ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ενας Αλγόριθμος για το Σχεδιασμό Προσανατολισμένων Γράφων

Μαρία Καραβασίλη

Μεταπτυχιακή Εργασία

Ηράκλειο, Φεβρουάριος 1994

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ενας Αλγόριθμος για το Σχεδιασμό Προσανατολισμένων Γράφων

Εργασία που υποβλήθηκε από την
ΜΑΡΙΑ Π. ΚΑΡΑΒΑΣΙΛΗ
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Μαρία Καραβασίλη
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Μανώλης Γ. Η. Κατεβαίνης, Αναπληρωτής Καθηγητής, Επόπτης

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής, Μέλος

Γιώργος Γεωργακόπουλος, Επίκουρος Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 1994

Ενας Αλγόριθμος για το Σχεδιασμό Προσανατολισμένων Γράφων

Μαρία Καραβασίλη

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΛΗΨΗ

Η χρησιμοποίηση εικόνων και σχημάτων είναι τις περισσότερες φορές η πιο αποτελεσματική μέθοδος για τη μετάδοση αφηρημένης ή μη πληροφορίας, λόγω της έμφυτης ικανότητας του ανθρώπου να αντιλαμβάνεται ευκολότερα πληροφορία που του παρουσιάζεται γραφικά. Μία μορφή γραφικής αναπαράστασης πληροφορίας είναι ο γράφος.

Η παρούσα εργασία ασχολείται με την ανάπτυξη και την υλοποίηση ενός αλγορίθμου σχεδιασμού σε ιεραρχική μορφή προσανατολισμένων γράφων. Ο αλγόριθμος δέχεται ως είσοδο έναν κατάλογο με τις ακμές του γράφου και στην έξοδο υπολογίζει τις συντεταγμένες στις οποίες πρέπει να τοποθετηθεί κάθε κόμβος έτσι ώστε να δημιουργηθεί ένα αισθητικά αρεστό σχέδιο του γράφου. Ένας πολύ γνωστός αλγόριθμος σχεδιασμού προσανατολισμένων γράφων είναι ο αλγόριθμος των Sugiyama, Tagawa και Toda (αλγόριθμος STT), ο οποίος απασχόλησε αρκετά τους ερευνητές κατά τη διάρκεια της προηγούμενης δεκαετίας. Αρχικά υλοποιήσαμε τον αλγόριθμο αυτό, αλλά διαπιστώσαμε ότι είναι πολύ αργός για μεγάλους γράφους και επιπλεον συχνά δημιουργεί ανεπιθύμητες κάμψεις ακμών. Γι'αυτό αναπτύξαμε ένα νέο αλγόριθμο.

Ο νέος αλγόριθμος αρχικά επιλέγει ένα ζευγνύον δέντρο (spanning tree) του γράφου. Κατόπιν διασχίζοντας το ζευγνύον δέντρο αναδιατάσσει τους κόμβους του με στόχο την τελική ελλάτωση των τομών μεταξύ ακμών του γράφου. Στη συνέχεια χρησιμοποιεί έναν αλγόριθμο σχεδιασμού δέντρων ο οποίος υπολογίζει τις συντεταγμένες των κόμβων του ζευγνύοντος δέντρου.

Παρουσιάζουμε τον αλγόριθμο που σχεδιάσαμε και τον αξιολογούμε συγκρίνοντάς τον με τον αλγόριθμο STT. Ο αλγόριθμός για μικρούς και μέτριους γράφους πετυχαίνει χρόνους μικρότερους από 1 δευτερόλεπτο ενώ για σχετικά μεγάλους γράφους είναι μέχρι και δέκα φορές γρηγορότερος από τον αλγόριθμο STT. Επίσης ελαχιστοποιεί τον αριθμό των κάμψεων ακμών.

Επόπτης : Μανώλης Γ. Η. Κατεβαίνης, Αναπληρωτής Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

An Algorithm to Layout Directed Graphs

Maria Karavassilh

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

The use of images and diagrams is most of the times the most effective method for the transmission of abstract as well as concrete information, due to the inherent human ability to perceive more easily information displayed graphically. Graphs are a form of graphical representation of information.

This thesis deals with the design and implementation of a graph layout algorithm for hierarchically drawn directed graphs. The algorithm is given as input the list of the edges of the graph and produces in the output the coordinates where each node should be put in order to produce an aesthetically pleasing drawing of the graph. A well known graph layout algorithm for directed graphs is the algorithm which was developed by Sygiyama, Tagawa and Toda (STT algorithm), and was a major concern for scientists during the past decade. We initially implemented this algorithm, which is time inefficient when drawing large graphs and it often creates unwanted edge bends.

The new algorithm initially selects a spanning tree of the graph. Then, it traverses the spanning tree and rearranges its nodes in order to reduce the number of crossings between edges of the graph. In the end, it uses a tree layout algorithm which computes the coordinates of the nodes of the spanning tree.

We present the algorithm we designed and evaluate it by comparing it with the STT algorithm. Our algorithm achieves layout times under 1 second for small and medium sized graphs and it can be as much as ten times faster than the STT algorithm for large graphs. Furthermoreit minimizes the number of edge bends.

Supervisor : Manolis G. H. Katevenis, Associate Professor
Department of Computer Science
University of Crete

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μανώλη Κατεβαίνη που μου έδωσε την ευκαιρία να ασχοληθώ μ'αυτή την εργασία, αλλά και για τη καθοδήγησή του κατά τη διάρκειά της.

Ευχαριστώ τον δρ. Martin Dörr σε μία αρχική ιδέα του οποίου βασίστηκε η εκπόνηση αυτής της εργασίας.

Ευχαριστώ επίσης τα μέλη της επιτροπής εξέτασης της μεταπτυχιακής μου εργασίας καθηγητές κ.κ. Πάνο Κωνσταντόπουλο και Γιώργο Γεωργακόπουλο για τις χρήσιμες παρατηρήσεις τους οι οποίες συντέλεσαν στην τελική διαμόρφωση αυτού του κειμένου.

Δεν θα μπορούσα να μην ευχαριστήσω όλους τους συνεργάτες μου στο project ITHACA, για τη συνεργασία τους καθ'όλη τη διάρκεια αυτής της εργασίας αλλά και για τη καλή παρέα τους. Ιδιαίτερα ευχαριστώ τον Νίκο Τσατσάκη, την Αφροδίτη Μιχαηλίδου και τη Μάγδα Χατζάκη για την πολύτιμη βοήθειά τους.

Ακόμη, θα ήθελα να ευχαριστήσω το Γιώργο Γεωργιαννάκη ο οποίος με την συμπαράσταση και την υπομονή του έκανε πιο ευχάριστες τις δύσκολες στιγμές κατά τη διάρκεια των σπουδών μου, και φυσικά τους γονείς μου για την υποστήριξή τους.

Ιδιαίτερα ευχαριστώ το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υλικοτεχνική υποδομή που μου παρείχε κατά τη διάρκεια των προπτυχιακών και μεταπτυχιακών μου σπουδών.

Περιεχόμενα

Περίληψη **i**

Abstract **iii**

Ευχαριστίες **v**

Περιεχόμενα **viii**

Κατάλογος Πινάκων **ix**

Κατάλογος Σχημάτων **xii**

1 Εισαγωγή **1**

1.1 Βασική Ορολογία Γράφων και Αλγορίθμων Σχεδιασμού Γράφων 4

1.2 Η Οργάνωση της Εργασίας 7

1.3 Η Υλοποίηση του Αλγορίθμου 8

2 Προηγούμενες Ερευνητικές Εργασίες **9**

2.1 Δέντρα 9

2.2 Γενικοί Προσανατολισμένοι Γράφοι 14

3 Περιγραφή του Αλγορίθμου **23**

3.1 Εισαγωγή 23

3.2 Προεπεξεργασία του Γράφου 28

3.2.1 Χειρισμός των κύκλων 28

3.2.2 Τοποθέτηση κόμβων σε επίπεδα 28

3.2.3 Τοποθέτηση των ριζών στο μεγαλύτερο δυνατόν επίπεδο 29

3.2.4 Κατασκευή του ζευγνύοντος δέντρου 30

3.3 Συγκέντρωση Πληροφορίας κατά την Πρώτη Φάση 32

3.3.1 Ακμές γράφου και μονοπάτια κόμβων 32

3.3.2 Ομάδες συσχετιζόμενων κόμβων 35
3.4 Αναδιάταξη των Κόμβων κατά τη Δεύτερη Φάση 38
3.4.1 Η κατεύθυνση ενός κόμβου 39
3.4.2 Αναδιάταξη κόμβων με βάση την κατεύθυνσή τους 42
3.4.3 Διάταξη των κόμβων κάθε ομάδας 43
3.4.4 Διάταξη των Κύριων κόμβων 44
3.4.5 Διάταξη των Δευτερεύοντων κόμβων 48
3.5 Τελική Επεξεργασία και Εισαγωγή των Τεχνητών Κόμβων 56
3.6 Συνοπτική παρουσίαση των βημάτων του αλγορίθμου 58
3.7 Ένα Παράδειγμα Δημιουργίας ενός Σχεδίου από τον Αλγόριθμο 59
3.7.1 Προεπεξεργασία. 60
3.7.2 Πρώτη Φάση. 61
3.7.3 Δεύτερη Φάση. 62
3.7.4 Εισαγωγή του πρώτου τεχνητού κόμβου σε μακριές ακμές γράφου. . . . 67
3.7.5 Αναδιάταξη του δέντρου και σχεδιασμός 68

