$ επαναλήψεις για να λύσει το $Ax = b$

Ορισμένοι preconditioners χρειάζονται μικρή ή καθόλου φάση κατασκευής (για παράδειγμα ο *Jacobi* preconditioner), αλλά για άλλους, όπως για τις ελλιπείς παραγοντοποιήσεις, μπορεί να απαιτείται σημαντική εργασία. Αν και η εργασία σε *scalar* όρους μπορεί να συγκριθεί με μία μόνο επανάληψη, η κατασκευή του preconditioner μπορεί να μην επιδέχεται παραλληλισμού ακόμα και αν η εφαρμογή του preconditioner είναι. Στην περίπτωση αυτή, το αρχικό κόστος πρέπει να αποσβένεται κατά την διάρκεια των επαναλήψεων, ή σε επαναλαμβανόμενη χρήση του ίδιου preconditioner σε πολλαπλά γραμμικά συστήματα.

Κεφάλαιο 4

Έννοιες θεωρίας γράφων στην ανάλυση κυκλωμάτων

4.1 Η Έννοια της Θεωρία των Γράφων

Η θεωρία των γράφων για υποστήριξη preconditioner, γνωστή ως *Support Theory*, είναι μια αρκετά πρόσφατη μεθοδολογία οριοθέτησης αριθμών κατάστασης preconditioned συστημάτων. Πιο συγκεκριμένα, πρόκειται για ένα σύνολο εργαλείων και τεχνικών για την οριοθέτηση *extremal* ιδιοτιμών. Για μερικές επαναληπτικές μεθόδους (*συζυγές κλίσεις* συγκεκριμένα), ο λόγος των μεγαλύτερων προς τις μικρότερες ιδιοτιμές παρέχει ένα άνω όριο για τον αριθμό των επαναλήψεων.

4.2 Όροι της Θεωρίας των Γράφων

Για την καλύτερη κατανόηση της επεξήγησης της δημιουργίας των preconditioner της παρούσας εργασίας που θα ακολουθήσει, γίνεται μια σύντομη αναφορά στις βασικές ορολογίες της θεωρία των γράφων.

Ένας *μη κατευθυνόμενος γράφος* $G=(V,E)$ είναι μια συλλογή από V κόμβους ή κορυφές μαζί με ένα σύνολο από E ακμές, όπου κάθε ακμή στο E είναι ένα ζεύγος κόμβων.

Walk είναι μια εναλλασσόμενη σειρά από κορυφές και ακμές που αρχίζει και τελειώνει με κορυφή, τέτοια ώστε κάθε ακμή στην αλληλουχία συνδέει την κορυφή που προηγείται με την κορυφή μετά από αυτήν.

Path είναι ένα μονοπάτι του γράφου, δηλαδή ένα *walk* στο οποίο όλες οι κορυφές είναι απομακρυσμένες.

Length ενός *path* είναι το μήκος ενός μονοπατιού του γράφου που ορίζεται ως το άθροισμα των βαρών των ακμών του *path*.

Shortest path δυο κορυφών είναι το *path* που συνδέει αυτές τις δύο κορυφές και έχει το μικρότερο *length*.

Συνδεδεμένος γράφος είναι ένα γράφος εάν υπάρχει ένα path μεταξύ κάθε ζεύγους κορυφών.

Πλήρης γράφος είναι ένας γράφος στον οποίο όλες οι κορυφές είναι ανά δύο γειτονικές.

Distance ανάμεσα σε δυο κορυφές είναι η απόστασή τους, δηλαδή το *length* του *shortest path* που τις ενώνει.

Cycle ενός γράφου είναι ένα *walk* στο οποίο η πρώτη και η τελευταία κορυφή είναι η ίδια.

Tree είναι ένας συνδεδεμένος γράφος που δεν περιέχει κύκλο.

Κατευθυνόμενος γράφος είναι ένας γράφος στον οποίο οι ακμές είναι ταξινομημένα ζεύγη, δηλαδή υποδηλώνουν κατεύθυνση προσδιορίζοντας τις συνδεόμενες κορυφές τους ως κεφαλή ή ουρά.

Weighted ή σταθμισμένος γράφος (κατευθυνόμενος ή μη) είναι ένας γράφος που περιέχει μία συνάρτηση $w: E \to \mathbb{R}$ η οποία αναθέτει βάρη στις ακμές.

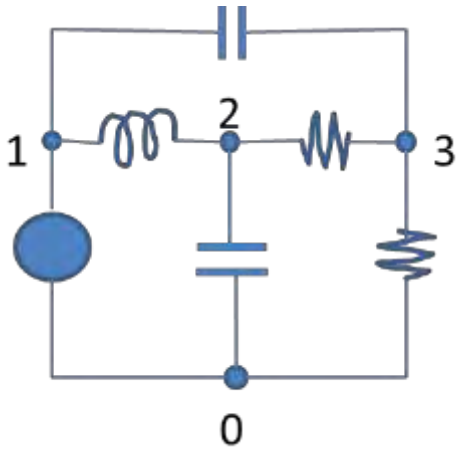
Έστω ότι $G = (V(G), E(G))$ *και* $H = (V(H), E(H))$ είναι γράφοι. Ο H είναι *υπογράφος* του G αν $V(H) \subseteq V(G)$ *και* $E(H) \subseteq E(G)$. Τότε ο G είναι *υπεργράφος* του H.

4.3 Τα Ηλεκτρικά Κυκλώματα από την Οπτική των Γράφων

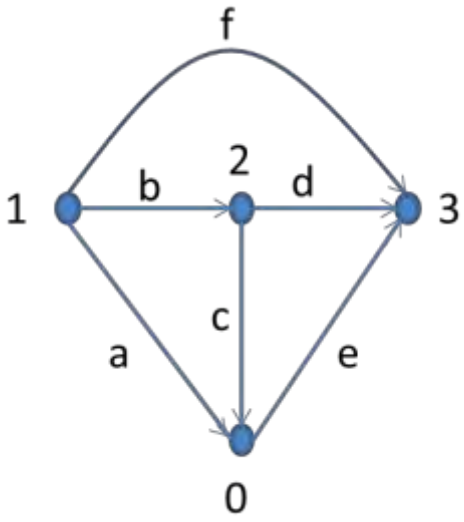
Η χρήση γράφων για την αναπαράσταση ηλεκτρικών κυκλωμάτων είναι ένας πολύ πρακτικός τρόπος αναπαράστασης, καθώς επιτρέπει τον εύκολο χειρισμό αυτών.

Η απευθείας αναπαράσταση ενός δικτύου ηλεκτρικής ενέργειας σε γράφο είναι αρκετά απλή διαδικασία. Ουσιαστικά αναπαριστούμε τους κόμβους του δικτύου σε κορυφές στον γράφο και τις συνδέσεις των κόμβων σε ακμές.

Για παράδειγμα, αν έχουμε το κύκλωμα της εικόνας 4.1, μπορούμε να το αναπαραστήσουμε με τον γράφο της εικόνας 4.2 όπου η κατευθυνόμενες ακμές υποδηλώνουν την κατεύθυνση του ρεύματος σύμφωνα με το [2] . Μπορεί να είναι και μη κατευθυνόμενο αν δεν μας χρειάζεται αυτή η πληροφορία.



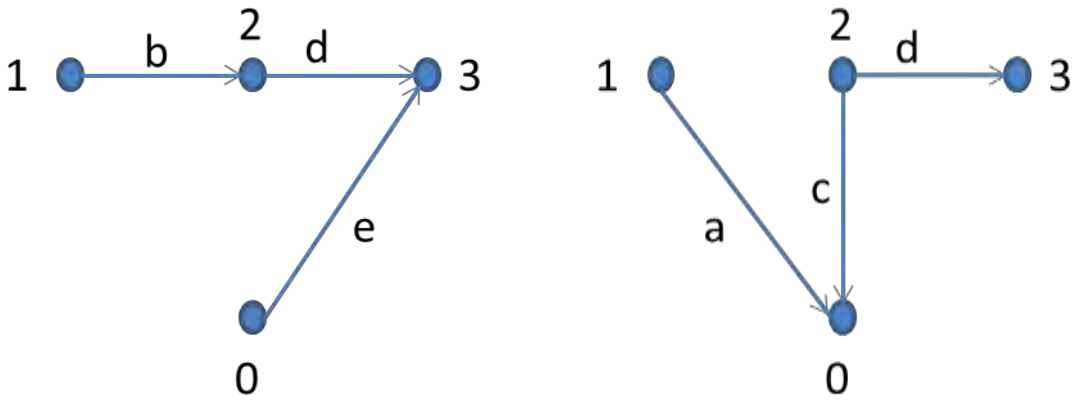
Εικόνα 4.1: Κύκλωμα ηλεκτρικής ενέργειας



Εικόνα 4.2: Γράφος αναπαράστασης κυκλώματος ηλεκτρικής ενέργειας

Για τον πιο αποδοτικό χειρισμό του κυκλώματος μέσω γράφου είναι σύνηθες να γίνεται ελάττωση κάποιων ακμών έτσι ώστε ο γράφος στον οποίο θα δουλέψουμε να είναι ο ελάχιστος συνδεδεμένος γράφος, δηλαδή να είναι δέντρο. Ουσιαστικά θέλουμε κάθε κόμβος να έχει μοναδικό μονοπάτι προς οποιονδήποτε άλλο κόμβο.

Κάποια δέντρα που προκύπτουν από τον παραπάνω γράφο απεικονίζονται στην εικόνα 4.3



Εικόνα 4.3: Δέντρα εξαγόμενα από γράφο

Όπως είναι φανερό υπάρχουν πολλές εναλλακτικές στην επιλογή του δέντρου που προκύπτει απ τον γράφο. Πρέπει λοιπόν να καθοριστεί μια σειρά προτεραιοτήτων ώστε να επιλεγεί ο καταλληλότερος γράφος. Αυτές οι προτεραιότητες όμως εξαρτιούνται από τον σκοπό για τον οποίο δημιουργήσαμε τον γράφο και από τον τρόπο με τον οποίο σκοπεύουμε να τον χρησιμοποιήσουμε.

Στους preconditioners που θα μελετήσουμε παρακάτω λοιπόν, θα καθοριστούν οι προτεραιότητες με τις οποίες αυτοί επιλέγουν το δέντρο που θα εξάγουν από τον γράφο του δικτύου ηλεκτρικής ενέργειας.

Η διαφορά της παραπάνω διαδικασίας στους preconditioners που θα μελετήσουμε είναι ότι τον γράφο τον εξάγουμε από τον πίνακα Α και όχι απευθείας από το κύκλωμα, χρησιμοποιώντας βέβαια την ίδια λογική.

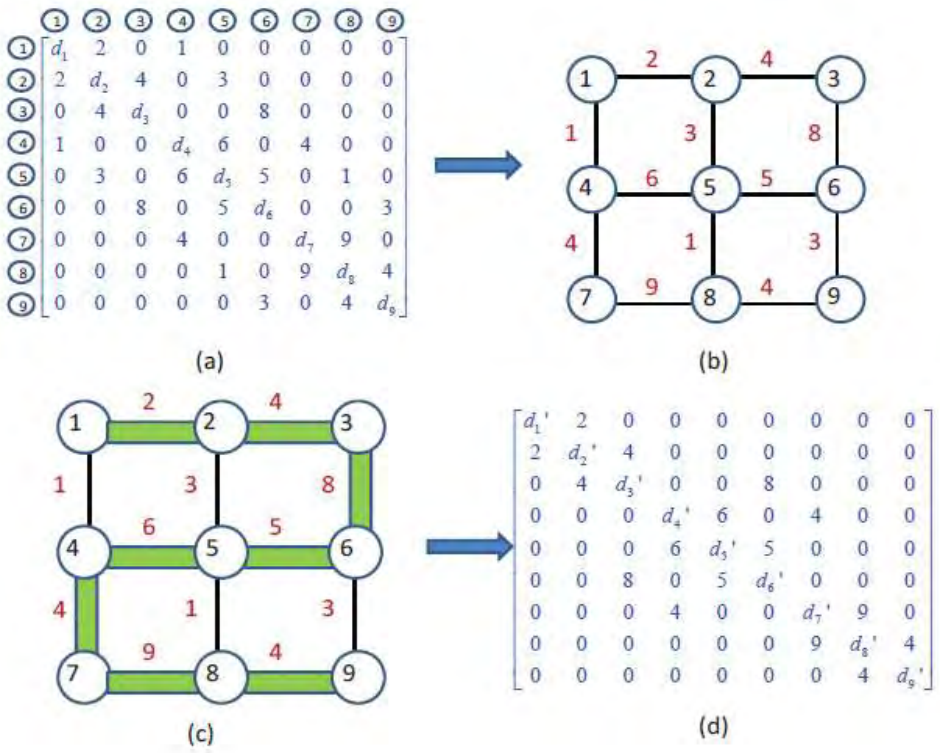
Κεφάλαιο 5

Maximum Spanning Tree

5.1 Ο *Maximum Spanning Tree* Preconditioner

Το *support-graph preconditioning* είναι το να κατασκευαστεί πρώτα ένα γράφημα σύμφωνα με έναν δεδομένο πίνακα *A*, και στη συνέχεια να υπολογιστεί ο preconditioner *P* ως έναν ελαττωμένος πίνακας με βάση την υποδομή του γράφου. Η υποδομή είναι συνήθως ένα ειδικό υπογράφημα που εξάγεται από το πλήρες γράφημα. Θα μελετήσουμε σε αυτή την ενότητα το *maximum weighted spanning tree* σύμφωνα με το [3] ως παράδειγμα υποδομής , όπως φαίνεται στην εικόνα 5.1.

- Έστω για ένα γραμμικό σύστημα, ο πίνακας *A* απεικονίζεται στην εικ. 5.1 (a).
- Μπορούμε να μετατρέψουμε όλα τα εκτός διαγωνίου στοιχεία σε έναν γράφο όπως φαίνεται στην εικ. 5.1 (b). Στη συνέχεια, κάθε εκτός διαγωνίου στοιχείο (αγωγιμότητα σύρματος) στον πίνακα, αντιστοιχεί σε μια weighted ακμή του γράφου.
- Έπειτα, δημιουργείται ένα *maximum spanning tree* όπως φαίνεται στην εικ. 5.1 (c).
- Τέλος, μπορούμε να κατασκευάσουμε τον preconditioner πίνακα *P* σύμφωνα με το spanning tree που εξήγαμε, όπως φαίνεται στην εικ. 5.1 (d).



Εικόνα 5.1: *Maximum spanning tree* preconditioner

Έχει δειχθεί ότι με την εφαρμογή του πίνακα P ως preconditioner, το γραμμικό σύστημα $P^{-1}A \cdot x = P^{-1}b$ θα έχει φραγμένο αριθμό κατάστασης . Έχει αποδειχθεί ότι μια τέτοια τεχνική preconditioning μπορεί να εφαρμοστεί σε οποιουσδήποτε SPD πίνακες με κυρίαρχη διαγώνιο (SDD) .

Επισήμως, το *support-graph preconditioning* είναι το να υπολογίσουμε έναν preconditioner P έτσι ώστε οι γενικευμένες ιδιοτιμές και ο αριθμός κατάστασης του *pencil* πίνακα (A, P) να είναι φραγμένες . Αν και οι δύο A και P είναι SPD πίνακες, η σύγκλιση εξαρτάται από τον αριθμό κατάστασης $k(A, P)$ και υπολογίζεται ως εξής :

$$k(A,P) = \frac{\lambda_{max}(A,P)}{\lambda_{min}(A,P)}$$

όπου το $\lambda(A, P)$ δηλώνει τη γενικευμένη ιδιοτιμή .