4 Αξιολόγηση του αλγορίθμου **71**
4.1 Το σύνολο των παραδειγμάτων 72
4.2 Τα κριτήρια σύγκρισης 73
4.3 Συγκριτική παρουσίαση των αλγορίθμων *ReArrange* και *STT* 74

5 Συμπεράσματα και Μελλοντική Εργασία **81**
5.1 Συμπεράσματα 81
5.2 Βελτιώσεις και Επεκτάσεις 82

A Ο αλγόριθμος για σχεδιασμό δέντρων του S. Moen **85**

B Οι γράφοι του συνόλου μετρήσεων **89**

Βιβλιογραφία **110**

Κατάλογος Πινάκων

3.1 Πληροφορία που συγκεντρώνεται στη πρώτη φάση 61
3.2 Οι ομάδες κόμβων που δημιουργούνται κατά τη πρώτη φάση 61

4.1 Το σύνολο των γράφων που χρησιμοποιήθηκαν στις συγκριτικές μετρήσεις . . 72
4.2 Χρόνοι Εκτέλεσης (σε δευτερόλεπτα). 75
4.3 Αριθμός Τομών Ακμών 76
4.4 Συνολικό Μήκος Ακμών (σε cm). 77
4.5 Αριθμός Κάμψεων Ακμών 78
4.6 Μέγιστο Εμβαδό (σε cm^2). 79

Κατάλογος Σχημάτων

1.1 Ενα παράδειγμα προσανατολισμένου γράφου 4
1.2 Ενα παράδειγμα σχεδίου γράφου σε ιεραρχική μορφή 6

2.1 Ενα παράδειγμα σχεδίου δέντρου 10
2.2 Σχέδιο δέντρου με τον αλγόριθμο του Knuth 11
2.3 Ο αλγόριθμος των Reingold και Tilford για σχεδιασμό δέντρων 12
2.4 Υπολογισμός του βαρυκέντρου 16
2.5 Ο αλγόριθμος του Carpano για ιεραρχίες μήκους k. 17

3.1 Σχέδιο ενός προσανατολισμένου γράφου πριν την αναδιάταξη των κόμβων . . 25
3.2 Ο γράφος του σχήματος 3.1 μετά την αναδιάταξη των κόμβων 26
3.3 Κλωστές μεταξύ χαρακτηριστικών μονοπατιών ακμών 33
3.4 Κατασκευή των ομάδων συσχετιζόμενων κόμβων 37
3.5 Δύο διαφορετικές ομάδες συσχετιζόμενων κόμβων, παιδιών του A 38
3.6 Δύο δυνατές τοποθετήσεις του ίδιου κόμβου μέσα στη λίστα των αδελφιών του. 40
3.7 Δύο δυνατές τοποθετήσεις των Κύριων κόμβων μιας ομάδας συσχετιζόμενων κόμβων 44
3.8 Τομές ακμών δημιουργούμενες κατά την τοποθέτηση των Κύριων κόμβων . . . 47
3.9 Φωλιασμένη τοποθέτηση Δευτερεύοντων κόμβων γύρω από τον Κύριο κόμβο με τον οποίο σχετίζονται 53
3.10 Δημιουργούμενες τομές ακμών κατά την τοποθέτηση του Δευτερεύοντα κόμβου Z - Πρώτη περίπτωση. 54
3.11 Δημιουργούμενες τομές ακμών κατά την τοποθέτηση του Δευτερεύοντα κόμβου Z - Δεύτερη περίπτωση 55
3.12 Τα βήματα του αλγορίθμου 59
3.13 Ενα παράδειγμα γράφου 60
3.14 Ενα ενδιάμεσο στάδιο στην κατασκευή του σχεδίου του G 68
3.15 Το τελικό σχέδιο του G 69

A.1 (a) Ενα φύλλο και το περιθώριό του, (b) το πολύγωνο ενός φύλλου 87
A.2 Επέκταση του πολυγώνου ενός πατέρα ώστε να συμπεριλάβει τα πολύγωνα των παιδιών του 88

B.1 Σχέδιο του ReArrange για το γράφο mod_call 90
B.2 Σχέδιο του STT για το γράφο mod_call 91
B.3 Σχέδιο του ReArrange για το γράφο shortlist 92
B.4 Σχέδιο του STT για το γράφο shortlist 92
B.5 Σχέδιο του ReArrange για το γράφο world2 93
B.6 Σχέδιο του STT για το γράφο world2 93
B.7 Σχέδιο του ReArrange για το γράφο world1 94
B.8 Σχέδιο του STT για το γράφο world1 95
B.9 Σχέδιο του ReArrange για το γράφο SelfPortrait 96
B.10 Σχέδιο του STT για το γράφο SelfPortrait 97
B.11 Σχέδιο του ReArrange για το γράφο pert 98
B.12 Σχέδιο του STT για το γράφο pert 99
B.13 Σχέδιο του ReArrange για το γράφο p104 100
B.14 Σχέδιο του STT για το γράφο p104 101
B.15 Σχέδιο του ReArrange για το γράφο t_hierarchy 102
B.16 Σχέδιο του STT για το γράφο t_hierarchy 103
B.17 Σχέδιο του ReArrange για το γράφο main1 104
B.18 Σχέδιο του STT για το γράφο main1 105
B.19 Σχέδιο του ReArrange για το γράφο call_graph1 106
B.20 Σχέδιο του STT για το γράφο call_graph1 107
B.21 Σχέδιο του ReArrange για το γράφο main2 108
B.22 Σχέδιο του STT για το γράφο main2 109

Κεφάλαιο 1

Εισαγωγή

Η χρησιμοποίηση εικόνων και σχημάτων είναι τις περισσότερες φορές η πιο αποτελεσματική μέθοδος για τη μετάδοση αφηρημένης ή μη πληροφορίας. Αυτό οφείλεται στην έμφυτη ικανότητα των ανθρώπων να αντιλαμβάνονται καλύτερα και να εξάγουν ευκολότερα συμπεράσματα από πληροφορία που τους παρουσιάζεται γραφικά. Ετσι είναι προτιμότερη η γραφική αναπαράσταση πληροφορίας σε σχέση με την αντίστοιχη αυστηρή συντακτική περιγραφή της ίδιας πληροφορίας σε μορφή κειμένου [26]. Μια μορφή γραφικής αναπαράστασης είναι ο *γράφος* ή *γράφημα*. Εδώ χρησιμοποιούμε τον όρο γράφος με την έννοια που αυτός έχει στη θεωρία των μαθηματικών. Ετσι, ένας γράφος G αποτελείται από ένα πεπερασμένο, μη κενό σύνολο σημείων V = V(G) και ένα σύνολο E απο ζεύγη στοιχείων του V ($E \subseteq V \times V$) [17]. Τα στοιχεία του συνόλου V συνήθως ονομάζονται *κόμβοι* ενώ τα στοιχεία του συνόλου E *ακμές*. Στην παράγραφο 1.1 δίνονται μερικοί ορισμοί από τη Θεωρία Γραφημάτων οι οποίοι θα χρησιμοποιηθούν στη συνέχεια.

Οι προσανατολισμένοι γράφοι είναι ένα εργαλείο για αναπαράσταση εννοιών και σχέσεων που χρησιμοποιείται ευρύτατα σε διάφορους τομείς μεταξύ των οποίων και στην Επιστήμη Υπολογιστών. Ενδεικτικά μπορούμε να αναφέρουμε, στους Υπολογιστές, τα διαγράμματα οντοτήτων-συσχετίσεων, τα δίκτυα Petri, τα συντακτικά δέντρα, τις μηχανές πεπερασμένων καταστάσεων, τα δίκτυα PERT και CPM, τα διαγράμματα ροής δεδομένων, τα διαγράμματα κλήσεων υπορουτινών, και τα πιο γενικά από όλα τα σημασιολογικά δίκτυα.

Υπάρχουν διάφορα συστήματα που επιτρέπουν το σχεδιασμό και την επεξεργασία γράφων τα οποία μπορούμε να τα εντάξουμε σε δύο κατηγορίες με βάση το ποσοστό επέμβασης του ανθρώπου στην όλη διαδικασία της επεξεργασίας ή αντίστοιχα το βαθμό αυτοματοποίησης της διαδικασίας. Στην πρώτη κατηγορία διακρίνουμε διαλογικά συστήματα τα οποία επιτρέπουν στους χρήστες να σχεδιάζουν και να τροποποιούν γράφους ειδικής μορφής όπως τα *διαγράμματα PERT* ή τα *διαγράμματα οντοτήτων-συσχετίσεων*. Ενα τέτοιο σύστημα είναι

το Mac - Project [1] το οποίο επιτρέπει στους χρήστες να σχεδιάζουν διαγράμματα PERT καθορίζοντας οι ίδιοι το μέγεθος και τη θέση κάθε κόμβου καθώς και τις συνδέσεις μεταξύ κόμβων και τη δρομολόγηση των ακμών. Τέτοια συστήματα έχουν προφανή πλεονεκτήματα έναντι της μεθόδου σχεδίασης με χαρτί και μολύβι, αλλά και μειονεκτήματα, αφού ο σχεδιασμός γίνεται δυσκολότερος όσο μεγαλώνει το μέγεθος του γράφου και δεν υπάρχει η δυνατότητα να σχεδιαστεί ένας γράφος του οποίου η περιγραφή παράγεται αυτόματα από μία εφαρμογή. Τα συστήματα που ανήκουν στη δεύτερη κατηγορία δεν έχουν τα μειονεκτήματα της πρώτης, γιατί περιέχουν *αλγόριθμους σχεδίασης γράφων* οι οποίοι αυτοματοποιούν τη διαδικασία του σχεδιασμού. Απ'την άλλη μεριά, αυτά τα συστήματα παράγουν σχέδια τα οποία συχνά είναι χειρότερα από αυτά που δημιουργούνται από ανθρώπους, αν τα κρίνουμε σύμφωνα με διάφορους αισθητικούς κανόνες. Είναι όμως μία ικανοποιητική εναλλακτική λύση στη περίπτωση που ο διαθέσιμος χρόνος για τη παραγωγή ενός σχεδίου είναι λίγος. Γενικά αυτά τα συστήματα χρησιμοποιούν μια περιγραφή του γράφου (π.χ. μια λίστα των ακμών του γράφου) και παρέχουν στο χρήστη γραφικά μέσα για να διερευνήσει και/ή να αλλάξει το σχεδιασμό που παράχτηκε με αυτόματο τρόπο. Ενα τέτοιο σύστημα είναι το DAG [16].

Ενας αλγόριθμος σχεδίασης γράφων δέχεται ως είσοδο μια τυπική περιγραφή του συνόλου των κόμβων και των ακμών του γράφου, και υπολογίζει τις συντεταγμένες στις οποίες πρέπει να τοποθετηθούν οι κόμβοι (ίσως και οι ακμές αν αυτές δεν σχεδιάζονται ευθείες). Το σχέδιο που θα προκύψει αν τοποθετήσουμε τους κόμβους στις υπολογισμένες συντεταγμένες πρέπει να μπορεί να ''διαβαστεί'' εύκολα και να είναι αισθητικά αρεστό στο χρήστη. Γενικά τα διάφορα κριτήρια που χρησιμοποιούν οι χρήστες για να κρίνουν την ποιότητα του σχεδίου ενός γράφου είναι υποκειμενικά και δεν μπορούν να περιγραφούν με τυπικό τρόπο, γι'αυτό δεν είναι εύκολη η σχεδίαση αλγορίθμων με βάση τέτοιου είδους κριτήρια. Υπάρχουν όμως και αντικειμενικά κριτήρια τα οποία όλοι οι αλγόριθμοι σχεδιασμού γράφων θα πρέπει να τα παίρνουν υπόψη τους. Τέτοια κριτήρια είναι ο μικρός χρόνος εκτέλεσης του αλγορίθμου, ο ελάχιστος αριθμός ακμών που τέμνονται, η σχεδίαση των ακμών όσο το δυνατόν λιγότερο τεθλασμένες (δηλαδή με όσο το δυνατόν λιγότερες κάμψεις), η ομοιόμορφη κατανομή των κόμβων στο επίπεδο κ.α. Μπορούμε να θεωρήσουμε αυτά τα κριτήρια σαν επιθυμητούς στόχους τους οποίους κάθε αλγόριθμος σχεδιασμού γράφων θα πρέπει να επιδιώκει. Αυτοί οι περιορισμοί διακρίνονται σε *συντακτικούς* (αυτοί που σχετίζονται με τη δομή του γράφου) και *εννοιολογικούς* (αυτοί που επιβάλλονται από την εφαρμογή ή το χρήστη). Τα διάφορα είδη περιορισμών περιγράφονται στα [21], [31].

Το πρόβλημα της γρήγορης σχεδίασης γράφων απασχόλησε τον Robins [26] ο οποίος σχεδίασε έναν αλγόριθμο με γραμμική πλοκή. Ενώ ο αλγόριθμός του πετυχαίνει πολύ

μικρούς χρόνους σχεδίασης, αποτυχαίνει ως προς τη ποιότητα των παραγόμενων σχεδίων. Σ'αυτή την εργασία περιγράφεται ένας νέος αλγόριθμος σχεδιασμού προσανατολισμένων γράφων ο οποίος προσπαθεί να πετύχει μικρούς χρόνους σχεδίασης. Σε γενικές γραμμές η δομή του αλγορίθμου είναι η εξής. Αρχικά κατασκευάζει ένα δέντρο το οποίο περιέχει όλους τους κόμβους του γράφου (spanning tree). Κατόπιν κάνει μία inorder διάσχιση του δέντρου και σε κάθε κόμβο αναδιατάσσει τα παιδιά του παίρνοντας υπόψη ήδη υπολογισμένη πληροφορία και έχοντας ως στόχο τη βελτίωση του τελικά παραγόμενου σχεδίου. Η τελική σχεδίαση του γράφου βασίζεται στη σχεδίαση του αναδιαταγμένου δέντρου.

Η εργασία αυτή αναπτύχθηκε στα πλαίσια του ερευνητικού έργου ITHACA[1] [6] [7]. Σκοπός του έργου είναι η δημιουργία ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης εφαρμογών με οντοκεντρικές τεχνικές (object-oriented techniques). Το σύστημα περιλαμβάνει μια οντοκεντρική γλώσσα προγραμματισμού και βάση δεδομένων, μία βάση λογισμικού, και εργαλεία ανάπτυξης εφαρμογών και υποστήριξης τους. Ο αλγόριθμος χρησιμοποιείται για το σχεδιασμό προσανατολισμένων γράφων στο γραφικό μέρος της επαφής χρήσεως (user-interface) του συστήματος.

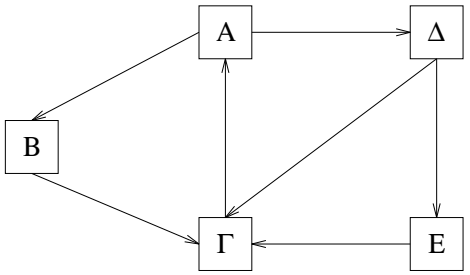
Αρχικά υλοποιήσαμε τον αλγόριθμο σχεδιασμού γράφων των Sugiyama, Tagawa και Toda [29]. Αργότερα βελτιώσαμε την απόδοση του αλγορίθμου χρησιμοποιώντας τις προτάσεις που περιγράφονται στα [27] και [21]. Παρά τις βελτιώσεις υπήρχαν περιπτώσεις όπου η ποιότητα των παραγόμενων σχεδίων δεν ήταν καθόλου ικανοποιητική (πολλές αδικαιολόγητες κάμψεις σε ακμές - edge bends, οι οποίες μπορούσαν να σχεδιαστούν ευθείες ή περιέχοντας μεγάλα ευθύγραμμα τμήματα) και επιπλέον ο χρόνος εκτέλεσης για μέτριους έως μεγάλους γράφους (με 100 - 250 κόμβους) αν και όχι πολύ μεγάλος μπορούσε να βελτιωθεί περισσότερο. Οι απαιτήσεις σε χρόνο υπαγορεύονται από το γεγονός ότι ο αλγόριθμος αυτός αποτελεί τμήμα του γραφικού διερευνητή (graph browser) της επαφής χρήσεως ενός πληροφοριακού συστήματος (information system), όπου υπάρχει απαίτηση για πολύ μικρό χρόνο απόκρισης στις ερωτήσεις που διατυπώνει ο χρήστης του συστήματος. Ετσι προσπαθήσαμε να σχεδιάσουμε ένα αλγόριθμο του οποίου ο χρόνος εκτέλεσης νά πληρεί τους περιορισμούς ενός γραφικού διερευνητή ακόμα και για σχετικά μεγάλους γράφους. Ταυτόχρονα ο νέος αλγόριθμος θα πρέπει να έχει συγκρίσιμα ή και καλύτερα αποτελέσματα από τον αλγόριθμο των Sugiyama, Tagawa και Toda, όσον αφορά τα άλλα κριτήρια αξιολόγησης των αλγορίθμων σχεδιασμού γράφων.