5.2 Υλοποίηση του Maximum Spanning Tree Preconditioner

Πέρασμα τιμών σε γράφο

Το πρώτο βήμα της δημιουργίας του *support-graph* preconditioner είναι να περάσουμε τα δεδομένα του πίνακα *A* στον γράφο. Να σημειώσουμε εδώ ότι ο πίνακας πρέπει να είναι SPD. Ο πίνακας *A* περιέχει τις αγωγιμότητες των αντιστάσεων του κυκλώματος. Έτσι διατρέχουμε τον πίνακα του συστήματος MNA και εισάγουμε, με την κατάλληλη συνάρτηση, τις αρνητικές τιμές των μη μηδενικών και εκτός διαγωνίου στοιχείων στον γράφο. Θεωρούμε ότι αν *A[i][j]=value*, τότε ο κόμβος *i* συνδέεται με τον κόμβο *j* με ακμή βάρους -value.

Τον γράφο τον αναπαριστούμε με έναν πίνακα τόσων θέσεων όσος ο αριθμός των κόμβων *n*, ο οποίος αναπαριστάται από δείκτες σε στοιχεία struct. Με αυτόν τον τρόπο κάθε θέση του πίνακα αναπαριστά έναν κόμβο του γράφου και τα πεδία struct που δείχνει αυτή η θέση αναπαριστούν τους γείτονες αυτού του κόμβου με τις πληροφορίες που αυτοί περιέχουν (βάρος ακμής που συνδέει τον τρέχον κόμβο με τον κάθε γείτονα). Αυτή η δομή μας βολεύει γιατί δεν ξέρουμε εκ των προτέρων πόσους γείτονες έχει ο κάθε κόμβος του γράφου αλλά τους προσθέτουμε σταδιακά.

Αλγόριθμος του *Kruskal* για *maximum weight spanning tree*

Αφού έχει ολοκληρωθεί ο γράφος, σειρά έχει η εξαγωγή του *maximum weight spanning tree* από τον γράφο. Με βάση τον αλγόριθμο του *Kruskal*, θα ακολουθήσουμε τα παρακάτω βήματα για την εύρεση του :

1. Ταξινομώ τις ακμές σε φθίνουσα σειρά. Έστω *T* είναι το σύνολο των ακμών που αποτελούν το *maximun spanning tree*. *T=0* αρχικά.
2. Προσθέτω την $1^{η}$ ακμή στο *T*.
3. Προσθέτω την επόμενη ακμή στο *T* ανν δεν σχηματίζει κύκλο στο *T*. Αν δεν έχει μείνει άλλη ακμή, ανακηρύσσω το γράφημα σαν μη συνδεδεμένο.
4. Αν το *T* έχει *n-1* ακμές (όπου *n* ο αριθμός των κόμβων) σταμάτα και δώσε σαν έξοδο το *T*. Αλλιώς πήγαινε στο βήμα 3.

Οπότε δημιουργούμε ένα while loop, με συνθήκες τερματισμού το αν έχουμε φτάσει στο τέλος του πίνακα ταξινομημένων ακμών και το αν το *maximum spanning tree* έχει *n-1* ακμές, στο οποίο μέσα εισάγουμε μια ακμή σε κάθε επανάληψη του loop με την σειρά του πίνακα φθίνοντα ταξινομημένων ακμών, μόνο εάν δεν σχηματίζει κύκλο.

Έλεγχος ύπαρξης κύκλου μέσω *depth-first search*

Τον έλεγχο ύπαρξης κύκλου τον δημιουργούμε με βάση την μέθοδο *depth-first search* (DFS) σε ξεχωριστή αναδρομική συνάρτηση. Θα κάνουμε χρήση δυο στοιχείων σ΄ αυτήν την συνάρτηση, την κατάσταση του κόμβου (0=ανεξερεύνητη, 1=υπό-εξέταση, 2=εξερευνημένη) και τον πατέρα του κόμβου.

- Αρχίζω απ΄ τον $1^{ο}$ κόμβο, τον χρίζω υπό-εξέταση και προχωράω κατά βάθος (για κάθε ανεξερεύνητο κόμβο που προσπελάζω, ορίζομαι ως ο πατέρας του και καλώ την συνάρτηση τρεξίματος κατά βάθος με τον αυτόν τον κόμβο τώρα) μέχρι να μην υπάρχει άλλος συνδεδεμένος κόμβος να εξετάσω (ορίζομαι ως εξερευνημένος κόμβος) ή να "ακουμπήσω" σε ήδη εξερευνημένο κόμβο (τερματισμός συνάρτησης λόγω ύπαρξης κύκλου).

- Αν δεν υπάρχει άλλος κόμβος να τσεκάρω από το μονοπάτι με αρχή τον $1^{ο}$ κόμβο, αρχίζω καινούριο μονοπάτι από τον αμέσως επόμενο ανεξερεύνητο κόμβο και ακολουθώ την ίδια διαδικασία μέχρι να μην υπάρχει κανένας ανεξερεύνητος κόμβος ή να "ακουμπήσω" σε ήδη εξερευνημένο κόμβο .

- Αν εξερευνήσω όλους τους κόμβους του γράφου χωρίς να "πέσω πάνω" σε κόμβο ήδη εξερευνημένο , τότε δεν υπάρχει κύκλος.

Έπειτα από την εξαγωγή του *maximum spanning tree* το τελευταίο βήμα είναι να δημιουργήσουμε τον πίνακα preconditioner *P.* Ο πίνακας αυτός θα έχει ίδιο μέγεθος με τον *A* και ίδια στοιχεία με την διαφορά ότι τα στοιχεία (αντίστοιχες ακμές) που δεν ανήκουν στο *maximum spanning tree* του γράφου παίρνουν την τιμή 0 .

Οι βασικές συναρτήσεις που χρησιμοποιούνται στον κώδικα δημιουργίας του *maximum spanning tree* preconditioner είναι οι παρακάτω.

- *CalculateSpanning*

  Η συνάρτηση αυτή ταξινομεί τις ακμές κατά φθίνουσα σειρά και έπειτα αποφασίζει με βάση τις συνθήκες που προαναφέραμε αν θα καλέσει την συνάρτηση πρόσθεσης ακμής στο *maximum spanning tree* ή όχι.

- *MaximumSpanning*

  Η συνάρτηση αυτή προσθέτει σταδιακά ακμές με βάση τον ταξινομημένο πίνακα ακμών ελέγχοντας όμως να μην σχηματίζεται κύκλος.

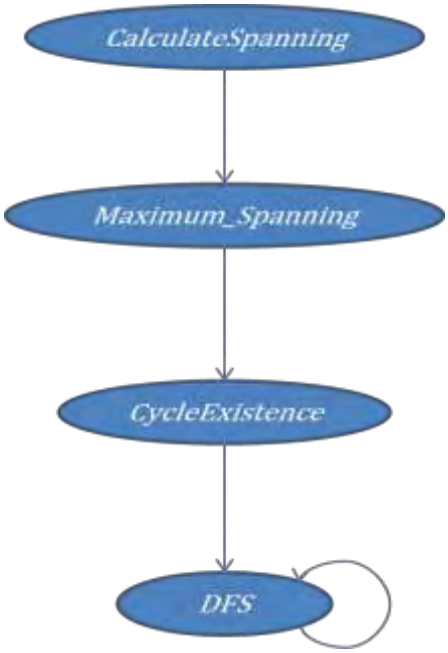
- *CycleExistence*

  Η συνάρτηση αυτή εφαρμόζει την μέθοδο αναζήτησης κατά βάθος, καλώντας την κατάλληλη συνάρτηση, για την εύρεση ύπαρξης κύκλου επαναληπτικά για κάθε κόμβο.

- *DFS*

  Η αναδρομική συνάρτηση αυτή υλοποιεί την αναζήτησης κατά βάθος (depth-first search) για εύρεση ύπαρξης κύκλου , επιστρέφοντας 1 αν υπάρχει κύκλος και 0 διαφορετικά.

Στην εικόνα 5.2 φαίνεται το διάγραμμα κλήσεων συναρτήσεων του κώδικα.



Εικόνα 5.2: Διάγραμμα κλήσεων συναρτήσεων

5.3 Εφαρμογή του *Maximum Spanning Tree* Preconditioner

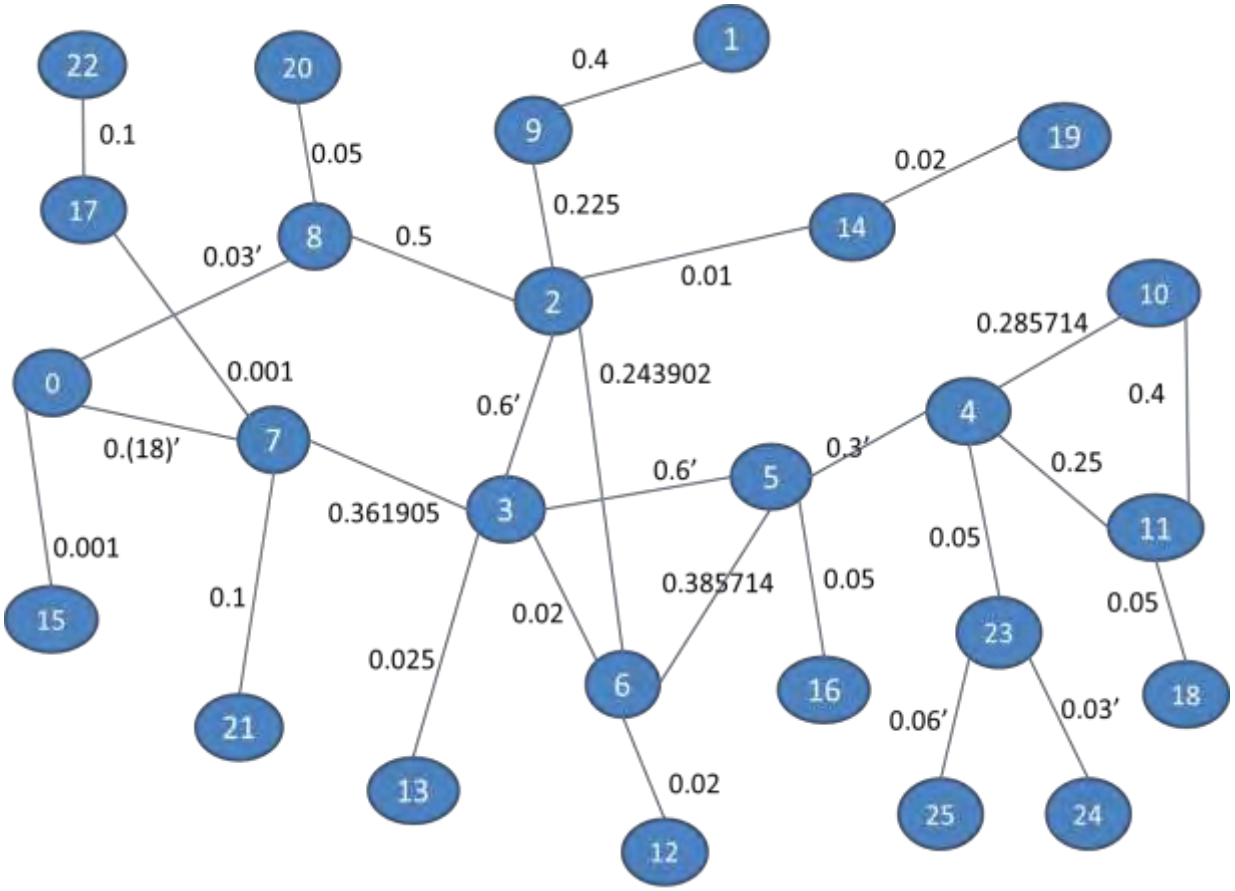
Σε αυτό το σημείο θα εφαρμόσουμε τον αλγόριθμο του *Maximum spanning tree* preconditioner σε ένα συγκεκριμένο παράδειγμα netlist, έτσι ώστε να γίνει ευκολότερη η κατανόησή της λειτουργίας του.

Το netlist στο οποίο θα εφαρμόσουμε τον αλγόριθμο είναι το:

```
I1 n1_4 n1_8 2e-3
I2 0 n1_6 1e-3
I3 n1_2 n1_10 3e-3
R1 n1_1 n1_5 3.5
R2 n1_1 n1_2 1.5
R3 n1_5 n1_2 50
R4 n1_5 n1_6 4.1
R5 n1_2 n1_6 1.5
R6 n1_3 n1_4 5.5
R7 n1_8 0 1e3
R8 n1_4 0 10
R9 n1_5 0 2
R10 n1_3 n1_2 3
R11 n1_7 n1_6  2
R12 n1_9 n1_8 2.5
R13 n1_1 n1_10 3
R14 n1_10 n1_11 3.5
R15 n1_11 n1_12 2.5
R16 n1_10 n1_12 4
R17 n1_9 n1_6 5
R18 n1_5 n1_1 10
R19 n1_4 n1_7 30
R20 n1_6 n1_9 40
R21 n1_2 n1_3 35
R22 0 n1_8 1e2
R23 n1_13 n1_5 50
R24 n1_14 n1_2 40
R25 n1_15 n1_6 1e2
R26 n1_16 n1_4 1e3
R27 n1_17 n1_1 20
R28 n1_3 n1_18 1e3
R29 n1_12 n1_19 20
R30 n1_15 n1_20 50
R31 n1_21 n1_7 20
R31 n1_3 n1_22 10
R33 n1_23 n1_18 10
R34 n1_25 n1_24 30
R35 n1_25 n1_26 30
R36 n1_25 n1_10 20
.OPTIONS SPANNING
```

Έπειτα από την δημιουργία του πίνακα αγωγιμοτήτων και το πέρασμα των αρνητικών τιμών των στοιχείων του σε γράφο όπως εξηγήθηκε προηγουμένως, προκύπτει ο παρακάτω γράφος απεικόνισης του κυκλώματος της εικόνας 5.3.

X’=περιοδικός αριθμός

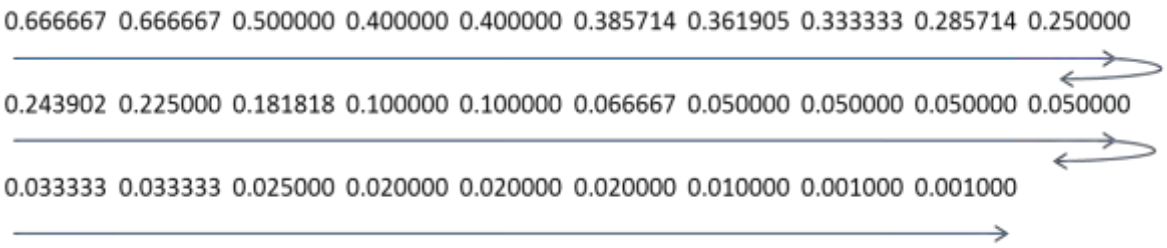


Εικόνα 5.3: Γράφος απεικόνισης του ηλεκτρικού κυκλώματος

Αυτή η απεικόνιση των κόμβων προέκυψε μετά από μετατροπή των ονομάτων των κόμβων του netlist μέσω hash table.

Μετά την δημιουργία του γράφου μπορούμε πλέον να προχωρήσουμε στο $1^{ο}$ βήμα του αλγόριθμου, δηλαδή την ταξινόμηση των ακμών κατά φθίνουσα σειρά.

Η σειρά με την οποία ταξινομήθηκαν οι ακμές του παραπάνω γράφου φαίνεται στην εικόνα 5.4.