[1] Integrated tool for highly advanced computer applications

1.1 Βασική Ορολογία Γράφων και Αλγορίθμων Σχεδιασμού Γράφων

Αυτή η παράγραφος περιγράφει τη βασική ορολογία της Θεωρίας Γραφημάτων καθώς και ορολογία που χρησιμοποιείται σε αλγορίθμους σχεδιασμού γράφων. Πολλοί όροι ίσως είναι ήδη γνωστοί στον αναγνώστη αλλά δίνονται εδώ για λόγους πληρότητας.

Έστω ένα μή κενό σύνολο V = $\{v_1, \ldots, v_n\}$, και E ένα σύνολο μή διατεταγμένων ζευγών (u, v) διακριτών στοιχείων του V. Έτσι, το σύνολο E ορίζει μια διμελή σχέση πάνω στα στοιχεία του V. Τότε το ζεύγος (V, E) ονομάζεται *μη προσανατολισμένο γράφημα* ή απλά *γράφος*. Τα στοιχεία του V ονομάζονται *κορυφές* ή *κόμβοι* ή *σημεία* και τα στοιχεία του E *ακμές* ή *τόξα* (arcs) ή *δεσμοί* (links). Στα πλαίσια αυτής της εργασίας χρησιμοποιούμε τους όρους κόμβοι και ακμές για τα στοιχεία του V και E αντίστοιχα. Αν τα στοιχεία του E είναι διατεταγμένα ζεύγη, τότε το ζεύγος (V, E) ονομάζεται *προσανατολισμένο γράφημα*. Έτσι τα ζεύγη (v, w) και (w, v) είναι διαφορετικές ακμές ενός προσανατολισμένου γράφου με κόμβους τους v και w. Για τη γεωμετρική αναπαράσταση ενός γραφήματος G = (V, E), κάθε κόμβος του V παριστάνεται στο επίπεδο με μία τελεία, ένα μικρό κύκλο ή ένα μικρό τετράγωνο και η κορυφή u συνδέεται με τη κορυφή v μέσω μιας συνεχόμενης ευθείας αν και μόνον αν (u, v) $\in$ E. Στο σχήμα 1.1 δίνεται ο προσανατολισμένος γράφος G = (V, E) όπου



Σχήμα 1.1: Ενα παράδειγμα προσανατολισμένου γράφου

V = {Α, Β, Γ, Δ, Ε} και E = {(Α,Β), (Β,Γ), (Γ,Α), (Α,Δ), (Δ,Γ), (Δ,Ε), (Ε,Γ)}.

Υπογράφημα G(S) ενός γραφήματος G = (V, E) είναι ένα γράφημα με σύνολο κόμβων S $\in$ V και σύνολο ακμών E(S) $\in$ E οι οποίες συνδέουν κόμβους του S και μόνον αυτές.

Δοθείσας μιας ακμής e = (v,w), λέμε ότι η e είναι *προσκείμενη* στους κόμβους v και w και αντίστροφα, ο v είναι προηγούμενος (predecessor) του w, και ο w είναι επόμενος (successor) του v. Επίσης ο v είναι *αφετηρία* της e και ο w *προορισμός* της. Ο *έσω-βαθμός* ενός κόμβου v $\in$ V ενός προσανατολισμένου γράφου G = (V, E) είναι ίσος με τον αριθμό των ακμών (u, v). Αντίστοιχα ο *έξω-βαθμός* ενός κόμβου v είναι ίσος με τον αριθμό των ακμών (v, u). *Πυκνότητα* του γράφου ονομάζουμε το ποσοστό $\frac{100 \times Card(E)}{Card(V) \times Card(V)}$, όπου Card(E) είναι ο πληθικός αριθμός

του E και Card(V) είναι ο πληθικός αριθμός του V. Η πυκνότητα εκφράζει τι ποσοστό του μέγιστου αριθμού είναι ο αριθμός των ακμών του γράφου.

Μοναπάτι (path) σε ένα γράφο είναι μια ακολουθία από κόμβους $v_1, v_2, \ldots, v_k \in V$ που συνδέονται μεταξύ τους και $(v_i, v_{i+1}) \in E$ για 1≤i < k. Δοθέντος ενός προσανατολισμένου μονοπατιού $v_1, \ldots, v_j, \ldots, v_n$, ο κόμβος v_j ονομάζεται *πρόγονος* (ancestor) όλων των κόμβων που είναι επόμενοι του στο μονοπάτι (δηλ. των κόμβων $v_{j+1}, \ldots, v_n$). Ομοια ο v_j είναι *απόγονος* (descendant) όλων των προηγούμενων κόμβων από αυτόν στο μονοπάτι (δηλ. των κόμβων $v_1, \ldots, v_{j-1}$). Για παράδειγμα στο σχήμα 1.1 ο κόμβος A είναι πρόγονος του κόμβου E και ο E είναι απόγονος του κόμβου A στο μονοπάτι B, Γ, A, Δ, E. *Κύκλος* είναι κάθε μονοπάτι στο οποίο $v_1 = v_k$. Η εκφυλισμένη περίπτωση όπου μια ακμή συνδέει ένα κόμβο με τον εαυτό του λέγεται *βρόχος*. Αν ένας γράφος περιέχει κύκλους ονομάζεται *κυκλικός*, ενώ αν δεν περιέχει *άκυκλος*. Ο γράφος του σχήματος 1.1 περιέχει 3 κύκλους, συγκεκριμένα τους (A, B, Γ, A), (A, Δ, Γ, A) και (A, Δ, E, Γ, A). Κάθέ κυκλικός προσανατολισμένος γράφος μπορεί να μετατραπεί σε έναν άκυκλο προσανατολισμένο γράφο ή DAG (Directed Acyclic Graph) αν αντιστρέψουμε τη φορά κάποιων από τις ακμές του. Ένας γράφος ονομάζεται *συνεκτικός* ή *συνδεδεμένος* αν για κάθε δύο κόμβους u και v όπου u ≠ v, υπάρχει τουλάχιστον ένα μονοπάτι από τον u στον v, ή από τον v στο u, ή και τα δύο.

Ελεύθερο δέντρο (free tree) T είναι ένα συνεκτικό και άκυκλο μή προσανατολισμένο γράφημα. Ισοδύναμα ένα ελεύθερο δέντρο n κόμβων έχει n-1 ακμές και κάθε δύο κόμβοι συνδέονται μεταξύ τους με ένα απλό μονοπάτι. Ένα *ριζωμένο* ή προσανατολισμένο δέντρο (rooted or directed tree) είναι ένα ελεύθερο δέντρο για το οποίο επιλέγεται ένας κόμβος που ονομάζεται ρίζα (root), και εισάγεται προσανατολισμός στις ακμές.

Για τα ριζωμένα δέντρα χρησιμοποιείται η πιο κάτω ορολογία: Κάθε κόμβος v έχει ακριβώς έναν προηγούμενο κόμβο Π, ο οποίος ονομάζεται *πατέρας* (parent) του κόμβου και ο v ονομάζεται *παιδί* (child) του κόμβου Π. Τα παιδιά ενός κόμβου είναι μεταξύ τους *αδέρφια* (siblings). Ορισμένες φορές είναι χρήσιμο να θεωρούμε ότι τα αδέρφια έχουν μεταξύ τους μια συγκεκριμένη διάταξη, επομένως η λίστα των αδερφιών είναι ταξινομημένη. Ο κόμβος που βρίσκεται στο επίπεδο 1 ονομάζεται *ρίζα* του δέντρου. Ενας κόμβος δέντρου ονομάζεται *φύλλο* όταν δεν έχει διαδόχους.

Σε ένα δέντρο T αν θεωρήσουμε ένα κόμβο του v, το υπογράφημα T(v) το οποίο αποτελείται από τον κόμβο v και όλους τους απογόνους του, ονομάζεται *υποδέντρου* του T. Ο v θεωρείται ρίζα του T(v).

Για τις ανάγκες αυτής της εργασίας ορίζουμε την έννοια του *συνόρου* ενός υποδέντρου, το οποίο είναι ένα μονοπάτι κόμβων με αρχή τη ρίζα ενός υποδέντρου οι οποίοι έχουν μια συγκεκριμένη ιδιότητα. Αν θεωρήσουμε ένα προσανατολισμένο δέντρο με φορά ακμών από

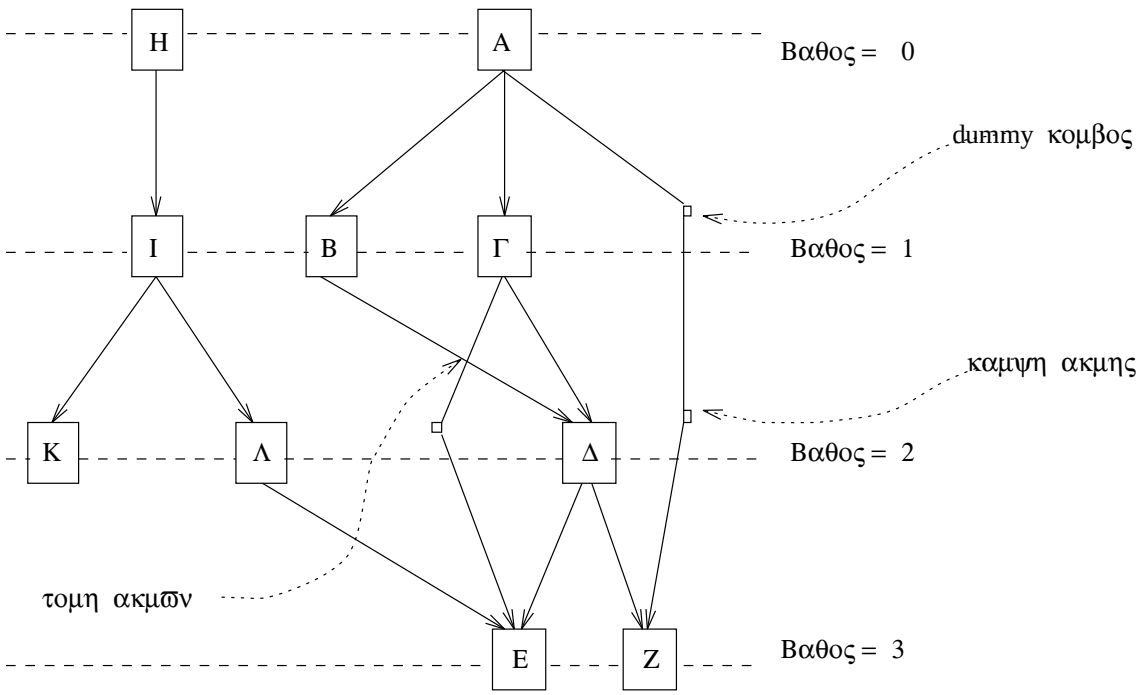
πάνω προς τη κάτω, τότε κάθε υποδέντρο του έχει ένα *δεξί-σύνορο* και ένα *αριστερό σύνορο*. Οι ορισμοί των παραπάνω μονοπατιών δίνονται αναδρομικά. Έστω ένα υποδέντρο T_1 του T με ρίζα τον κόμβο v. Το δεξί σύνορο του T_1 είναι το μονοπάτι $v_1, \ldots, v_k$ όπου:

$$v_{i+1} = \begin{cases} v & \text{εάν } i = 0 \\ \text{ο πρώτος κόμβος στη λίστα παιδιών του } v_i & \text{εάν } i < k \end{cases}$$

Το αριστερό σύνορο του T_1 είναι το μονοπάτι $v_1, \ldots, v_l$ όπου:

$$v_{i+1} = \begin{cases} v & \text{εάν } i = 0 \\ \text{ο τελευταίος κόμβος στη λίστα παιδιών του } v_i & \text{εάν } i < l \end{cases}$$

Οι ορισμοί που δίνονται στη συνέχεια είναι σχετικοί με αλγορίθμους σχεδιασμού γράφων. Συνήθως όταν σχεδιάζουμε έναν άκυκλο προσανατολισμένο γράφο τον σχεδιάζουμε σαν μια *ιεραρχία* (hierarchy) ή *διαστρωματομένο γράφο* (layered graph). Το σχέδιο ενός άκυκλου προσανατολισμένου γράφου είναι μια ιεραρχία όταν η κατεύθυνση όλων των ακμών είναι η ίδια (π.χ. από πάνω προς τα κάτω ή από αριστερά προς τα δεξιά). Ετσι ο γράφος του σχήματος 1.1 δεν είναι μια ιεραρχία, ενώ ο γράφος του σχήματος 1.2 είναι μια ιεραρχία όπου οι ακμές έχουν κατεύθυνση από πάνω προς τα κάτω. Το πλεονέκτημα ενός σχεδίου



Σχήμα 1.2: Ενα παράδειγμα σχεδίου γράφου σε ιεραρχική μορφή

γράφου σε ιεραρχική μορφή είναι ότι είναι πολύ εύκολο να διακρίνει κανείς βλέποντας το τις σχέσεις πρόγονος/απόγονος ακριβώς επειδή οι πρόγονοι ενός κόμβου v βρίσκονται στο σχέδιο πάνω από τον v (ή αν πρόκειται για σχέδιο με κατεύθυνση ακμών από αριστερά προς τα δεξιά, αριστερότερα του v). Ενας απλός τρόπος να τοποθετήσουμε τους κόμβους ενός άκυκλου προσανατολισμένου γράφου ώστε να δημιουργήσουμε μια ιεραρχία είναι να χρησιμοποιήσουμε το *βάθος* των κόμβων. Εάν Π_v είναι το σύνολο των προκατόχων του κόμβου v, τότε το βάθος του δίνεται από τον αναδρομικό ορισμό:

$$\text{βάθος}(v) = \begin{cases} 0 & \text{εάν } \Pi_v = \emptyset \\ 1 + \max\limits_{w \in \Pi_v} \{\text{βάθος}(w)\} & \text{εάν } \Pi_v \neq \emptyset \end{cases}$$

Ετσι ένας άκυκλος προσανατολισμένος γράφος, όπου το μεγαλύτερο βάθος κόμβου είναι B, μπορεί να μετατραπεί σε μια ιεραρχία με B *επίπεδα*, όπου στο επίπεδο i τοποθετούμε όλους τους κόμβους του γράφου με βάθος i. Η ιεραρχία στο σχήμα 1.2 έχει 4 επίπεδα. Οι κόμβοι του γράφου οι οποίοι δεν έχουν προκάτοχο και άρα έχουν βάθος 0 ονομάζονται *ρίζες*.

Μήκος μιας ακμής ονομάζουμε την αριθμητική διαφορά του επιπέδου της αφετηρίας της και του επιπέδου του προορισμού της. Μια ακμή με μήκος μεγαλύτερο από 1 ονομάζεται *μακριά ακμή* (long edge). Στο σχήμα 1.2 βλέπουμε δύο μακριές ακμές, συγκεκριμένα τις (A,Z) με μήκος 3 και την (Γ,E) με μήκος 2. Μια *κανονική ιεραρχία* είναι μια ιεραρχία όπου κάθε ακμή έχει μήκος το πολύ 1. Για να μετατρέψουμε μια ιεραρχία σε κανονική, εισάγουμε στο γράφο επιπλέον κόμβους τους οποίους ονομάζουμε τεχνητούς (dummy) κόμβους. Σε κάθε μακριά ακμή με μήκος M εισάγουμε M-1 τεχνητούς κόμβους έτσι ώστε να δημιουργήσουμε μία ακολουθία από ακμές με μήκος 1 οι οποίες αντικαθιστούν την αρχική μακριά ακμή. Στο σχήμα 1.2 έχουμε εισάγει τρείς τεχνητούς κόμβους. Σε κάθε τεχνητό κόμβο που εισάγουμε μπορούμε να δημιουργήσουμε μία *κάμψη* (bend) στην αρχική μακριά ακμή. Γενικά μια μακριά ακμή με μήκος M είναι δυνατόν να έχει μέχρι και M-1 κάμψεις. Οσο περισσότερες κάμψεις έχουν οι μακριές ακμές σε ένα σχέδιο τόσο χειρότερη είναι η ποιότητα του σχεδίου.

Ενας γράφος (προσανατολισμένος ή μη) λέγεται *επίπεδος* (planar) όταν μπορεί να σχεδιαστεί στο επίπεδο έτσι ώστε καμία ακμή να μην τέμνεται με άλλη ακμή. Οταν ο γράφος δεν είναι επίπεδος τότε δημιουργούνται *τομές ακμών* (edge crossings). Στο σχήμα 1.2 έχουμε μία τομή ακμής, αφού οι ακμές (B, Δ) και (Γ, E) τέμνονται. Γενικά όταν σε ένα σχέδιο υπάρχει μεγάλος αριθμός τομών ακμών αναλογικά με τον αριθμό των ακμών τότε η ποιότητα του σχεδίου δεν είναι καλή και ο χρήστης δυσκολεύεται να κατανοήσει το περιεχόμενό του.

1.2 Η Οργάνωση της Εργασίας

Το υπόλοιπο αυτής της εργασίας οργανώνεται ως εξής.