Εικόνα 5.4: Ταξινομημένες ακμές γράφου

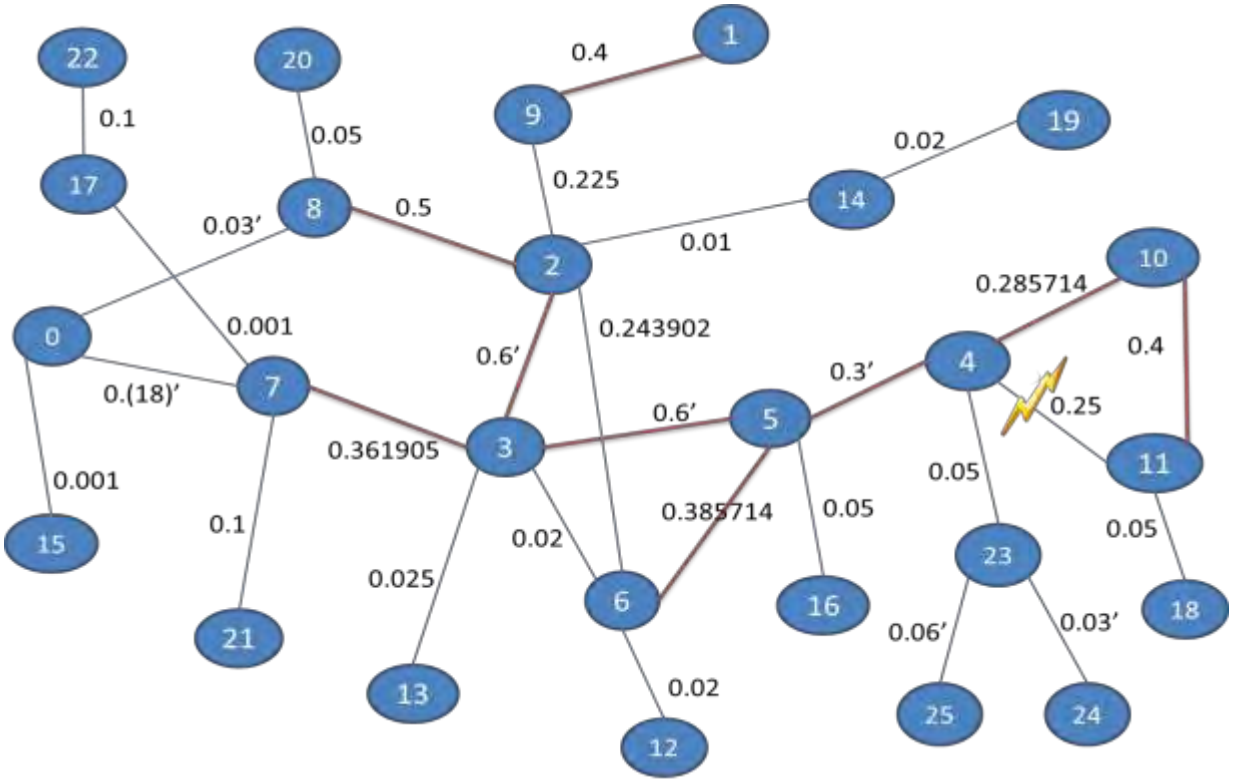
Στο 2^{ο} βήμα του αλγόριθμου προσθέτουμε την 1^{η} ακμή του πίνακα ταξινόμησης στο σύνολο των ακμών που αποτελούν το *maximum spanning tree*, έστω *T*. Τις ακμές που ανήκουν στο *T* θα τις αναπαριστούμε με κόκκινη ακμή στον γράφο. Εδώ λοιπόν προσθέτουμε την ακμή 0.6' που ενώνει τους κόμβους 2 και 3, έστω ακμή 2-3.

Στο 3^{ο} βήμα προσθέτουμε την επόμενη ακμή επαναληπτικά, μέχρις ότου είτε δημιουργηθεί κύκλος είτε δεν έχει μείνει άλλη ακμή στον πίνακα ταξινόμησης είτε το *T* έχει *n-1* ακμές, όπου n ο αριθμός των κόμβων του γράφου (ένα δέντρο δεν μπορεί να έχει πάνω από *n-1* ακμές).

Σε αυτήν την φάση λοιπόν προσθέτουμε συνεχόμενα χωρίς κάποιο πρόβλημα τις ακμές 0.6' (3-5), 0.5 (2-8), 0.4 (1-9), 0.4 (10-11), 0.385714 (5-6), 0.361905 (3-7), 0.3' (4-5) μέχρι και την 0.285714 (4-10).

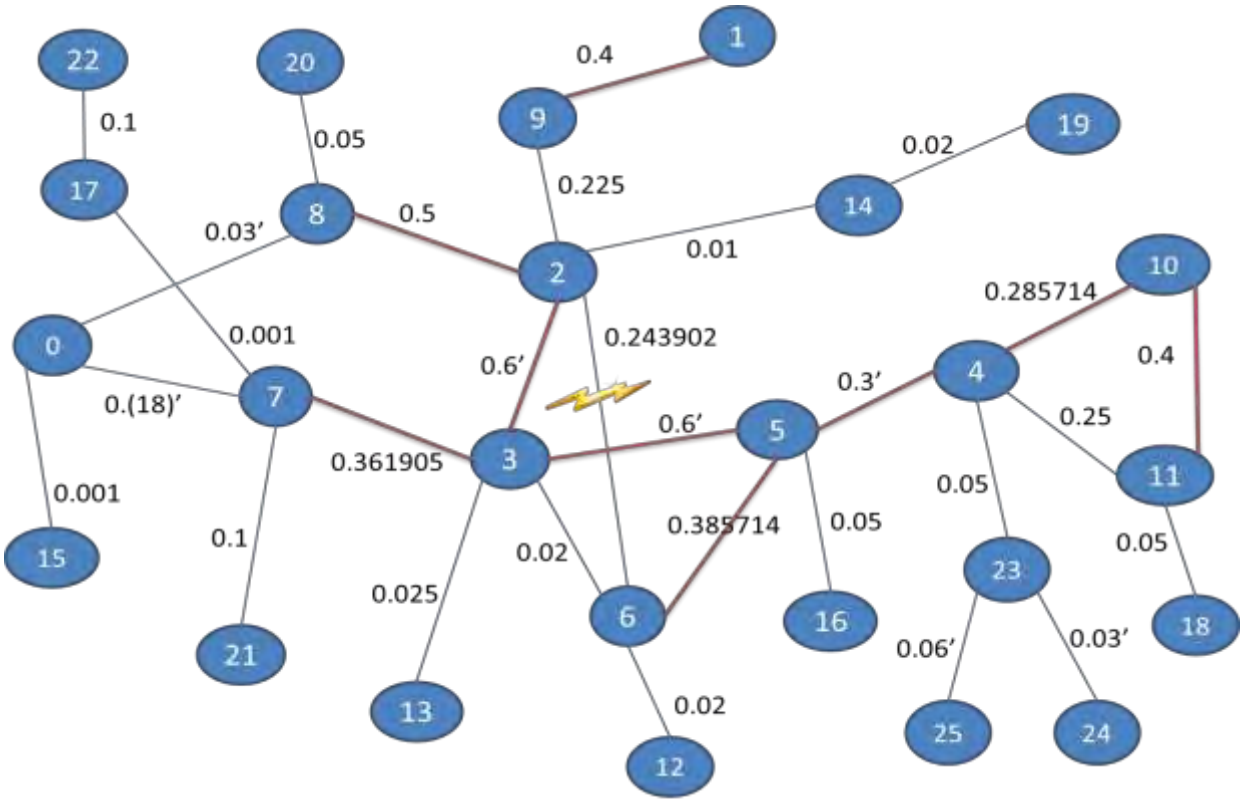
Όταν φτάνουμε στην ακμή 0.25 των κόμβων 4-11, όπως βλέπουμε και στην εικόνα 5.5, σχηματίζεται ο κύκλος 4-10-11-4, οπότε η ακμή αυτή δεν μπορεί να ενταχθεί στο *T*.

Έτσι συνεχίζουμε με τον έλεγχο της επόμενης ακμής στον πίνακα ταξινόμησης.



Εικόνα 5.5: Σχηματισμός κύκλου 4-10-11-4

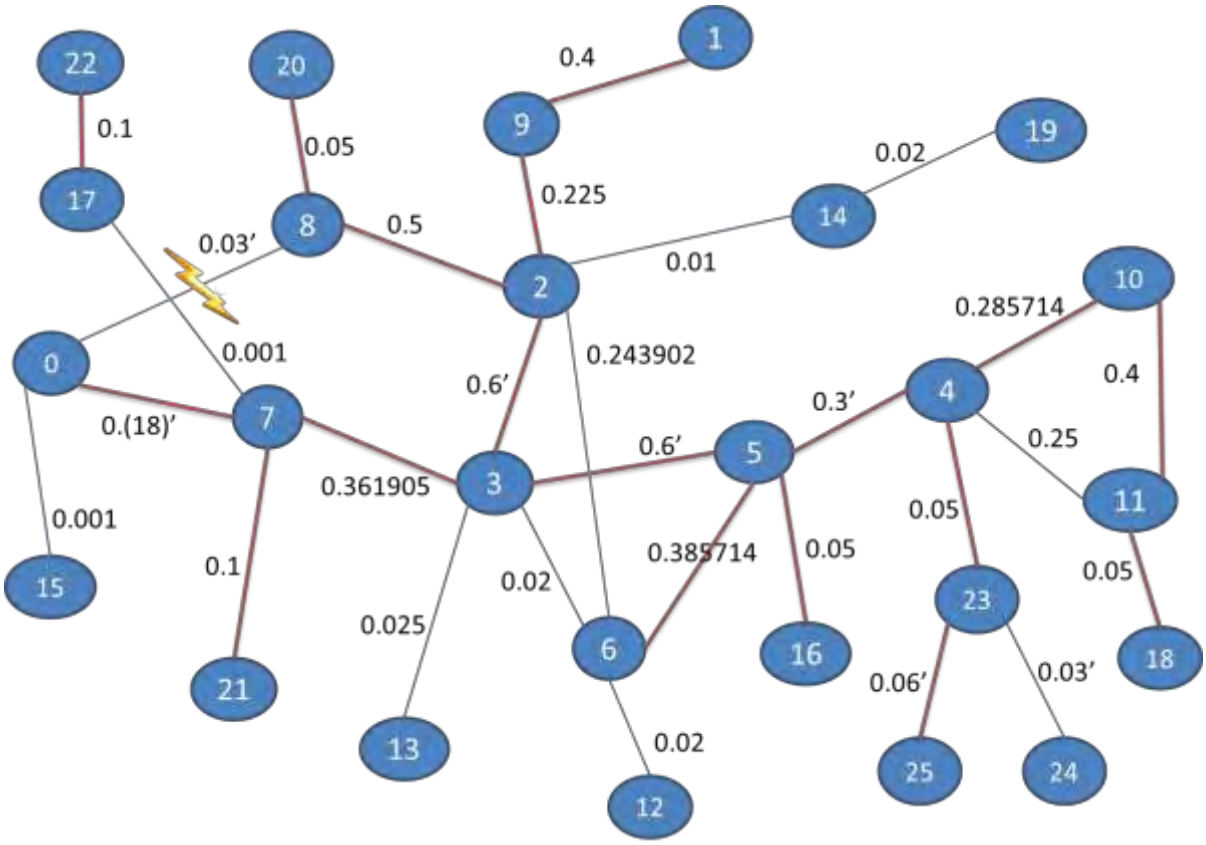
Όμως και η επόμενη ακμή, η 0.243902 (2-6), σχηματίζει κύκλο (εικόνα 5.6).



Εικόνα 5.6: Σχηματισμός κύκλου 2-3-5-6-2

Ο έλεγχος της επόμενης ακμής είναι θετικός και εισάγουμε συνεχόμενα στο T τις ακμές 0.225 (2-9), 0.(18)' (0-7), 0.1 (7-21), 0.1 (17-22), 0.06' (23-25), 0.05 (4-23), 0.05 (5-16), 0.05 (8-20), και 0.05 (11-18).

Όταν φτάνουμε στον έλεγχο της ακμής με βάρος 0.03' που ενώνει τους κόμβους 0-8, βλέπουμε (εικόνα 5.7) ότι σχηματίζεται κύκλος οπότε δεν την εισάγουμε στο T και προχωράμε στην επόμενη ακμή.

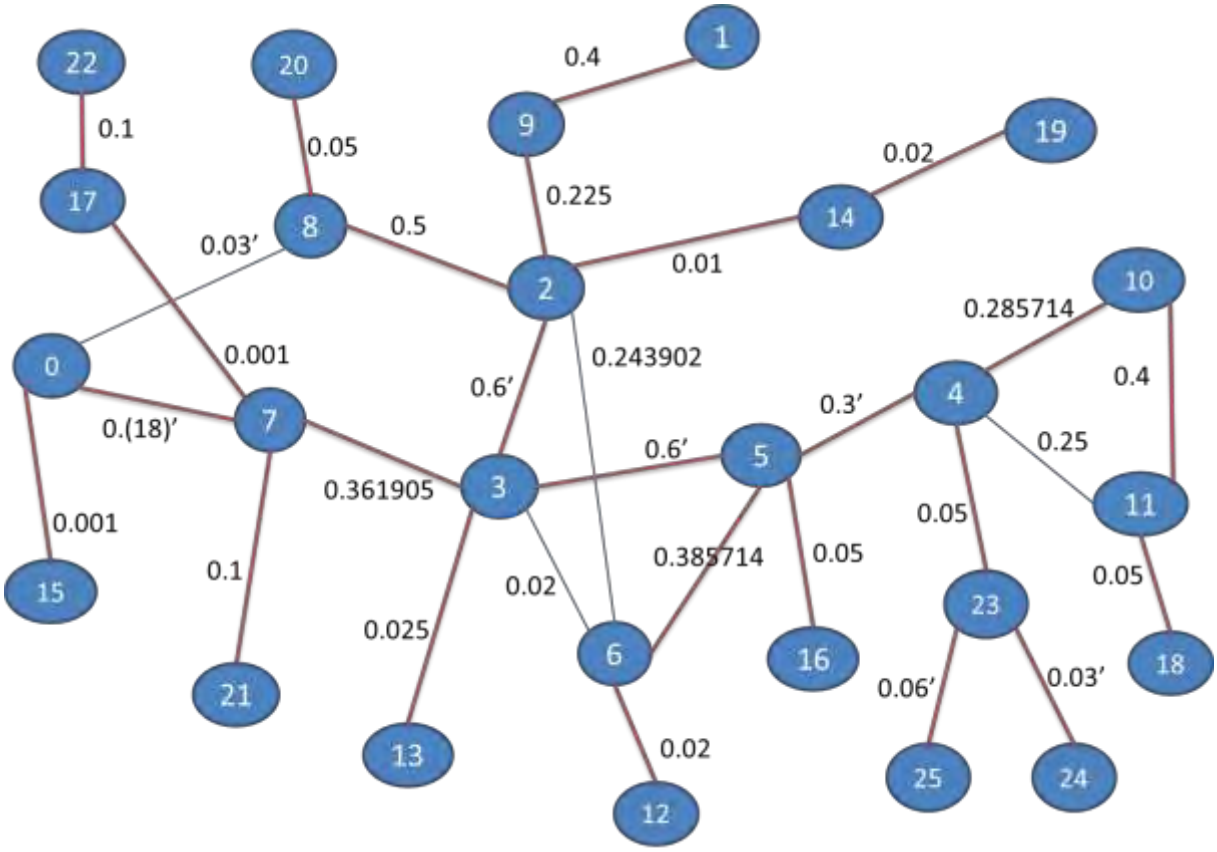


Εικόνα 5.7: Σχηματισμός κύκλου 0-7-3-2-8-0

Στην συνέχεια εισάγουμε στο T τις ακμές 0.03' (23-24) και 0.025 (3-13) και έπειτα συναντάμε τον κύκλο 3-5-6-3 λόγω της ακμής 3-6 βάρους 0.02 την οποία δεν εισάγουμε.

Τέλος προσθέτουμε όλες τις εναπομείνασες ακμές 0.02 (6-12), 0.02 (14-19), 0.01 (2-14), 0.001 (0-15) και 0.001 (7-17) χωρίς κάποιο πρόβλημα.

Το maximum spanning tree που προκύπτει τελικά είναι αυτό της εικόνας 5.8.