Το δεύτερο κεφάλαιο αναφέρεται σε προϋπάρχουσες ερευνητικές εργασίες σχετικές με το πρόβλημα του σχεδιασμού δέντρων και προσανατολισμένων γράφων σε ιεραρχική μορφή.

Στο τρίτο κεφάλαιο περιγράφουμε τον αλγόριθμο που σχεδιάσαμε. Επίσης επεξηγείται η διαδικασία της δημιουργίας του σχεδίου ενός παραδείγματος προσανατολισμένου γράφου από τον αλγόριθμο, βήμα-βήμα.

Το τέταρτο κεφάλαιο περιέχει μια σειρά μετρήσεων της απόδοσης του αλγορίθμου ως προς διάφορα κριτήρια. Γίνεται επίσης μία σύγκριση του νέου αλγορίθμου και του αλγορίθμου των Sugiyama, Tagawa και Toda, χρησιμοποιώντας ένα σύνολο από 12 γράφους διαφόρων μεγεθών. Αρκετοί από τους γράφους του παραπάνω συνόλου έχουν χρησιμοποιηθεί και σε προηγούμενες εργασίες.

Τελος, το πέμπτο κεφάλαιο συνοψίζει τα αποτελέσματα που παρουσιάζονται σ'αυτή την εργασία και προτείνει δυνατές βελτιώσεις και επεκτάσεις του αλγορίθμου που σχεδιάσαμε.

1.3 Η Υλοποίηση του Αλγορίθμου

Ο αλγόριθμος υλοποιήθηκε στη γλώσσα C++ [28] χρησιμοποιώντας τη βιβλιοθήκη κλάσεων LEDA έκδοση 2.1 [24] και το λειτουργικό σύστημα UNIX.

Συνολικά ασχοληθήκαμε 2.5 χρόνια με το πρόβλημα του σχεδιασμού γράφων. Ο πρώτος χρόνος αφιερώθηκε στην υλοποίηση του αλγορίθμου των Sugiyama, Tagawa και Toda (αλγόριθμος STT), ενώ το υπόλοιπο διάστημα αφιερώθηκε στη σχεδίαση του νέου αλγορίθμου και την υλοποίησή του η οποία έγινε παράλληλα με τη σχεδίαση. Ο κώδικας του νέου αλγορίθμου αποτελείται από 8000 γραμμές από τις οποίες οι 2000 είναι αναχρησιμοποιούμενος κώδικας. Η υλοποίηση του αλγορίθμου STT αποτελείται από 4500 γραμμές κώδικα.

Ο αλγόριθμος δέχεται ως είσοδο μια περιγραφή του γράφου η οποία αποτελείται από το σύνολο των ακμών του γράφου (ζευγάρια κόμβων). Στην έξοδο ο αλγόριθμος δημιουργεί ένα αρχείο στο οποίο περιγράφεται η τοποθέτηση των κόμβων στις υπολογισμένες συντεταγμένες. Το αρχείο αυτό έχει κατάλληλη κωδικοποίηση εφόσον είναι η είσοδος στο σύστημα Labyrinth [18] που χρησιμοποιείται για τη σχεδίαση του γράφου.

Ο αλγόριθμος έχει δοκιμαστεί επί 6 μήνες περίπου με εισόδους γράφους διαφόρων μεγεθών.

Κεφάλαιο 2

Προηγούμενες Ερευνητικές Εργασίες

Ο πρώτος αλγόριθμος σχεδιασμού γράφων αποδίδεται στον Knuth [19]. Ο Knuth το 1963 προσπάθησε να αυτοματοποιήσει το σχεδιασμό των διαγραμμάτων ροής (flow charts). Στις τρεις δεκαετίες που μεσολάβησαν εως σήμερα έχει γίνει σημαντική έρευνα πάνω στον *αυτόματο σχεδιασμό γράφων*. Εχουν αναπτυχθεί πολλοί αλγόριθμοι σχεδιασμού γράφων, ενώ συνεχίζουν να υπάρχουν ανοιχτά ερευνητικά θέματα. Οι περισσότεροι από τους αλγορίθμους αυτούς έχουν σχεδιαστεί ειδικά για συγκεκριμένες κλάσεις γράφων και παίρνουν υπόψη τους διάφορα κριτήρια που επιβάλουν οι εφαρμογές που τους χρησιμοποιούν.

Αυτό το κεφάλαιο παρουσιάζει διάφορες ερευνητικές εργασίες που αναπτύσσουν αλγορίθμους για σχεδιασμό *δέντρων* και *γενικών προσανατολισμένων γράφων*. Αναφερόμαστε μόνο σε αλγορίθμους που εξετάζουν αυτές τις δύο κλάσεις γράφων γιατί ο νέος αλγόριθμος που προτείνουμε αντιμετωπίζει το πρόβλημα του σχεδιασμού προσανατολισμένων γράφων σε ιεραρχική μορφή και ένα τμήμα του είναι ένας αλγόριθμος σχεδιασμού δέντρων. Περισσότερο εκτεταμένες παρουσιάσεις αλγορίθμων που σχεδιάζουν γράφους που ανήκουν σ'αυτές τις δύο κλάσεις αλλά και άλλες, όπως *επίπεδοι γράφοι* (planar graphs), *γενικοί μη προσανατολισμένοι γράφοι* (general undirected graphs) κ.α. δίνονται στα [2] και [21].

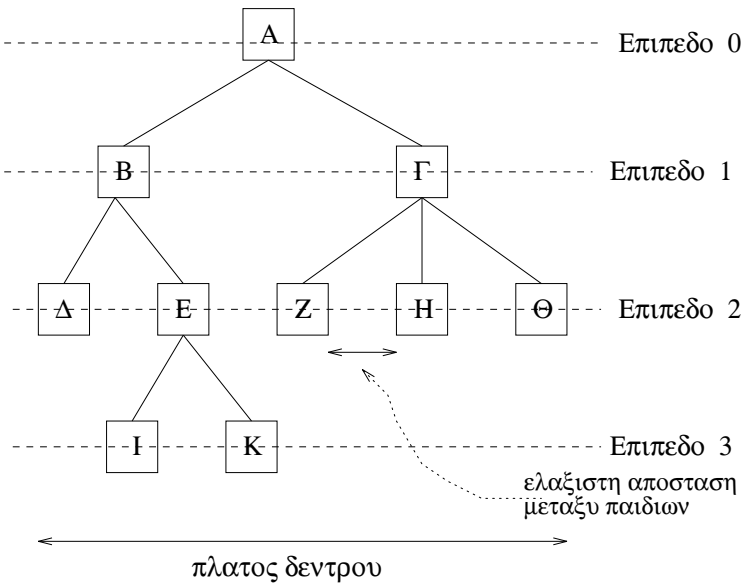
2.1 Δέντρα

Τα δέντρα είναι μία κλάση γράφων που χρησιμοποιείται πολύ συχνά για την αναπαράσταση ιεραρχιών όπως γενεαλογικά δέντρα ή στα πλαίσια της Επιστήμης Υπολογιστών για την αναπαράσταση *δέντρων αποφάσεων* (decision trees), *δέντρων αναζήτησης* (search trees) και γενικά για την αναπαράσταση πληροφορίας που περιέχει σχέσεις τύπου ''πατέρας-παιδί''. Οι αλγόριθμοι που σχεδιάζουν δέντρα γενικά υπακούουν στους παρακάτω βασικούς αισθητικούς κανόνες:

1. Το σχέδιο είναι σε ιεραρχική μορφή.
2. Ο σχεδιασμός είναι επίπεδος (καμιά ακμή δέντρου δεν τέμνει καμία άλλη).
3. Οι ακμές σχεδιάζονται με ευθείες γραμμές.
4. Το μέγιστο επίπεδο της ιεραρχίας είναι ίσο με το μέγιστο βάθος.

Με βάση τους παραπάνω αισθητικούς κανόνες, ένας πολύ απλός κανόνας για το καθορισμό της συντεταγμένης ψ κάθε κόμβου είναι ο εξής: $y = c \times$ βάθος(v), όπου c είναι μια σταθερά. Οι αλγόριθμοι που εξετάζουμε στη συνέχεια χρησιμοποιούν και τους παρακάτω αισθητικούς κανόνες (όχι όλους μαζί κατ'ανάγκη)

1. Οι γονείς (parents) σχεδιάζονται στη μεσοκάθετο της οριζόντιας απόστασης μεταξύ των παιδιών τους και πάνω από αυτά.
2. Υπάρχει μια ελάχιστη απόσταση μεταξύ δύο διαδοχικών παιδιών.
3. Το πλάτος του σχεδίου είναι το ελάχιστο δυνατό.