Εικόνα 5.8: Τελικό maximum spanning tree

Οπότε όπως βλέπουμε και από το αποτέλεσμα η γενικότερη λογική αυτού του αλγόριθμου είναι όταν υπάρχει κύκλος να αφαιρεί την ακμή αυτού το κύκλου με το μικρότερο βάρος.

Με αυτόν τον τρόπο αφαιρούνται ακμές μικρού βάρους οι οποίες δεν έχουν μεγάλη επίδραση στο τελικό αποτέλεσμα της λύσης.

Αφού λοιπόν υπολογιστεί το *maximum spanning tree*, στον preconditioner πίνακα μπαίνουν μόνο οι ακμές που ανήκουν σε αυτό.

Τέλος προχωράμε στην λύση του συστήματος με την χρήση του preconditioner πίνακα, μέσω της τεχνικής που εξηγήσαμε σε προηγούμενο κεφάλαιο.

Κεφάλαιο 6

Low Stretch Tree Preconditioner

6.1 Preconditioning με *Low Stretch Spanning Trees*

Εισαγωγή στους έννοιες του stretch

Θεωρούμε ένα συνδεδεμένο, μη κατευθυνόμενο σταθμισμένο γράφο $G=(V,E,l)$, όπου $l: E \mapsto \mathbb{R}$ προσδιορίζει ένα μη αρνητικό βάρος $l(e)$ σε κάθε ακμή e, αντιπροσωπεύοντας το μήκος της.

Για ένα spanning tree T του V, ορίζουμε την απόσταση στο T μεταξύ ενός ζεύγους κορυφών $u, v \in V$, που συμβολίζεται $dist_T(u,v)$, να είναι το άθροισμα των μηκών των ακμών στο μοναδικό μονοπάτι στο T μεταξύ του u και του v.

Μπορούμε τώρα να ορίσουμε το stretch[2] μίας ακμής $(u,v) \in E$ να είναι

$$stretch_T(u,v) = \frac{dist_T(u,v)}{l(u,v)}.$$

Μας ενδιαφέρουν δύο όροι αυτού. Ο πρώτος είναι το *maximum stretch* του δέντρου T,

$$max\_stretch(T) = max_{u,v}\{stretch_T(u,v)\}.$$

Δοθέντος ενός γράφου G, το spanning tree T που ελαχιστοποιεί το $max\_stretch(T)$ αναφέρεται ως το *minimum max stretch tree* (MMST) του G. Το πρόβλημα *MMST*, που παρουσιάζεται στο [14], είναι το να βρεθεί το MMST ενός δοθέντος γράφου G.

[2] Ο ορισμός μας του stretch διαφέρει ελαφρώς από αυτόν που χρησιμοποιείται στο [13]: $dist_T(u,v)/dist_G(u,v)$, όπου $dist_G(u,v)$ είναι το μήκος του shortest path μεταξύ u και v.

Ο δεύτερος όρος είναι το *average stretch* του γράφου T,

$$ave\_stretch\ (T) = \frac{1}{\binom{n}{2}} \sum_{u,v \in E} stretch(u, v).$$

Το spanning tree T που ελαχιστοποιεί το $ave\_stretch\ (T)$ αναφέρεται ως το *minimum average stretch tree* (MAST) του G. Το πρόβλημα MAST, που παρουσιάζεται στο [13], είναι το να βρεθεί το MAST ενός δοθέντος γράφου.

Λύνοντας γραμμικά συστήματα

Ο Alon στο [13] απέδειξε ότι κάθε συνδεδεμένος σταθμισμένος γράφος $G = (V, E, l)$ n κορυφών και m ακμών περιέχει ένα spanning tree T τέτοιο ώστε

$$ave\text{-}stretch_T(E) = exp\left(O\left(\sqrt{log\ n\ log\ log\ n}\right)\right)$$

και ότι υπάρχει μια συλλογή $\tau = \{T_1, \ldots, T_h\}$ από spanning trees του G και μια κατανομή πιθανότητας Π του τα τέτοια ώστε κάθε ακμή $e \in E$,

$$E_{T \leftarrow \Pi}[stretch_T(e)] = exp\left(O\left(\sqrt{log\ n\ log\ log\ n}\right)\right).$$

Οι Boman και Hendrickson [15] ήταν οι πρώτοι που συνειδητοποίησαν ότι τα *low-stretch spanning trees* μπορούσαν να χρησιμοποιηθούν για να λύσουν συμμετρικά με κυρίαρχη διαγώνιο συστήματα(SDD). Εφάρμοσαν τα spanning trees του [13] για να σχεδιάσουν επιλυτές που εκτελούνται σε χρόνο

$$m^{3/2} 2^{O\left(\sqrt{\log n \log\log n}\right)} log(1\ /\ \epsilon)$$

όπου ϵ είναι η ακρίβεια της λύσης.

Οι Spielman και Teng [18] βελτίωσαν τα αποτελέσματά τους σε

$$m 2^{O\left(\sqrt{\log n \log\log n}\right)} \log(1\ /\ \epsilon)$$

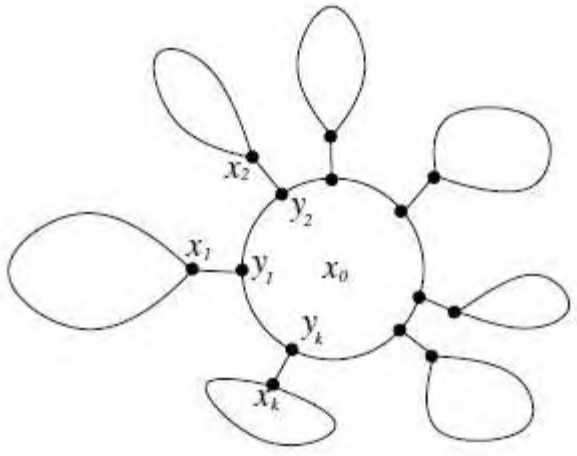
Δυστυχώς, τα δέντρα που παράγονται από τους αλγορίθμους των Bartal [20, 21], και Fakcharoenphol, Rao και Talwar [19], δεν μπορούν να χρησιμοποιηθούν για τη βελτίωση αυτών των γραμμικών επιλυτών, και αυτή τη στιγμή δεν είναι γνωστό αν είναι δυνατή η επίλυση γραμμικών συστημάτων αποτελεσματικά χρησιμοποιώντας τα δέντρα που δεν είναι υπογράφοι. Με την εφαρμογή των *low-stretch spanning trees* που αναπτύσσονται σε αυτό το έγγραφο, μπορούμε να μειώσουμε το χρόνο για την επίλυση αυτών των γραμμικών συστημάτων σε

$$m \log^{O(1)} n \log(1 / \epsilon)$$

και σε $O(n \log^2 n \log\log n \log(1 / \epsilon))$ όταν τα συστήματα είναι επίπεδα.

Η τεχνική

Για την δημιουργία των *low-stretch spanning trees* χρησιμοποιούμε την αναδρομική εφαρμογή μίας νέας τεχνικής αποσύνθεσης γράφων που ονομάζουμε *star-decomposition*. Η *star-decomposition* ενός γράφου είναι ένας διαχωρισμός των κορυφών σε σετ που ενώνονται σε μορφή αστεριού: ένα κεντρικό σετ συνδέεται με κάθε άλλο σετ μέσω μια μοναδικής ακμής (βλέπε εικόνα 6.1).



6.1: *Star-decomposition*

Εξηγούμε πώς να βρούμε *star-decompositions* που να μην κόβουν πάρα πολλές ακμές μικρού βάρους και έτσι ώστε η ακτίνα του γράφου που εξάχθηκε από την *star-decomposition* να μην είναι πολύ μεγαλύτερη από την ακτίνα του αρχικού γράφου.

Ο αλγόριθμος που χρησιμοποιείται σε αυτήν την διπλωματική εργασία για την εύρεση ενός *star-decomposition* χαμηλού κόστους, εφαρμόζει μία γενίκευση της ball-growing τεχνικής του Awerbuch [22] για την ανάπτυξη cones, όπου cone σε μία κορυφή x που προκαλείται από ένα σετ κορυφών του S είναι ένα σετ κορυφών των οποίων το shortest path στο S περνάει μέσω του x.

Ορολογία

Στην συνέχεια της εργασίας θα υποθέτουμε ότι ο γράφος εισόδου είναι ένας σταθμισμένος συνδεδεμένος πολυγράφος $G=(V,E,l)$, όπου l είναι μία *length* συνάρτηση από το E στους θετικούς πραγματικούς αριθμούς.

Εκτός αν οριστεί διαφορετικά, θεωρούμε ότι n και m αποτελούν τον αριθμό των κορυφών και ακμών του γράφου αντίστοιχα.

Το *cost*, ή αλλιώς κόστος, μιας ακμής $e \in E$, ορίζεται ως το αντίστροφο του μήκους της και συμβολίζεται $cost(e) = 1/l(e)$. Το cost ενός σετ $F \subseteq E$ ακμών, που συμβολίζεται $cost(F)$, είναι το άθροισμα των *costs* των ακμών του F.

Έστω u,v είναι δύο κορυφές του V. Ορίζουμε *distance* μεταξύ των u και v, που συμβολίζεται $dist(u,v)$, το μήκος του συντομότερου μονοπατιού μεταξύ u και v του G. Το γράφουμε ως $dist_G(u,v)$ για να τονίσουμε ότι η *distance* είναι στον γράφο G. Για ένα υποσετ κορυφών $S \subseteq V$, ορίζουμε την απόσταση μεταξύ u και S ως $dist_G(u,S)=min\{dist_G(u,x)|x \in S\}$.

$E(S)$ είναι το σετ των ακμών που έχουν και τα δύο άκρα τους στο S.

Boundary του S, που συμβολίζεται $\partial(S)$, είναι ένα σετ ακμών που έχουν ακριβώς ένα άκρο στο S.

Αν T είναι ένα σετ κορυφών και $S \cap T = \emptyset$, τότε $E(S,T)$ είναι ένα σετ ακμών με μία άκρη στο S και την άλλη στο T.

Το *volume* ενός σετ ακμών F, που συμβολίζεται $vol(F)$, είναι το μέγεθος του σετ $|F|$. Το *volume* ενός σετ κορυφών S, που συμβολίζεται $vol(S)$, είναι ο αριθμός ακμών με τουλάχιστον ένα άκρο στο S

Έστω v είναι μία κορυφή στο V. Το radius του G με βάση το v, που συμβολίζεται $rad_G(v)$, είναι το μικρότερο r τέτοιο ώστε κάθε κορυφή του G βρίσκεται σε απόσταση (το πολύ) r από την v.

Ball ακτίνας r γύρω από την v, που συμβολίζεται $B(r,v)$, είναι το σετ κορυφών σε απόσταση το πολύ r από την v.

Ball shell ακτίνας r γύρω από την v, που συμβολίζεται $BS(r,v)$, είναι το σετ κορυφών ακριβώς έξω από τη $B(r,v)$. Αυτό σημαίνει ότι το $BS(r,v)$ αποτελείται από κάθε κορυφή $u \in V - B(r,v)$ με έναν γείτονα $w \in B(r,v)$ τέτοιο ώστε $dist(v, u) = dist(v,w) + l(u,w)$.

6.2 *Star Decomposition*

Ορισμός *star-decomposition*

Μία πολυκατευθυντική διαμοίραση $\{V_0, ..., V_k\}$ είναι *star-decomposition* ενός σταθμισμένου συνδεδεμένου γράφου $G=(V, E, l)$ με κέντρο $x_0 \in V$ (βλέπε εικόνα 6) εάν $x_0 \in V_0$ και

1. για όλα τα $0 \leq i \leq k$, ο επαγόμενος υπογράφος του V_i είναι συνδεδεμένος και
2. για όλα τα $i \geq 1$, το V_i περιέχει μία κορυφή άγκυρα x_i που είναι συνδεδεμένη με μία κορυφή $y_i \in V_i$ μέσω μιας ακμής $(x_i, y_i) \in E$. Ονομάζουμε την ακμή (x_i, y_i) την *γέφυρα* ανάμεσα στα V_0 και V_i.

Ορισμός *(δ, ϵ)-star-decomposition*

Έστω $r = rad_G(x_0)$, και $ri = rad_{V_i}(xi)$ για κάθε $0 \leq i \leq k$. Για $\delta, \epsilon \leq 1/2$, μία *star-decomposition* $\{V_0, \ldots, V_k\}$ *είναι μια* (δ, ϵ)*-star-decomposition* εάν

a. $r_0 \leq (1-\delta)r$
b. $dist(x_0,x_i) \geq \delta_r$ *για κάθε* $i \geq 1$ *και*
c. $dist(x_0,x_i) + r_i \leq (1+\epsilon)r$ *για κάθε* $i \geq 1$

Το *cost* του *star-decomposition* είναι $cost(\partial(V_0, \ldots, V_k))$

Ο αλγόριθμος διαίρει και βασίλευε

Η βασική ιδέα του αλγόριθμου είναι να χρησιμοποιήσουμε *low-cost star-decompositions* σε έναν διαίρει και βασίλευε (επαναληπτικό) αλγόριθμο για να δημιουργήσουμε ένα spanning tree. Έστω $G=(V, E, l)$ είναι ο γράφος είσοδος στην παρούσα αναδρομική κλήση του αλγόριθμου. Έστω επίσης ότι με $\tilde{n}$ και $\tilde{m}$ συμβολίζεται ο αριθμός των κορυφών και ακμών, αντίστοιχα, του αρχικού γράφου, είσοδος της πρώτης αναδρομικής κλίσης.

Ο υπάρχων ψευδοκώδικας του αναδρομικού αλγόριθμου *low stretch tree* είναι ο:

Fix $\beta = (2\lceil \log_{4/3}(2\tilde{n}+32)\rceil)^{-1}$.
$T = \text{LowStretchTree}(G=(V,E,l), x_0)$.

1. $(\{V_0, ..., V_k\}, x, y) = \text{StarDecomp}(G, x_0, 1/3, \beta)$.
2. For $0 \le i \le k$, set $T_i = \text{LowStretchTree}(G(V_i), x_i)$.
3. Set $T = U_i T_i$

Η βασική ιδέα της *StarDecomp* είναι να χρησιμοποιήσει πρώτα την τεχνική του *BallCut* για να δημιουργήσει το V_0 και στην συνέχεια να εφαρμόσει επαναληπτικά *ConeCut* για να δημιουργήσει τα $V_0, ..., V_k$.

Ο υπάρχων ψευδοκώδικας της *StarDecomp* είναι ο :

$(\{V_0, ..., V_k\}, x, y) = \textit{StarDecomp}(G=(V,E,l), x_0, \delta, \epsilon)$.

1. Set $\rho = \text{rad}_G(x_0)$;
   set $r_0 = \text{BallCut}(G, x_0, \rho, \delta)$ and $V_0 = B(r_0, x_0)$.
2. Let $S = BS(r_0, x_0)$.
3. Set $G''=(V',E',l')=G(V-V_0)$, the weighted graph induced by $V-V_0$.
4. Set $(\{V_1, ..., V_k, x\}) = \textit{ConeDecomp}(G', S, \epsilon\rho/2)$.
5. For each $i \in [1:k]$,
   set y_i to be a vertex in V_0 such that $(x_i, y_i) \in E$
   and y_i is on a shortest path from x_0 to x_i.
   Set $\mathbf{y} = (y_1, ..., y_k)$

Ο υπάρχων ψευδοκώδικας της *ConeDecomp* είναι ο :

$(\{V_0, ..., V_k\}, x) = ConeDecomp(G, S, \Delta)$.

1. Set $G_0=G$, $S_o=S$, and $k=0$.
2. While S_k is not empty,
   (a) Set $k=k+1$
   (b) Let x_k be an arbitrary vertex in S_{k-1}
   and set $r_k=ConeCut(G_{k-1}, x_k, 0, \Delta, S_{k-1})$.
   (c) Set $V_k=C_{S_{k-1}}(r_k, x_k)$.
   Set $G_k=G(V - \bigcup_{i=1}^{k} V_k)$ and $S_k=S_{k-1} - V_k$.
3. Set $\mathbf{x}=(x_1, ..., x_k)$.

Ο υπάρχων ψευδοκώδικας της *BallCut* είναι ο :

$r = BallCut(G, x, \rho, \delta)$.

1. Set $r = \delta\rho$.
2. While $cost(\partial(B(r,x))) > \frac{vol(B(r,x))+1}{(1-2\delta)\rho} \log_2(m+1)$,
   (a) Find a vertex $v \notin B(r, x)$ that minimizes $dist(x, v)$ and set $r = dist(x, v)$.

Ορισμός ideals και cones

Για κάθε σταθμισμένο γράφο $G=(V, E, l)$ και $S \subseteq V$, το σετ των εμπρόσθιων ακμών που επάγονται από το S είναι

$$F(S) = \{(u \to v) : (u, v) \in E, dist(u, S) + l(u, v) = dist(v, S)\}.$$

Για μια κορυφή $v \in V$, το ideal της v που επάγεται από το S, με συμβολισμό $I_S(v)$, είναι το σετ κορυφών που είναι προσεγγίσιμες από την v μέσω κατευθυνόμενων ακμών στο στο $F(S)$, συμπεραμβαλομένης και της v.

Για μια κορυφή $\upsilon \in V$, το cone μήκους l γύρω από την υ που επάγεται από το S, με συμβολισμό $C_S(l,\upsilon)$, είναι ένα σετ από κορυφές στο V που μπορεί να προσεγγιστεί από την υ μέσω ενός μονοπατιού, το άθροισμα των μηκών των οποίων ακμών e που δεν ανήκουν στο $F(S)$ είναι το πολύ l . Προφανώς, $C_S(0,\upsilon) = I_S(\upsilon)$ για όλες τις $\upsilon \in V$.

Ο ψευδοκώδικας της *ConeCut* είναι ο :

$r = ConeCut(G, \upsilon, \lambda, \lambda', S)$.

1. Set $r = \lambda$.
2. If $vol(E(C_S(\lambda, \upsilon))) = 0$,
   then set $\mu = (vol(C_S(r, \upsilon)) + 1) \log_2(m+1)$.
3. Otherwise, set $\mu = vol(C_S(r,\upsilon)) \log_2(m/vol(E(C_S(\lambda, \upsilon))))$.
4. While $cost(\partial(C_S(r, \upsilon))) > \mu/(\lambda' - \lambda)$,
   (a) Find a vertex $w \notin C_S(r, \upsilon)$ minimizing $dist(w, C_S(r, \upsilon))$
   and set $r = r + dist(w, C_S(r, \upsilon))$.

6.3 Υλοποίηση του *Low Stretch Tree* Preconditioner

Πέρασμα τιμών σε γράφο

Το πρώτο βήμα της δημιουργίας του *support-graph* preconditioner είναι να περάσουμε τα δεδομένα του πίνακα A στον γράφο. Να σημειώσουμε εδώ ότι ο πίνακας πρέπει να είναι SPD. Ο πίνακας A περιέχει τις αγωγιμότητες των αντιστάσεων του κυκλώματος. Έτσι διατρέχουμε τον πίνακα του συστήματος MNA και εισάγουμε, με την κατάλληλη συνάρτηση, τις αρνητικές και αντίστροφες τιμές των μη μηδενικών και εκτός διαγωνίου στοιχείων στον γράφο. Θεωρούμε ότι αν $A[i][j]=value$, τότε ο κόμβος i συνδέεται με τον κόμβο j με ακμή βάρους –(1/value).

Τον γράφο τον αναπαριστούμε με έναν πίνακα τόσων θέσεων όσος ο αριθμός των κόμβων *n*, ο οποίος αναπαριστάται από δείκτες σε στοιχεία struct. Με αυτόν τον τρόπο κάθε θέση του πίνακα αναπαριστά έναν κόμβο του γράφου και τα πεδία struct που δείχνει αυτή η θέση αναπαριστούν τους γείτονες αυτού του κόμβου με τις πληροφορίες που αυτοί περιέχουν (βάρος ακμής που συνδέει τον τρέχον κόμβο με τον κάθε γείτονα). Αυτή η δομή μας βολεύει γιατί δεν ξέρουμε εκ των προτέρων πόσους γείτονες έχει ο κάθε κόμβος του γράφου αλλά τους προσθέτουμε σταδιακά.

Συναρτήσεις που χρησιμοποιήθηκαν

Στον κώδικα της δημιουργίας του *low stretch* preconditioner υλοποιήθηκαν και χρησιμοποιήθηκαν οι παρακάτω συναρτήσεις.

- *LowStretchTree*

  Έπειτα από την δημιουργία του γράφου, η αρχική κεντρική αναδρομική συνάρτηση που καλείται είναι η *LowStretchTree* . Η συνάρτηση αυτή όπως αναφέρθηκε παραπάνω με βάση την κεντρική κορυφή x_0 που δίνεται σαν όρισμα στην κάθε αναδρομή καλεί τη συνάρτηση δημιουργίας της *star decomposition*, ώστε να γίνει η αποσύνθεση του δοθέντος γράφου κάθε φορά .

- *StarDecomp*

  Η *StarDecomp* , υπεύθυνη για την αποσύνθεση του δοθέντος γράφου κάθε φορά, καλεί με την σειρά της τις συναρτήσεις που δημιουργούν την αποσύνθεση εξάγοντας το κεντρικό ball και τα περιμετρικά cones αυτού, με βάση το δοθέν x_0. Επίσης, για κάθε cone προσδιορίζει την ακμή γέφυρα που το συνδέει με το κεντρικό ball.

- *BallCut*

  Η συνάρτηση αυτή η υλοποιεί τον αλγόριθμο του BallCut, με τον τρόπο που αναλύθηκε παραπάνω, υπολογίζοντας την ακτίνα του κεντρικού ball με βάση την κορυφή x_0 που μας δόθηκε.

- *ConeDecomp*

  Η *ConeDecomp* βρίσκει επαναληπτικά έναν περιμετρικό κόμβο x_k του κεντρικού ball και καλεί την συνάρτηση δημιουργίας cone ώστε να παραχθεί ένα cone με βάση τον x_k.

- *ConeCut*

  Η συνάρτηση αυτή υλοποιεί τον αλγόριθμο του ConeCut, με τον τρόπο που αναλύθηκε παραπάνω, πραγματοποιώντας την δημιουργία cone με βάση την κορυφή x_k που δόθηκε σαν είσοδο.

- *Make ball*

  Αυτή η αναδρομική συνάρτηση δημιουργεί το κεντρικό ball με βάση την ακτίνα που υπολογίστηκε από την συνάρτηση BallCut.

- *BallSearch*

  Η αναδρομική συνάρτηση αυτή πραγματοποιεί αναζήτηση για την εύρεση κόμβου που έχει την μικρότερη απόσταση από τον κεντρικό κόμβο x_0.

- *BallShell*

  Αυτή η αναδρομική συνάρτηση επιστρέφει, αν υπάρχει, έναν κόμβο που να ανήκει στο καβούκι (shell) του αναφερόμενου ball ή cone, ή αν δεν υπάρχει το αριθμό -1. Το καβούκι ενός σετ κόμβων είναι οι κόμβοι που βρίσκονται ακριβώς έξω από αυτό, και έχουν τουλάχιστον έναν γείτονα που να ανήκει σε αυτό το σετ.

- *CalculateEdges*

  Η αναδρομική αυτή συνάρτηση μετρά πόσες ακμές υπάρχουν που να έχουν και τα δύο άκρα τους μέσα στο δοθέν cone.

- *Dist*

  Σε αυτήν την αναδρομική συνάρτηση υπολογίζεται η απόσταση κάθε κόμβου από τον δοθέν κόμβο.

- *Radius*

  Αυτή η συνάρτηση καλείται για την εύρεση του $rad_G(x_0)$, δηλαδή του μικρότερου r τέτοιου ώστε κάθε κορυφή στο G να είναι σε απόσταση (το πολύ) r από το x_0 .

- *Boundary*

  Αυτή η αναδρομική συνάρτηση υπολογίζει το σετ των ακμών που έχουν το ένα άκρο τους στο τρέχον ball ή cone και το άλλο άκρο εκτός αυτού.

- *VerticesLeft*

  Αυτή η συνάρτηση ελέγχει αν υπάρχει κάποιος κόμβος ο οποίος δεν εντάχθηκε με βάσει την συνθήκη στο τρέχον cone, αλλά το τρέχον cone είναι το μόνο με το οποίο μπορεί να συνδεθεί λόγο μη ύπαρξης άλλου shell κόμβου γύρω του. Αυτός ο έλεγχος γίνεται ώστε να μην υπάρξει κάποιος κόμβος που να μην έχει ενταχθεί σε κανένα cone ή ball. Αν αν υπάρχει τέτοιος κόμβος επιστρέφει την τιμή 1, αλλιώς 0.

- *Path*

  Αυτή η αναδρομική συνάρτηση βοηθάει την*VerticesLeft* στην απόφασή της βρίσκοντας αν υπάρχει κόμβος που δεν ανήκει σε κάποιο cone από τον κόμβο γέφυρα (bridge) και μετά. Κόμβος bridge είναι αυτός που αν αποκόψω μια ακμή ενός γείτονα του, αυτός ο γείτονας δεν έχει άλλο μονοπάτι (path) για αυτόν.

Ο ψευδοκώδικας κάποιων βασικών συναρτήσεων του υλοποιημένου κώδικα είναι :

LowStretchTree(x_0)

$\delta=1/3$

$\beta=(2\lceil \log_{4/3}(2\tilde{n}+32)\rceil)^{-1}$.

1. *StarDecomp(x_0, δ, β)*
2. *For $0 \le i \le k$ LowStretchTree(x_0)*

StarDecomp(x_0, δ, β)

1. *CalculateEdges(x_0)*
2. *ρ=Radius(x_0)*
3. *r_0=BallCut(x_0, ρ, δ)*
4. *Dist(x_0)*
5. *MakeBall($r_0.x_0$)*
6. *ConeDecomp(($\beta*\rho$)/2, x_0)*
7. *Find bridge edge*

BallCut(x_0, ρ, δ)

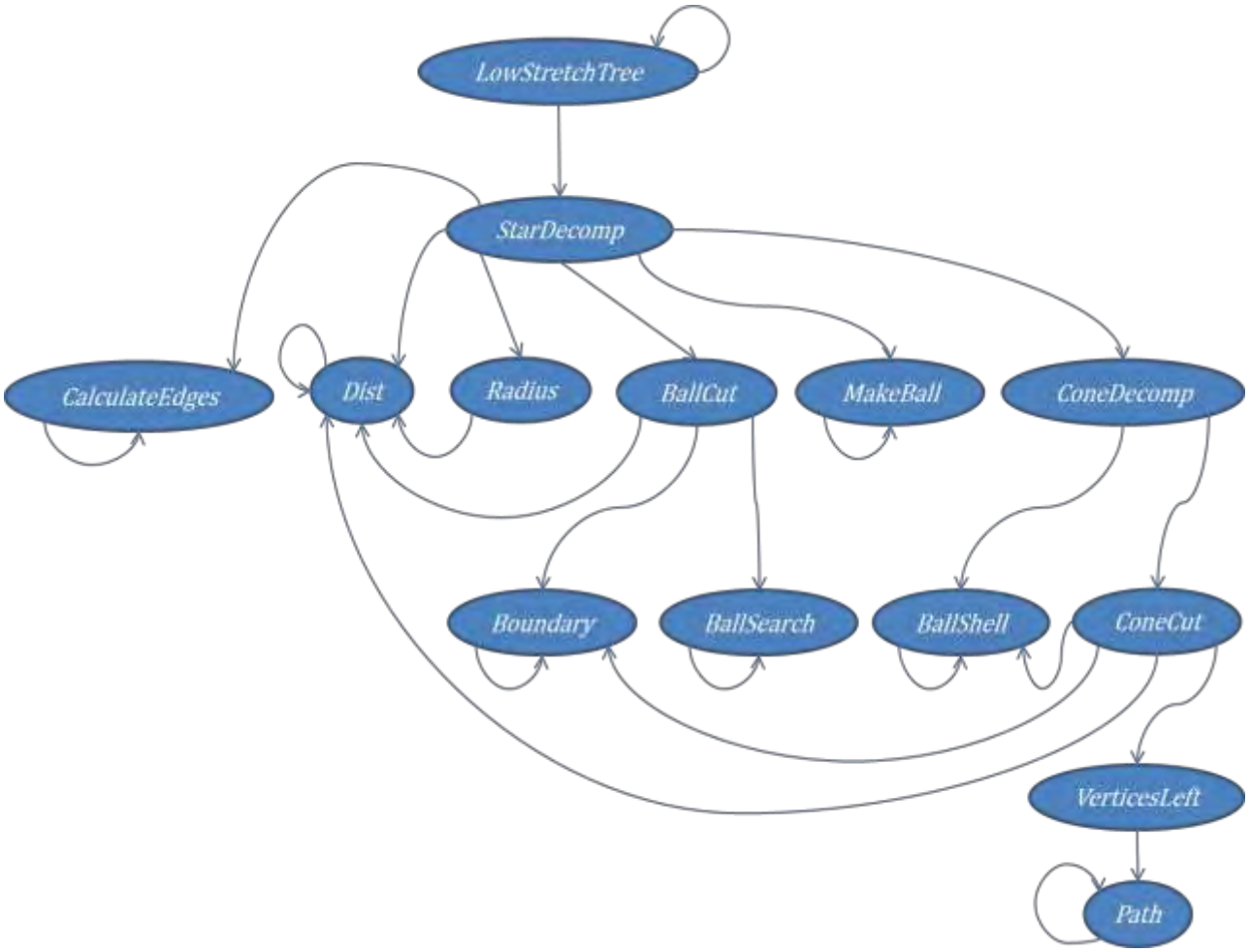
1. *Boundary(x_0)*
2. *While cost($\partial(B(r,x))$) $> \frac{vol(B(r,x))+1}{(1-2\delta)\rho}$ $\log_2(m+1)$*
3. *Dist(x_0)*
4. *BallSearch(x_0)*
5. *Boundary(x_0)*

ConeDecomp(Δ, x_0)

1. *BS=BallShell(x_0)*
2. *while(BS!=-1)*
3. *ConeCut(BS, x_0, Δ)*
4. *BS=BallShell(x_0)*

```
ConeCut(υ, x0, λ, λ')
    1. r=λ
    2. Boundary(υ)
    3. var=VerticesLeft(υ, x0)
    4. while ((cost( ∂ (CS (r, υ) ) ) > μ /(λ' – λ)) || (var==1))
    5.     BS=BallShell(υ, x0)
    6.     if(BS!=-1)
    7.         Dist(υ, x0)
               Find a vertex w ∉ CS (r, υ) minimizing dist (w, CS (r, υ) )
               and set r = r + dist (w, CS (r, υ) )
    8.     var=VerticesLeft(υ, x0)
```

Στην εικόνα φαίνεται το διάγραμμα κλήσεων συναρτήσεων του κώδικα.