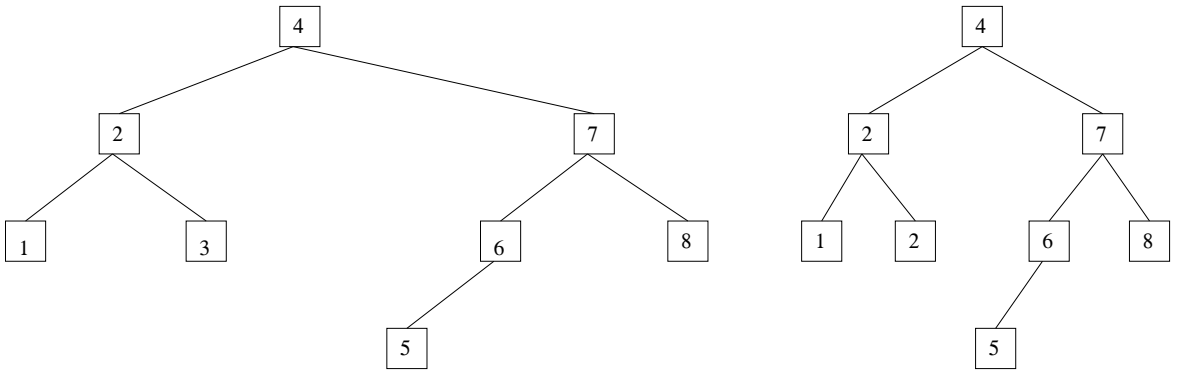


Σχήμα 2.1: Ενα παράδειγμα σχεδίου δέντρου

Ετσι για το δέντρο T = (V, E) όπου V = {A, B, Γ, Δ, E, Z, H, Θ, I, K} και E = {(A,B), (A,Γ), (B,Δ), (B,E), (Γ,Z), (Γ,H), (Γ,Θ), (E,I), (E,K)} όλοι οι αλγόριθμοι θα δημιουργούσαν ένα σχέδιο παρόμοιο με αυτό του σχήματος 2.1. Πρίν περιγράψουμε τους

αλγορίθμους, σημειώνουμε οτι όλοι οι αλγόριθμοι έχουν πολυπλοκότητα O(n) και δημιουργούν σχέδια με πλάτος O(n).

Ενας από τους παλαιότερους αλγορίθμους σχεδιασμού δέντρων είναι αυτός που παρουσιάστηκε από τον Knuth το 1971 [20] για το σχεδιασμό δυαδικών δέντρων. Ουσιαστικά ο αλγόριθμος αυτός αριθμεί τους κόμβους του δέντρου με τη σειρά που τους συναντά σε μία ενδοδιατεταγμένη (inorder) διάσχιση του δέντρου και τοποθετεί κάθε κόμβο σε μία οριζόντια συντεταγμένη ανάλογη με τον αριθμό που του έχει δοθεί. Ο αλγόριθμος του Knuth τείνει να δημιουργεί πλατιά σχέδια αφού επιτρέπει μόνο ένα κόμβο ανά κάθετη στήλη χώρου. Αυτό φαίνεται και στο σχήμα 2.2 όπου το δέντρο αριστερά έχει σχεδιαστεί με τον αλγόριθμο του



Σχήμα 2.2: Σχέδιο δέντρου με τον αλγόριθμο του Knuth

Knuth ενώ δεξιά φαίνεται πως θα μπορούσε να σχεδιαστεί το ίδιο δέντρο χωρίς το περιττό πλάτος.

Οι Wetherell και Shannon [37] προσπάθησαν να βελτιώσουν τον αλγόριθμο του Knuth. Σχεδίασαν ένα αλγόριθμο ο οποίος δημιουργούσε σχέδια με ελάχιστο πλάτος και γονείς ‘‘κεντραρισμένους’’ πάνω από τα παιδιά τους. Ο αλγόριθμός τους όμως δεν είχε γραμμικό χρόνο εκτέλεσης και γι’αυτό σχεδίασαν δύο άλλους αλγορίθμους με παρόμοια δομή και γραμμικό χρόνο εκτέλεσης, οι οποίοι ικανοποιούσαν ο καθένας έναν από τους παραπάνω αισθητικούς κανόνες αλλά όχι και τον άλλο. Οι δύο αλγόριθμοι τοποθετούν κάθε κόμβο του δέντρου κάνοντας δύο διασχίσεις του δέντρου. Στη πρώτη όλοι οι κόμβοι σπρώχνονται προς τα αριστερά χωρίς να καλύπτει ο ένας τον άλλο. Στη δεύτερη τοποθετούνται στη σωστή θέση σχετικά με τις θέσεις του πατέρα και των παιδιών τους.

Ο Vaucher [33] σχεδίασε έναν αλγόριθμο παρόμοιο με τον παραπάνω. Αν και η [33] δεν αναφέρεται στο πλάτος των σχεδίων που παράγονται, το πλάτος αυτό φαίνεται να είναι

ελάχιστο.

Οι Reingold και Tilford [25] αφού ασχολήθηκαν με τα προβλήματα που παρουσιάζουν οι παραπάνω αλγόριθμοι σχετικά με το σχεδιασμό ισομορφικών και συμμετρικών υποδέντρων, σχεδίασαν ένα αλγόριθμο (σχήμα 2.3) ο οποίος εγγυάται ότι αν δύο υποδέντρα είναι συμμε-

---

α. Εάν το *δέντρο* είναι κενό ή είναι ένας απλός κόμβος :

1. δημιούργησε ένα τετριμμένο σχέδιο του δέντρου
2. επέστρεψε.

β. Αναδρομικά δημιούργησε τα σχέδια του αριστερού και δεξιού υποδέντρου.

γ. Τοποθέτησε τα δύο υποδέντρα το ένα δίπλα στο άλλο έτσι ώστε να μην επικαλύπτονται.

δ. Τοποθέτησε τη ρίζα του δέντρου στο ενδιάμεσο της απόστασης μεταξύ των ριζών των δύο υποδέντρων.

Σχήμα 2.3: Ο αλγόριθμος των Reingold και Tilford για σχεδιασμό δέντρων

τρικά τότε αυτο φαίνεται και στο παραγόμενο σχέδιο όπου το ένα υποδέντρο σχεδιάζεται ως αντικατοπτρισμός του άλλου. Παρατηρούν όμως ότι ο σχεδιασμός συμμετρικών υποδέντρων συγκρούεται ορισμένες φορές με το περιορισμό για σχεδιασμό του δέντρου με ελάχιστο πλάτος. Η επέκταση του αλγορίθμου τους για δέντρα που δεν είναι δυαδικά είναι προφανής: αφού τα υποδέντρα έχουν σχεδιαστεί αναδρομικά τοποθετούνται το ένα δίπλα στο άλλο, η ρίζα μπορεί να τοποθετηθεί στο μέσον της απόστασης ανάμεσα στο δεξιότερο και το αριστερότερο παιδί της.

Οι Supowit και Reingold [30] μελέτησαν την υπολογιστική πολυπλοκότητα των αλγορίθμων σχεδιασμού δυαδικών δέντρων και έδειξαν ότι τα σχέδια που παράγονται από τον προηγούμενο αλγόριθμο μπορούν να είναι μεχρι και n φορές πλατύτερα από το ελάχιστο, όπου n είναι ο αριθμός των κόμβων του δέντρου. Επίσης, σχετικά με την πολυπλοκότητα των προβλημάτων σχεδιασμού δυαδικών δέντρων, συμπέραναν τα εξής:

α) το πρόβλημα της εύρεσης ενός σχεδίου με ελάχιστο πλάτος για ένα δυαδικό δέντρο, όπου τα συμμετρικά υποδέντρα σχεδιάζονται ομοιόμορφα, μπορεί να λυθεί σε πολυωνυμικό χρόνο αφού ανάγεται σε ένα πρόβλημα γραμμμικού προγραμματισμού και

β) αν περιορίσουμε το πρόβλημα έτσι ώστε οι συντεταγμένες χ των κόμβων να έχουν διακριτές τιμές (ακέραια πολ/σια του δ/2, όπου δ η ελάχιστη απόσταση μεταξύ δύο παιδιών), τότε το πρόβλημα είναι NP-hard.