Εικόνα 6.2: Διάγραμμα κλήσεων συναρτήσεων

6.5 Εφαρμογή του *Low Stretch Tree* Preconditioner

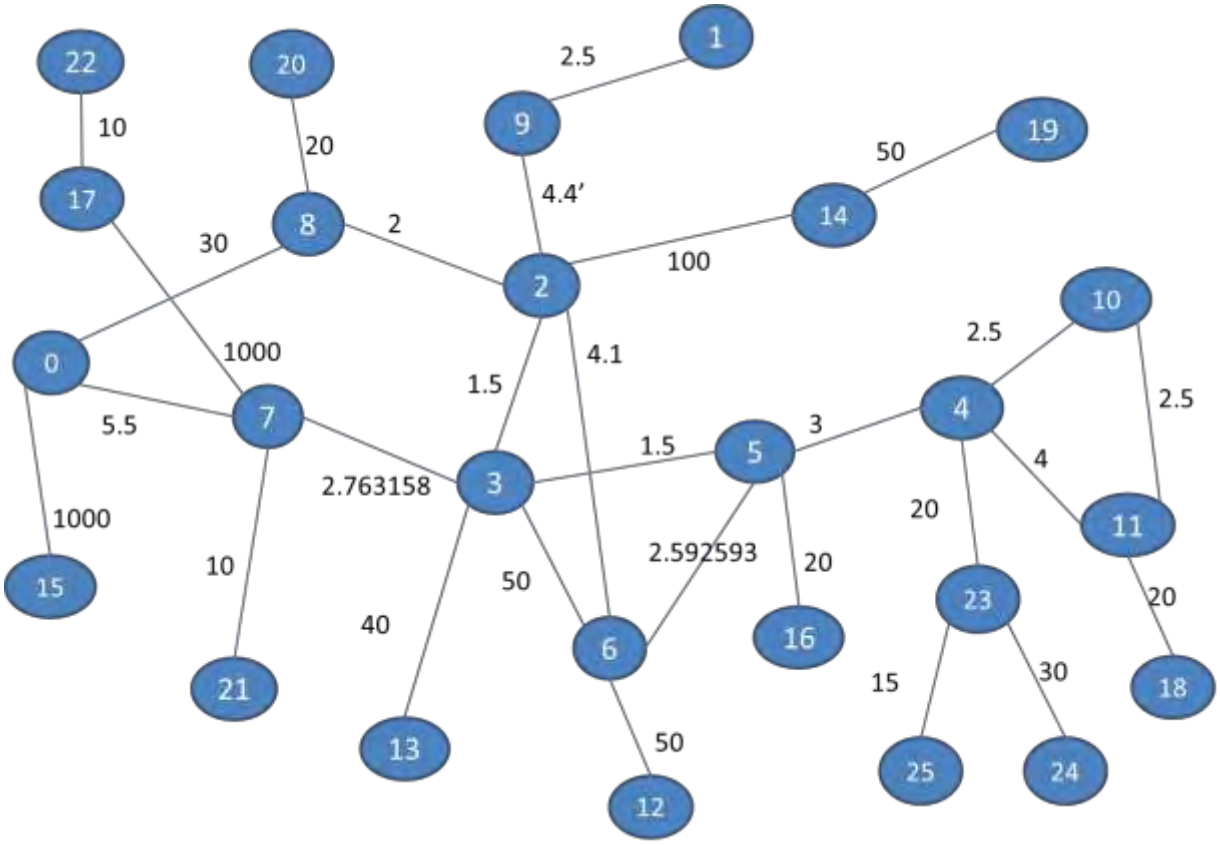
Σε αυτό το σημείο θα εφαρμόσουμε τον αλγόριθμο του *Low stretch tree* preconditioner σε ένα συγκεκριμένο παράδειγμα netlist, έτσι ώστε να γίνει ευκολότερη η κατανόησή της λειτουργίας του.

Το netlist στο οποίο θα εφαρμόσουμε τον αλγόριθμο είναι το:

```
I1  n1_4  n1_8  2e-3
I2  0  n1_6  1e-3
I3  n1_2  n1_10  3e-3
R1  n1_1  n1_5  3.5
R2  n1_1  n1_2  1.5
R3  n1_5  n1_2  50
R4  n1_5  n1_6  4.1
R5  n1_2  n1_6  1.5
R6  n1_3  n1_4  5.5
R7  n1_8  0  1e3
R8  n1_4  0  10
R9  n1_5  0  2
R10  n1_3  n1_2  3
R11  n1_7  n1_6  2
R12  n1_9  n1_8  2.5
R13  n1_1  n1_10  3
R14  n1_10  n1_11  3.5
R15  n1_11  n1_12  2.5
R16  n1_10  n1_12  4
R17  n1_9  n1_6  5
R18  n1_5  n1_1  10
R19  n1_4  n1_7  30
R20  n1_6  n1_9  40
R21  n1_2  n1_3  35
R22  0  n1_8  1e2
R23  n1_13  n1_5  50
R24  n1_14  n1_2  40
R25  n1_15  n1_6  1e2
R26  n1_16  n1_4  1e3
R27  n1_17  n1_1  20
R28  n1_3  n1_18  1e3
R29  n1_12  n1_19  20
R30  n1_15  n1_20  50
R31  n1_21  n1_7  20
R31  n1_3  n1_22  10
R33  n1_23  n1_18  10
R34  n1_25  n1_24  30
R35  n1_25  n1_26  30
R36  n1_25  n1_10  20
.OPTIONS STRETCH 3
```

Όπως βλέπουμε δώσαμε στην επιλογή low-stretch spanning tree σαν όρισμα την τιμή 3. Αυτό σημαίνει ότι θέλουμε η διαδικασία spanning να αρχίσει από τον κόμβο 3 (ονομασία μετά την hash table μετατροπή) . Η τιμή του κόμβου εισόδου είναι αυθαίρετη.

Έπειτα από την δημιουργία του πίνακα αγωγιμοτήτων και το πέρασμα των αρνητικών και αντίστροφων τιμών των στοιχείων του σε γράφο όπως εξηγήθηκε προηγουμένως, προκύπτει ο παρακάτω γράφος απεικόνισης του κυκλώματος:



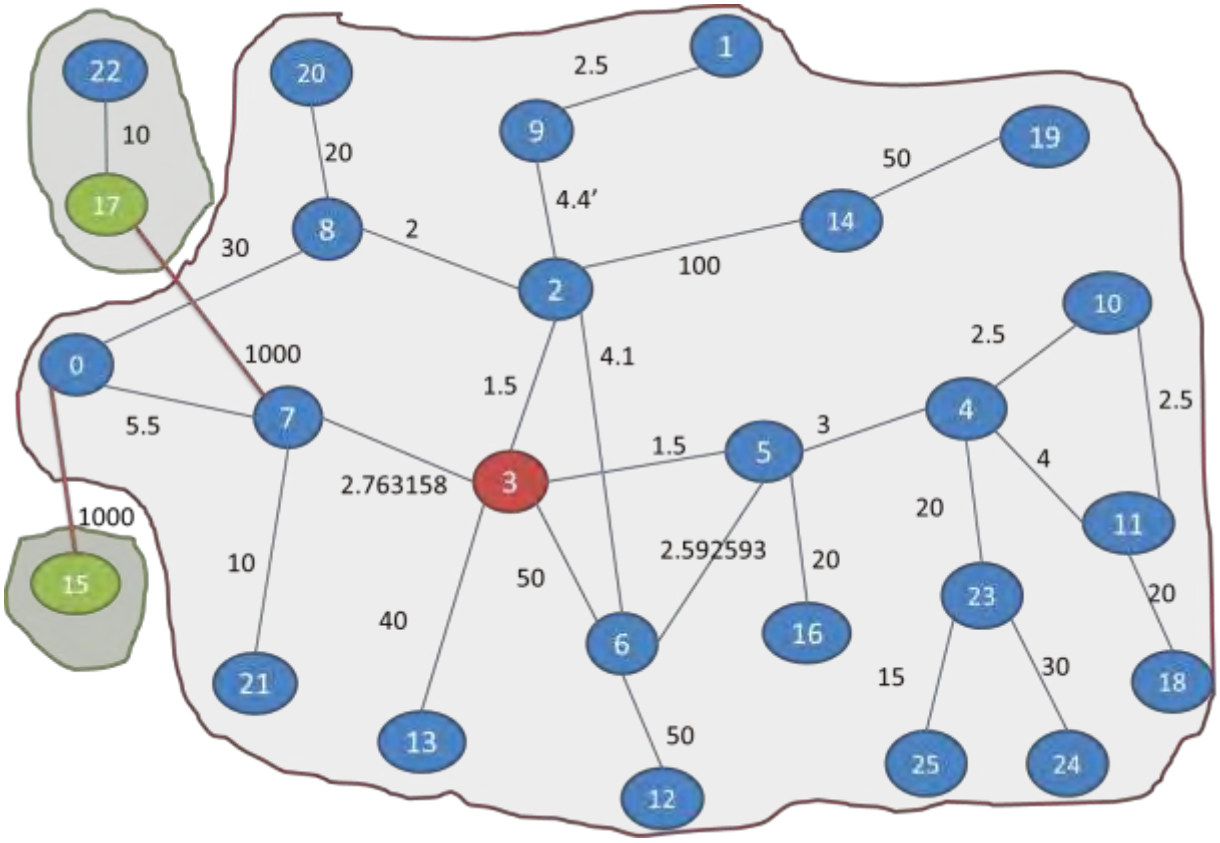
Εικόνα 6.3: Γράφος απεικόνισης του ηλεκτρικού κυκλώματος

Στο αρχικό βήμα της διαδικασίας πραγματοποιείται star decomposition για όλο τον γράφο για κεντρικό κόμβο τον κόμβο 3, ο οποίος μας δόθηκε σαν όρισμα στο netlist (.OPTIONS STRETCH 3).

Θεωρούμε ότι στις παρακάτω εικόνες θα ισχύει ο εξής συμβολισμός :

: center ball, bridge edges , : cones

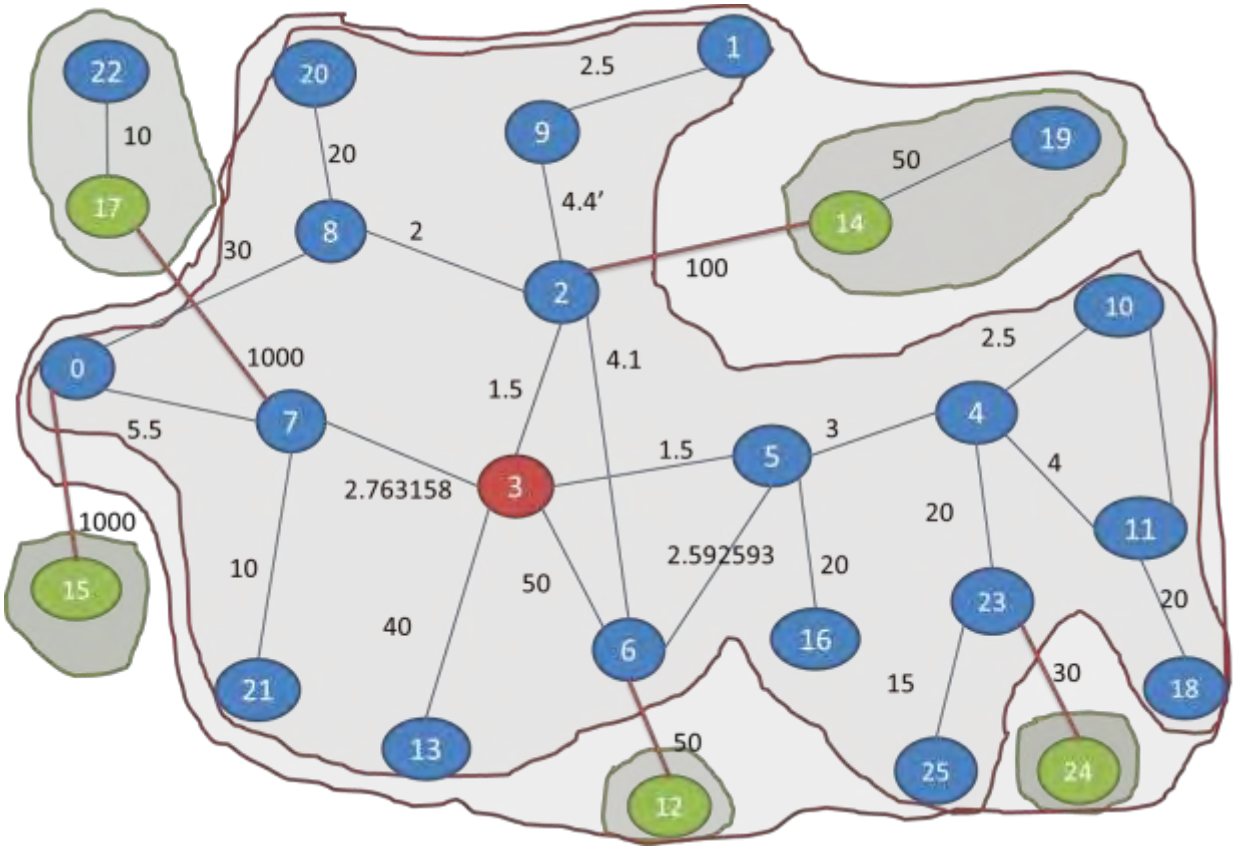
Όπως φαίνεται στην εικόνα δημιουργείτε το κεντρικό ball με κεντικό κόμβο τον 3 και δύο περιμετρικά cones με κεντρικούς κόμβους τους 15 και 17 αντίστοιχα. Επίσης προστίθενται οι bridges ακμές 0-15 και 17-22 στο *low stretch preconditioner tree.*



Εικόνα 6.4: Ball, cones και bridge ακμές μετά την 1η εφαρμογή της *star-decomposition*

Έπειτα, όπως φαίνεται στην εικόνα , στην 1η αναδρομή πραγματοποιείται star decomposition μόνο για το υποδέντρο που αποτελείται από τους κόμβους του ball που δημιουργήθηκε στο αρχικό βήμα.

Έτσι, δημιουργείται πάλι κεντρικό ball με κέντρο τον κόμβο 3 και τρία cones με κεντρικούς κόμβους , τους γειτονικούς του ball, 12, 14 και 24. Έτσι, προστίθενται στο δέντρο του preconditioneroι οι ακμές 6-12, 2-14 και 23-24.

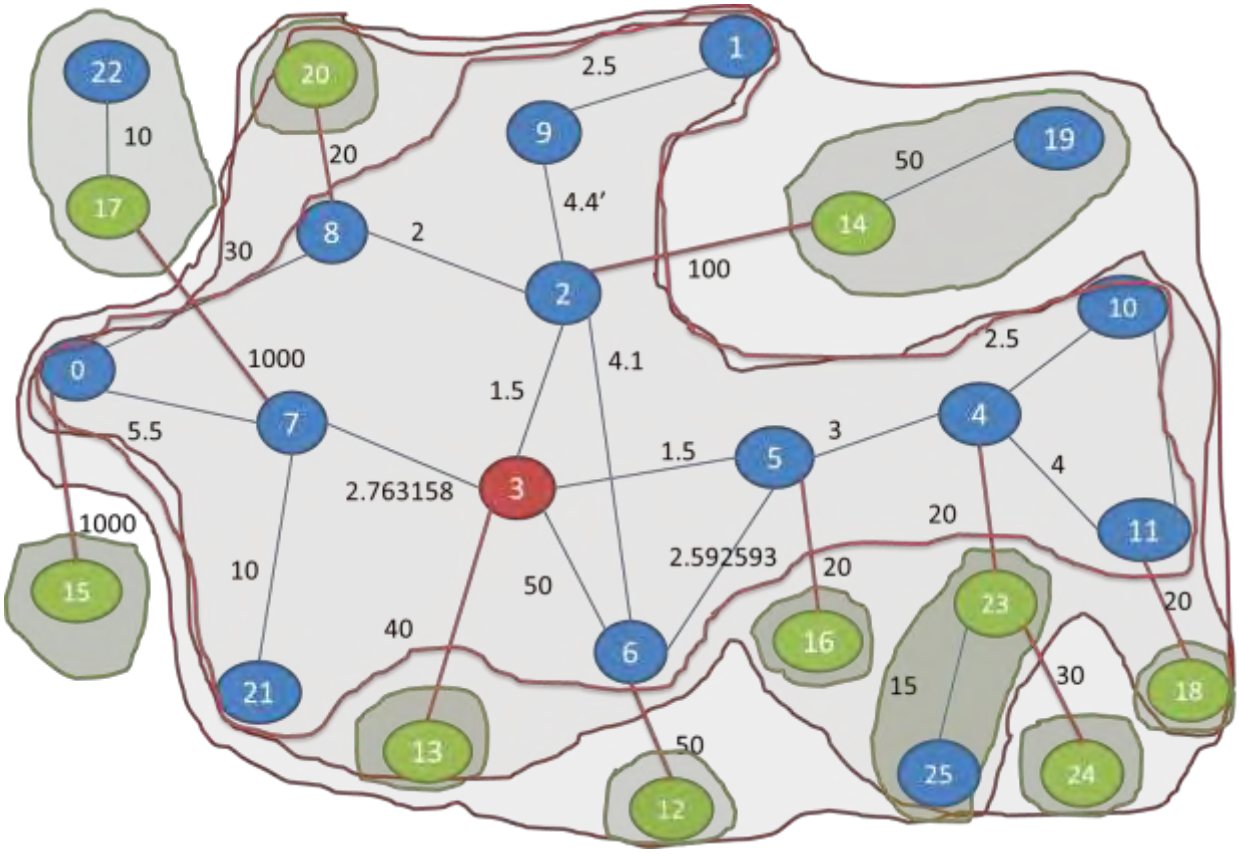


Εικόνα 6.5: Ball, cones και bridge ακμές μετά την 2η εφαρμογή της *star-decomposition*

Στην συνέχεια, πραγματοποιείται *star decomposition* για το υπόδεντρο που αποτελείται από τους κόμβους του καινούριου ball.

Έτσι, σχηματίζεται το κεντρικό ball με κέντο τον κόμβο 3 και τέσσερα cones με κεντρικούς κόμβους τους 13, 16, 20 και 23, όπως φαίνεται στην εικόνα.

Εισέρχονται επίσης οι bridges ακμές 3-13, 5-16, 8-20 και 4-23.

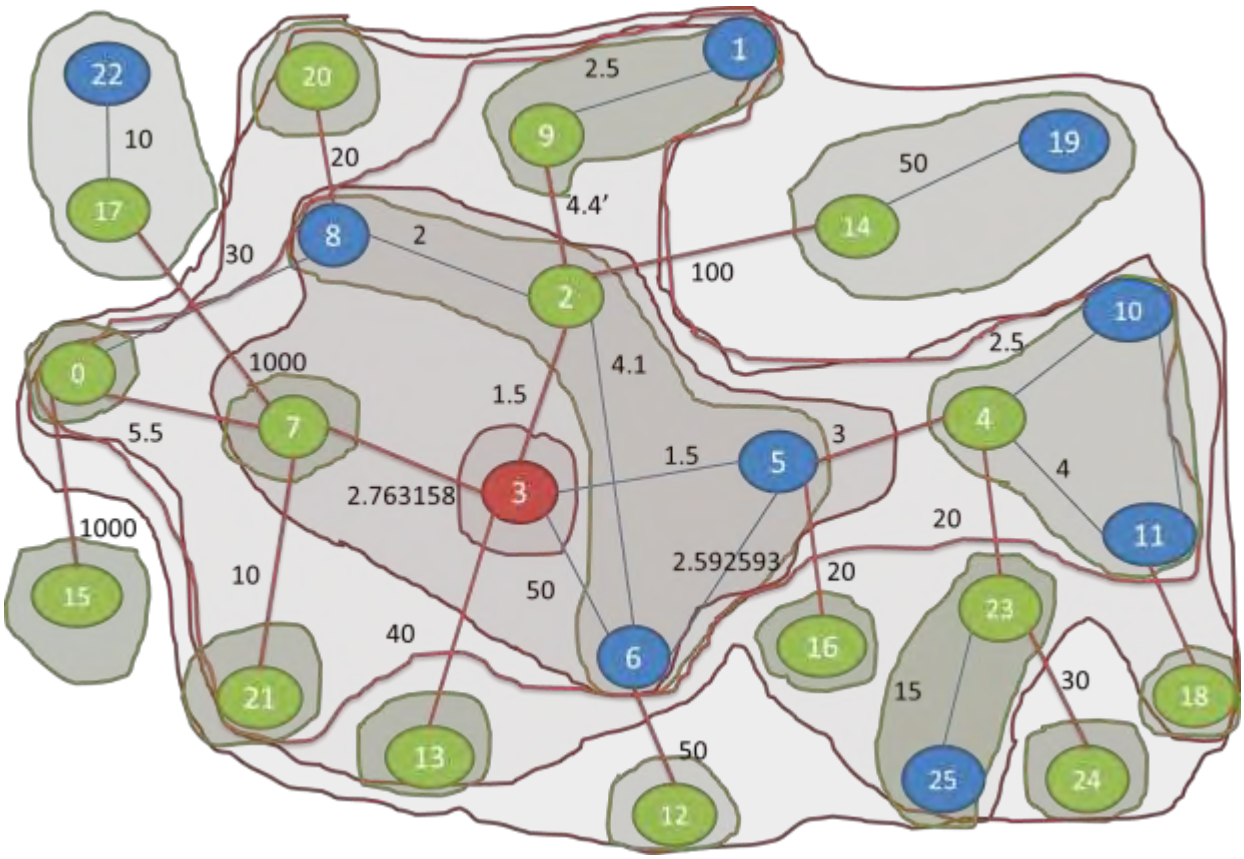


Εικόνα 6.6: Ball, cones και bridge ακμές μετά την 3η εφαρμογή της *star-decomposition*

Έπειτα, ακολουθώντας την ίδια διαδικασία, στο επόμενο βήμα δημιουργείται το ball που αποτελείται από τους κόμβους 2, 3, 5, 6, 7, 8 με κεντρικό κόμβο το 3 και τέσσερα cones με κεντρικούς κόμβους τους 0, 4, 9 και 21, αντίστοιχα. Οι bridge ακμές που εισέρχονται είναι οι 0-7, 4-5, 2-9 και 7-21.

Στο επόμενο βήμα, δημιουργείται το ball με μοναδικό κόμβο τον 3 και δύο cones κεντρικούς κόμβους τους 2 και 7, αντίστοιχα, όπως φαίνεται στην εικόνα. Οι bridge ακμές που εισέρχονται είναι οι 2-3 και 3-7.

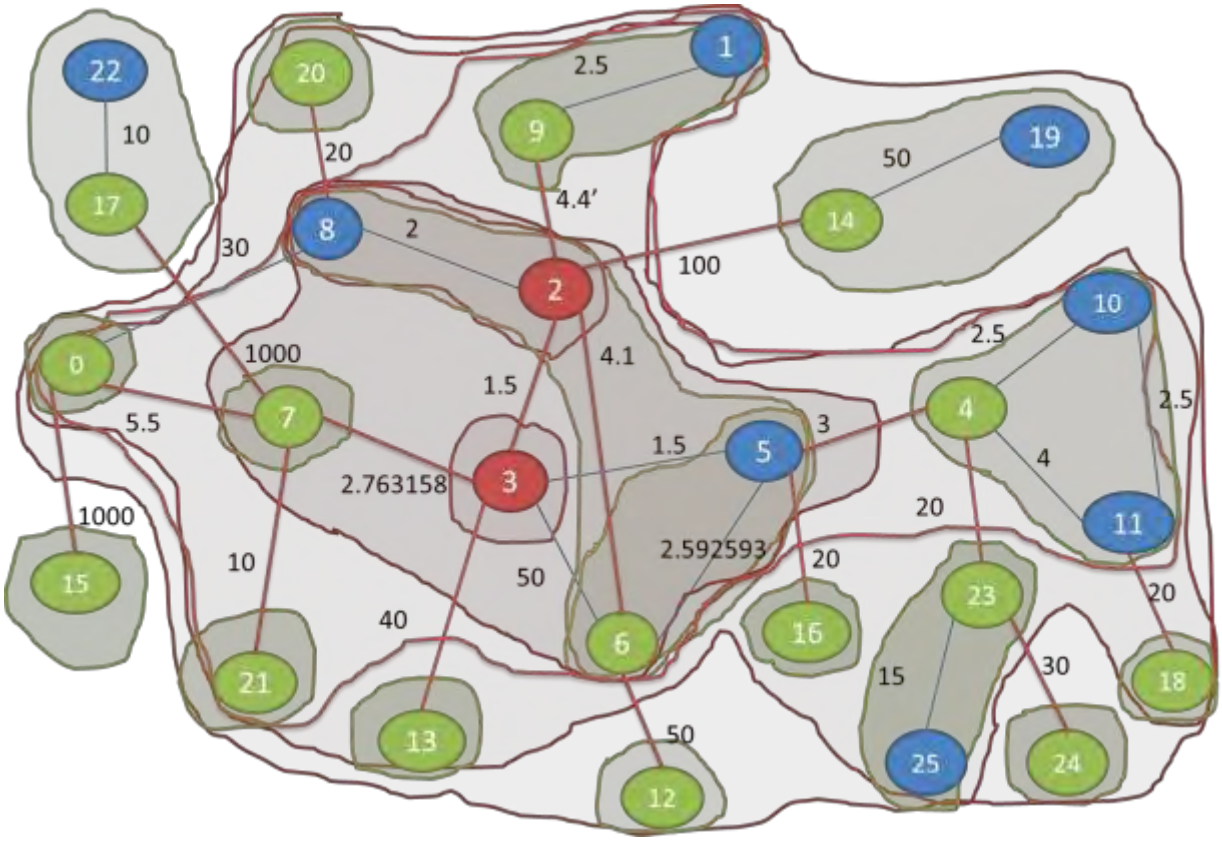
Το γεγονός ότι το ball περιέχει έναν μόνο κόμβο, είναι το σημάδι ότι πρέπει τώρα η αναδρομική διαδικασία της *star decomposition* να ξεκινήσει για τα cones αυτού του ball αφού αυτό δεν έχει άλλους κόμβους για αποσύνθεση.



Εικόνα 6.7: Ball, cones και bridge ακμές μετά την 5η εφαρμογή της *star-decomposition*

'Ετσι στην επόμενο αναδρομή το υπόδεντρο στο οποίο θα εφαρμοστεί η *star decomposition* είναι αυτό που αποτελείται από τους κόμβους του cone που συνδέεται με το τρέχον ball. Παίρνουμε ως σύμβαση εδώ ότι ξεκινάμε από το cone αυτό με τον μικρότερο ονομαστικά κεντρικό κόμβο. Το cone αυτό εδώ είναι το αποτελούμενο από τους κόμβους 2, 5, 6, και 8 με κεντρικό κόμβο τον 2.

Σε αυτό το βήμα λοιπόν θα δημιουργηθεί ένα ball με κεντρικό κόμβο τον 2, αποτελούμενο από τους κόμβους 2 και 8 και ένα cone με κεντρικό κόμβο το 6 όπως φαίνεται και στην εικόνα . Επίσης εισέρχεται η bridge ακμή 2-6.



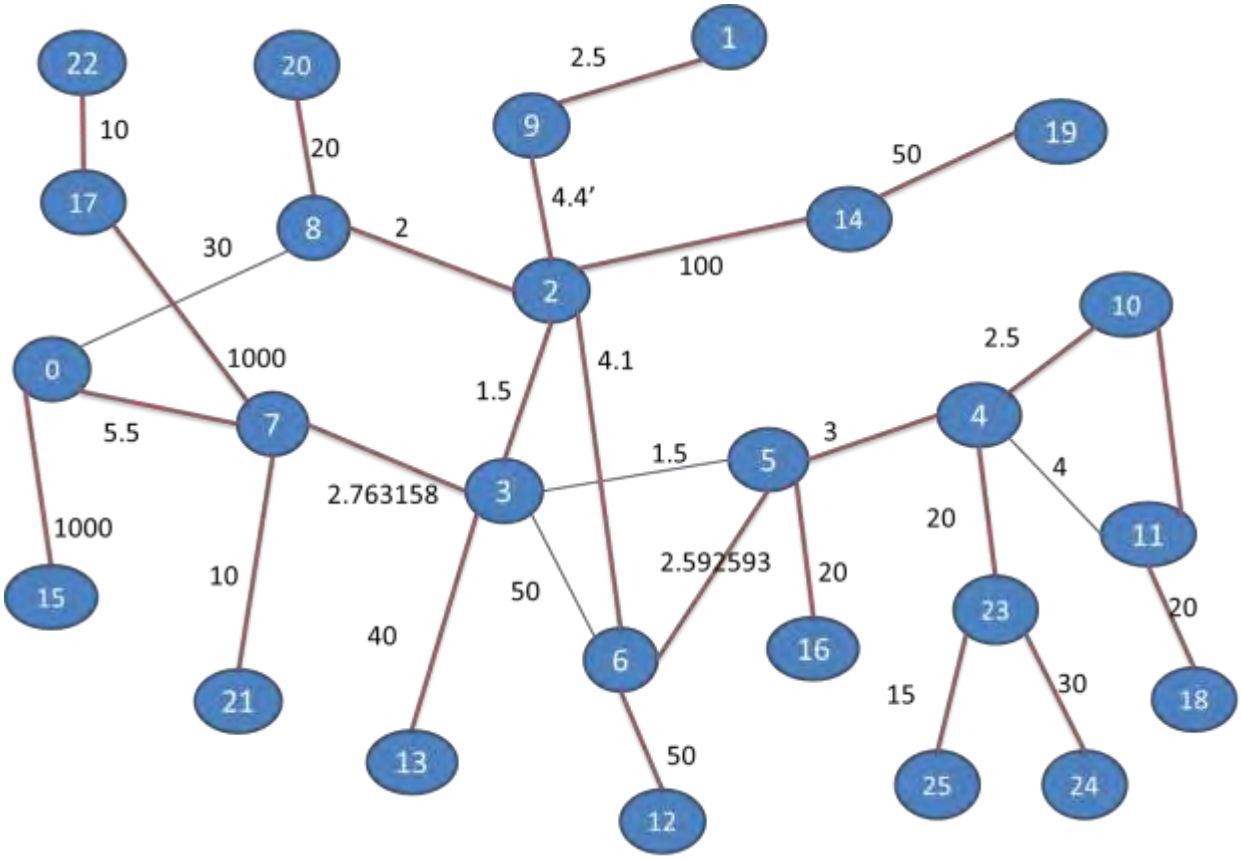
Εικόνα 6.8: Ball, cones και bridge ακμές μετά την 6η εφαρμογή της *star-decomposition*

Με την ίδια λογική συνεχίζεται η διαδικασία εφαρμογής *star decomposition* στο τρέχον ball και όταν το υπόδεντρο φτάσει να αποτελείται από έναν μόνο κόμβο συνεχίζεται η εφαρμογή στα cone του . Όταν τελειώσει η εφαρμογή και για τα cone του τότε γυρίζουμε στην προηγούμενη αναδρομή ώστε να εφαρμόσουμε τον αλγόριθμο σε cone στα οποία δεν έχει γίνει εφαρμογή.

Ουσιαστικά η αναδρομή τερματίζεται όταν όλοι οι κόμβοι έχουν χαρακτηριστεί ως κεντρικοί κόμβοι ενός ball.

Έτσι εφαρμόζεται η διαδικασία σε όλους τους κόμβους και τελικά έχουμε δημιουργήσει ένα spanning tree μέσα από τις bridge ακμές.

Μετά το τέλος των αναδρομικών *star decomposition* για αυτόν τον γράφο προκύπτει το spanning tree της εικόνας .



Εικόνα 6.9: Τελικό *low-stretch tree*

Βλέπουμε ότι σε αυτό το παράδειγμα προκύπτει το ίδιο spanning tree όπως και με την εφαρμογή του *maximum spanning tree*. Αυτό που διαφέρει είναι η διαδικασία εξαγωγής του.

Αφού λοιπόν υπολογιστεί *low stretch tree*, στον preconditioner πίνακα μπαίνουν μόνο οι ακμές που ανήκουν σε αυτό.

Τέλος προχωράμε στην λύση του συστήματος με την χρήση του preconditioner πίνακα, μέσω της τεχνικής που εξηγήσαμε σε προηγούμενο κεφάλαιο.

Κεφάλαιο 7

Αποτελέσματα και Συμπεράσματα

7.1 Αποτελέσματα Μετρήσεων

Ο κώδικας γράφηκε σε γλώσσα προγραμματισμού C και εκτελέστηκε με GNU Compiler Collection (GCC). Χρησιμοποιεί την GNU Scientific Library (GSL) και την βιβλιοθήκη Csparse. Τα αρχεία εκτελέστηκαν σε επεξεργαστή Intel Core 2 Quad σε 2,5 GHz και 8 GB κύρια μνήμη.

Σε αυτήν τη φάση θα συγκρίνουμε την απόδοση σε χρόνο σύγκλισης αλλά και αριθμό επαναλήψεων της CG, ανάμεσα στην εκτέλεση με τον συμβατικό *Jacobi* preconditioner και με τους νέους preconditioners *Maximum spanning tree* και *Low stretch tree.*

Εφαρμόζουμε τους preconditioners σε ένα netlist κυκλώματος μικρής κλίμακας με ιδιαίτερα θετικά αποτελέσματα. Αυτό μας δείχνει πόσο σημαντική μπορεί να είναι η μείωση χρόνου και επαναλήψεων στα γραμμικά κυκλώματα μεγάλης κλίμακας.

Για ένα netlist που αποτελείται από 158 αντιστάσεις, 139 κόμβους και 3 πηγές ρεύματος, θέτουμε κριτήριο τερματισμού σύγκλισης $||Ax - b|| \leq 1e-04 * ||b||$ και μετράμε τα μη μηδενικά στοιχεία του πίνακα πριν και μετά την εφαρμογή των preconditioner στην CG επίλυση. Τα μη μηδενικά στοιχεία του *Jacobi* preconditioner δεν τα υπολογίζουμε καθώς δεν θεωρείται πίνακας preconditioner αλλά διάνυσμα. Σαν όρισμα στον low stretch preconditioner δίνουμε αυθαίρετα τον κόμβο 3.

	Jacobi	*Maximum*	*Low-stretch*
Αρχικά μη μηδενικά στοιχεία	430	430	430
Τελικά μη μηδενικά στοιχεία	-	410	346
Αριθμός επαναλήψεων CG	103	8	25
Χρόνος σύγκλισης CG	0.03	0.01	0.01
Χρόνος κατασκευής preconditioner	0	0.01	0.02

Πίνακας 7.1: Αποτελέσματα επίδοσης με χρήση των precondittioner

Με την χρήση του *maximum spanning tree* preconditioner παρατηρούμε 5 % μείωση των μη μηδενικών στοιχείων του πίνακα αγωγιμοτήτων, 13x speedup στον αριθμό επαναλήψεων της CG καθώς και 3x speedup στον χρόνο σύγκλισης της μεθόδου. Όσον αφορά την εφαρμογή του low stretch preconditioner, παρατηρήθηκαν εξίσου αισιόδοξα αποτελέσματα με 20 % μείωση των μη μηδενικών στοιχείων του πίνακα αγωγιμοτήτων, 4x speedup στον αριθμό επαναλήψεων της CG αλλά και 3x speedup στον χρόνο σύγκλισης της μεθόδου.

Θα πρέπει επίσης να αναφερθεί , ότι αν σαν όρισμα στον *low stretch* preconditioner δώσουμε άλλον κόμβο, έστω τον κόμβο 10, παρατηρούμε ότι ο χρόνος κατασκευής του preconditioner αλλά και η απόδοση του αλλάζει.

Η διαφορά των αποτελεσμάτων της εφαρμογής του *low stretch* preconditioner με είσοδο τον κόμβο 10 από αυτή του κόμβου 3 φαίνεται στον πίνακα 7.2.

	Low-stretch (3)	*Low-stretch (10)*
Αρχικά μη μηδενικά στοιχεία	430	430
Τελικά μη μηδενικά στοιχεία	346	340
Αριθμός επαναλήψεων CG	25	27
Χρόνος σύγκλισης CG	0.01	0.01
Χρόνος κατασκευής preconditioner	0.02	0.05

Πίνακας 7.2: Αποτελέσματα επίδοσης *low-stretch* precondittioner με βάση την είσοδο

Αυτό αποδεικνύει ότι η επιλογή του κόμβου από τον οποίο θα ξεκινήσει η διαδικασία της star decomposition έχει σημασία καθώς επιδρά στον χρόνο ολοκλήρωσης αλλά και στο ποιες ακμές θα ελαττωθούν.

Σε γενικές γραμμές προτείνεται η εκκίνηση από κόμβο ο οποίος έχει μεγάλο αριθμό γειτόνων καθώς έτσι εξελίσσεται καλύτερα η star decomposition.

Δεν πρέπει επίσης να αγνοηθεί ο ρόλος που παίζει στα τελικά αποτελέσματα η ανοχή που έχουμε δώσει για την τελική σύγκλιση. Εάν παραδείγματος χάριν μειώσουμε την ανοχή στο $1e-06$ τότε παρατηρούμε αύξηση του αριθμού επαναλήψεων (βλέπε πίνακα 7.3) .

	Jacobi	*Maximum*	*Low-stretch*
Αρχικά μη μηδενικά στοιχεία	430	430	430
Τελικά μη μηδενικά στοιχεία	-	410	346
Αριθμός επαναλήψεων CG	111	10	30
Χρόνος σύγκλισης CG	0.03	0.01	0.01
Χρόνος κατασκευής preconditioner	0	0.01	0.02

Πίνακας 7.3: Αποτελέσματα επίδοσης με χρήση των precondittioner για νέο tolerance

7.2 Συμπεράσματα και Μελλοντικές επεκτάσεις

Είναι προφανές ότι με την χρήση των δύο preconditioner έχουμε πολύ καλύτερα αποτελέσματα, κυρίως στην μείωση αριθμού επαναλήψεων αλλά και χρόνου εκτέλεσης της CG.

Με την χρήση του maximum spanning tree preconditioner πετυχαίνουμε πολύ μεγάλο ποσοστό μείωσης του αριθμού των επαναλήψεων. Με την χρήση του low stretch tree preconditioner πετυχαίνουμε επίσης μεγάλο ποσοστό μείωσης του αριθμού των επαναλήψεων, που αν και μικρότερο αυτού του maximum spanning tree preconditioner, τελικά συγκλίνει στον ίδιο χρόνο με αυτόν αφού ο preconditioner πίνακάς του περιέχει λιγότερα μη μηδενικά στοιχεία.

Θα πρέπει εδώ να αναφέρουμε ότι χρόνος κατασκευής του low stretch tree preconditioner είναι ελαφρώς μεγαλύτερος από αυτόν του maximum spanning tree preconditioner λόγω της μεγαλύτερης πολυπλοκότητας της αναδρομικής διαδικασίας star-decomposition.

Μελλοντική επέκταση εργασίας χρησιμοποιώντας Ιεραρχικό *Support Graph*

Επειδή τα ρεαλιστικά δίκτυα παροχής ηλεκτρικής ενέργειας μπορεί να περιλαμβάνουν ένα μεγάλο αριθμό κόμβων, η κατασκευή ενός *support graph* για το σύνολο του δικτύου ισχύος/γείωσης μπορεί να είναι εξαιρετικά χρονοβόρα.

Σε αυτό το σημείο θα παρουσιάσουμε μια εναλλακτική διαδικασία, ως μελλοντική επέκταση της παρούσας εργασίας, η οποία μπορεί να εφαρμοστεί και να μειώσει ακόμα περισσότερο τον χρόνο εκτέλεσης.

Εδώ αντί να χρησιμοποιήσουμε το *low-stretch spanning tree* που παρουσιάσαμε με βάση το [17], χρησιμοποιούμε ένα *maximum spanning tree* για να το προσεγγίσουμε. Αν ορίσουμε μια λίστα εξετασμένων κόμβων, έστω V, ο αλγόριθμος δουλεύει ως εξής:

1. *Pick a random node*
2. *Add this node to V*
3. *choose the maximum weighted outgoing edge from V*
4. *add the new node in V*
5. *Repeat the above procedure until all nodes are inside V*

Σε υψηλό επίπεδο, ο γεωμετρικά ιεραρχικός *support-graph* preconditioner δημιουργείται ως εξής:

1. Χωρίζουμε όλο το πλέγμα κυκλώματος σε block ίσου μεγέθους που να περιέχουν τουλάχιστον ένα VDD pad όπως φαίνεται στην εικόνα όπου οι γραμμές πλέγματος αντιπροσωπεύουν το δίκτυο δίκτυο ηλεκτρικής ενέργειας και η χρωματιστή περιοχή αντιπροσωπεύει το χωρισμένο μπλοκ. Τα μαύρα τετράγωνα είναι τα αφηρημένα block τα οποία συνδέονται για να σχηματίσουν ένα block-level γράφο.

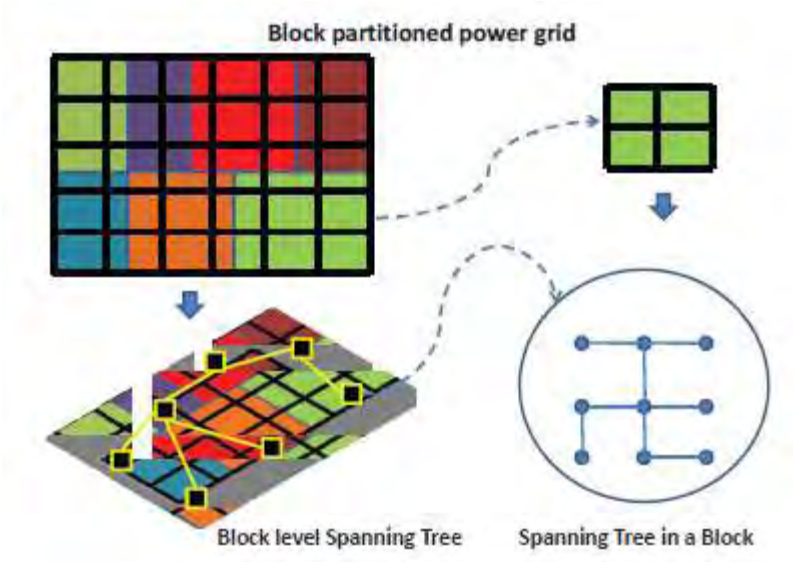
2. Μέσα σε κάθε block θα πρέπει να εξαχθεί από τον γράφο πλέγματος *ένα maximum spanning tree*, που ονομάζεται inner-block γράφο.

3. Ενώνουμε τα blocks μεταξύ τους με εικονικές ακμές, το βάρος των οποίων θα ισούται με το άθροισμα των βαρών όλων των ακμών που συνέδεαν τα αντίστοιχα blocks μεταξύ τους πριν το χώρισμά τους. Έτσι δημιουργείται ο block-level γράφος με εικονικές ακμές.

4. Υπολογίζουμε το *maximum spanning tree* του block-level γράφου.

5. Κάθε εικονική ακμή στο υπολογισμένο *maximun spanning tree* την αντικαθιστούμε με την μεγαλύτερου βάρους ακμή του αρχικού γράφου.

6. Τέλος, βρίσκουμε από τους inner-block γράφους και τον block-level γράφο το συνολικό *maximum spanning tree.*



Εικόνα 6.10: Κατασκευή ιεραρχικού *spanning tree*

Σε σύγκριση με τον άμεσο υπολογισμό του maximum spanning tree στον αρχικό πλέγμα κυκλώματος, αυτός ο ιεραρχικός υπολογισμός είναι πολύ πιο αποδοτικός, μιας και πραγματοποιείται με τρόπο διαίρει και βασίλευε.

Βιβλιογραφία

[1] H. Qian, S. R. Nassif, and S. S. Sapatnekar, “Power grid analysis using random walks,” *IEEE Trans. on Computer-Aided Design*, vol. 24, no.8, pp. 1204–1224, 2005.

[2] Lecture series on Dynamics of Physical Systems by Prof. Soumitro Banerjee, Department of Electrical Engineering, IIT Kharagpur.

[3] Xueqian Zhao, Jia Wang, Zhuo Feng, and Shiyan Hu, *"Power Grid Analysis with Hierarchical Support Graphs,"* Department of Electrical and Computer Engineering Michigan Technological University

[4] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, “A multigrid-like technique for power grid analysis,” *IEEE Trans. on Computer-Aided Design*, vol. 21, no. 10, pp. 1148–1160, 2002.

[5] V. Vineet, P. Harish, S. Patidar, and P. Narayanan, “Fast minimum spanning tree for large graphs on the GPU,” in *Proc. HPG*, New York, NY, USA, 2009, HPG ’09, pp. 167–171, ACM.

[6] T. H. Chen and C. C.-P Chen, “Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods,” in *Proc. IEEE/ACM DAC*, 2001, pp. 559–562.

[7] S. R. Nassif, *IBM power grid benchmarks*, [Online]. Available: http://dropzone.tamu.edu/ pli/PGBench/, 2008.

[8] E. Boman and B. Hendrickson, “Support theory for preconditioning,” *SIAM J. Matrix Anal. Appl.*, vol. 25, pp. 694–717, 2003.

[9] E. Boman, D. Chen, B. Hendrickson, and S. Toledo, “Maximum-weightbasis preconditioners,” *Numerical Linear Algebra and Applications*, vol. 11, pp. 695–721, 2004.

[10] Z. Feng and Z. Zeng, “Parallel multigrid preconditioning on graphics processing units (GPUs) for robust power grid analysis,” in *Proc. IEEE/ACM DAC*, 2010, pp. 661–666.

[11] E. Boman, B. Hendrickson, and S. Vavasis, “Solving elliptic finite element systems in near-linear time with support preconditioners,” *SIAM J. Numer. Anal.*, vol. 46, pp. 3264–3284, 2004.

[12] H. Qian and S. S. Sapatnekar, “A hybrid linear equation solver and its application in quadratic placement,” in *Proc. IEEE/ACM ICCAD*, 2005, pp. 905–909.

[13] N. Alon, R. M. Karp, D. Peleg, and D. West, "*A graph-theoretic game and its application to the k-server problem,"* SIAM J. Comput., 24 (1995), pp. 78–100.

[14] T. C. Hu, "*Optimum communication spanning trees,"* SIAM J. Comput., 3 (1974), pp. 188–195

.[15] E. Boman and B. Hendrickson, "*On Spanning Tree Preconditioners,"* manuscript, Sandia National Labs, Livermore, CA, 2001.

[16] G. Even, J. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms via spreading metrics*, J. ACM, 47 (2000), pp. 585–616.

[17] M. Elkin, Y. Emek, D. A. Spielman, and S. Teng, "*Lower-stretch spanning trees*," *SIAM J. Comput*, vol. 38, pp. 608–628, 2008.

[18] D. A. Spielman and S.-H. Teng, "*Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems,"* in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 81–90.

[19] J. Fakcharoenphol, S. Rao, and K. Talwar, "*A tight bound on approximating arbitrary metrics by tree metrics"*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 448–455.

[20] Y. Bartal, "*Probabilistic approximation of metric spaces and its algorithmic applications"*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 184–193.

[21] Y. Bartal, "*On approximating arbitrary metrices by tree metrics"*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 161–168.

[22] B. Awerbuch, "*Complexity of network synchronization"*, J. ACM, 32 (1985), pp. 804–823.

[23] David Peleg *"Low Stretch Spanning Trees," Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science*

[24] Harald Räcke "*Embedding into Sub-trees*," Theory of Metric Embeddings, lecture 6, Toyota Technological Institute at Chicago

[25] Ittai Abraham and Ofer Neiman, "*Using Petal-Decompositions to Build a Low Stretch Spanning Tree,"* Proceeding STOC '12 Proceedings of the forty-fourth annual ACM symposium on Theory of computing.