Ο Walker [34] περιέγραψε έναν αλγόριθμο για σχεδιασμό γενικών δέντρων (οι κόμβοι μπορούν να έχουν βαθμό μεγαλύτερο από δύο) με βάση τον αλγόριθμο των Reingold και Tilford. Ο αλγόριθμος έχει γραμμικό χρόνο εκτέλεσης. Ο Walker αναφέρει χωρίς να το αποδεικνύει, ότι ο αλγόριθμός του ελαχιστοποιεί το πλάτος των παραγόμενων σχεδίων και ισχυρίζεται ότι το πρόβλημα σχεδιασμού γενικών δέντρων δεν είναι NP-hard όπως το πρόβλημα του σχεδιασμού δυαδικών δέντρων. Στα δυαδικά δέντρα κάθε παιδί τοποθετείται είτε δεξιά είτε αριστερά του πατέρα του (ακόμα κι αν αυτός ο πατέρας έχει ένα μόνο παιδί) και αυτός ο περιορισμός κάνει το πρόβλημα NP-hard. Εκτός από τους βασικούς αισθητικούς κανόνες αυτός ο αλγόριθμος παίρνει υπόψη του αρκετές πρακτικές λεπτομέρειες όπως: εναλλακτικές κατευθύνσεις σχεδίασης του δέντρου, μεταβλητό μέγεθος κόμβων, κ.α. Για τη τοποθέτηση κάθε κόμβου χρησιμοποιούνται δύο πεδία:

α) μια προκαταρκτική συντεταγμένη χ και

β) ένα πεδίο τροποποίησης (modifier field) (χρησιμοποιείται από τους εσωτερικούς κόμβους -- αυτούς που έχουν προηγούμενο αδέρφι -- ώστε να μπορούν να τοποθετήσουν τα παιδιά τους)

Ο αλγόριθμος κάνει δύο διασχίσεις του δέντρου προκειμένου να υπολογίσει τις συντεταγμένες χ των κόμβων. Η πρώτη διάσχιση είναι μια μεταδιατεταγμένη (post-order) διάσχιση του δέντρου όπου τα μικρότερα υποδέντρα (φύλλα) τοποθετούνται πρώτα και κατόπιν από δεξιά προς τα αριστερά τοποθετούνται και τα μεγαλύτερα υποδέντρα. Κατά τη πρώτη διάσχιση δίνονται αρχικές τιμές στις συντεταγμένες χ και στο πεδίο τροποποίησης σε όλους τους κόμβους. Η δεύτερη διάσχιση είναι προδιατεταγμένη (pre-order) και κατά τη διάρκειά της υπολογίζεται η τελική συντεταγμένη χ κάθε κόμβου αθροίζοντας τη προκαταρκτική συντεταγμένη χ και τα πεδία τροποποίησης όλων των προγόνων του.

Ο Moen [23] σχεδίασε έναν αλγόριθμο ο οποίος έχει τα εξής τρία πλεονεκτήματα σε σχέση με τους αλγόριθμους που περιγράφηκαν προηγουμένως:

α) οι κόμβοι του δέντρου μπορούν να έχουν διαφορετικούς τύπους και μεγέθη,

β) ο αλγόριθμος κάνει τη μέγιστη δυνατή οικονομία χώρου και το παραγόμενο σχέδιο είναι συμπυκνωμένο (ιδιαίτερα χρήσιμη ιδιότητα όταν σχεδιάζονται μεγάλα δέντρα) και

γ) η πληροφορία που δημιουργείται κατά τη διάρκεια του σχεδιασμού μπορεί να αναχρησιμοποιηθεί ελαχιστοποιώντας έτσι το χρόνο που χρειάζεται για να σχεδιάσουμε το δέντρο μετά από κάποια αλλαγή.

Ο αλγόριθμος δημιουργεί το γεωμετρικό περίγραμμα κόμβων και υποδέντρων του δέντρου και σχηματίζει το τελικό σχέδιο ενώνοντας κατάλληλα τα περιγράμματα. Ο χρόνος εκτέλεσης είναι γραμμικός. Επειδή αυτός ο αλγόριθμος είναι σημαντικός στα πλαίσια αυτής της εργασίας αναλύεται περισσότερο στο Παράρτημα Α.

2.2 Γενικοί Προσανατολισμένοι Γράφοι

Οπως έχει ήδη αναφερθεί, οι προσανατολισμένοι γράφοι είναι μία κλάση γράφων που συναντάται σε μία πληθώρα εφαρμογών. Η πλειοψηφία των αλγορίθμων που σχεδιάζουν προσανατολισμένους γράφους δημιουργούν σχέδια σε *ιεραρχική* μορφή, δηλαδή θεωρούν υποσύνολα των κόμβων τα οποία ονομάζονται *επίπεδα*. Οι κόμβοι ενός επιπέδου σχεδιάζονται σε μία οριζόντια ή κάθετη γραμμή (ανάλογα με την επιλογή για τη κατεύθυνση των ακμών, από πάνω προς τα κάτω ή από αριστερά προς τα δεξιά). Η επιλογή των κόμβων ενός επιπέδου του γράφου δεν είναι τυχαία, αλλά έχει σκοπό τη μεγιστοποίηση του αριθμού των ακμών που έχουν την ίδια κατεύθυνση. Κατ'αυτό το τρόπο ο χρήστης αντιλαμβάνεται ευκολότερα τις σχέσεις πρόγονος/απόγονος που απεικονίζονται στο γράφο. Αυτός άλλωστε είναι και ο λόγος που οι περισσότεροι αλγόριθμοι επιλέγουν να δημιουργούν ιεραρχικά σχέδια σε αντίθεση με άλλους οι οποίοι δημιουργούν σχέδια όπου οι κόμβοι βρίσκονται ελεύθερα τοποθετημένοι στο επίπεδο.

Οι αλγόριθμοι που σχεδιάζουν γενικούς προσανατολισμένους γράφους γενικά υπακούουν στους παρακάτω βασικούς αισθητικούς κανόνες:

1. Η ιεραρχία σχεδιάζεται με φορά από πάνω προς τα κάτω.
2. Ο αριθμός τομών ακμών (edge crossings) θέλουμε να είναι όσο το δυνατόν μικρότερος.
3. Κάμψεις σε ακμές πρέπει να αποφεύγονται.
4. Ο χώρος που καταλαμβάνει το σχέδιο πρέπει να είναι ο ελάχιστος δυνατόν.
5. Οι κόμβοι του γράφου πρέπει να είναι ομοιόμορφα κατανεμημένοι στο επίπεδο.

Οπως είπαμε και στην αρχή αυτού του κεφαλαίου, ο Knuth παρουσίασε το 1963 ένα αλγόριθμο σχεδιασμού διαγραμμάτων ροής. Η δομή των γράφων που είχε να αντιμετωπίσει ο αλγόριθμος είναι σχετικά απλή και έτσι και ο αλγόριθμος δεν είναι ιδιαίτερα πολύπλοκος: κάθε κόμβος σχεδιάζεται ακριβώς πάνω από τον επόμενο σύμφωνα με τη ροή του προγράμματος, οι ακμές που δείχνουν ροή προς τα εμπρός σχεδιάζονται δεξιά από τους κόμβους ενώ αυτές που δηλώνουν ροή προς τα πίσω σχεδιάζονται αριστερά από τους κόμβους.

Ενώ ο αλγόριθμος τους Knuth αντιμετωπίζει μία ειδική περίπτωση προσανατολισμένων γράφων, ο Warfield το 1976 [35] παρουσίασε έναν αλγόριθμο για το σχεδιασμό γενικών προσανατολισμένων άκυκλων γράφων. Ο αλγόριθμός του εξασφαλίζει ότι όλες οι ακμές έχουν την ίδια φορά. Αυτό επιτυγχάνεται ως εξής: αρχικά βρίσκει όλους τους κόμβους οι οποίοι δεν έχουν διάδοχο και τους τοποθετεί στο τελευταίο επίπεδο του γράφου, αυτοί οι κόμβοι αφαιρούνται από το γράφο και η ίδια διαδικασία συνεχίζεται με το επόμενο επίπεδο προς τα πάνω έως το επίπεδο 1. Για γράφους οι οποίοι έχουν κύκλους, ο Warfield περιγράφει μία μέθοδο η οποία βρίσκει τους μέγιστους κύκλους σε ένα γράφο και τους συμπυκνώνει σε ένα ειδικό κόμβο ο οποίος αντιπροσωπεύει όλους τους κόμβους του κύκλου, καθώς και τους προκατόχους και τους διαδόχους τους. Το παραγόμενο σχέδιο περιέχει αυτούς τους ειδικούς κόμβους. Στην ίδια εργασία περιγράφονται διάφοροι τρόποι για τη παρουσίαση των κόμβων οι οποίοι έχουν συμπυκνωθεί για τη δημιουργία των ειδικών κόμβων, έτσι ώστε να φαίνεται η δομή του του αρχικού γράφου. Η Carpano [5] αναπτύσσει ένα τρόπο για την παρουσίαση των ειδικών κόμβων σε τρείς διαστάσεις. Αργότερα ο Davis [8] παρουσίασε μία μέθοδο για αφαίρεση των κύκλων από γράφους. Η μέθοδός του αντιστρέφει προσωρινά κάποιες επιλεγμένες ακμές (αυτές που ‘‘κλείνουν’’ κάποιο κύκλο), σχεδιάζει το γράφο και στο τελικό σχέδιο δίνει στις ακμές που έχουν αντιστραφεί τη πραγματική τους φορά.

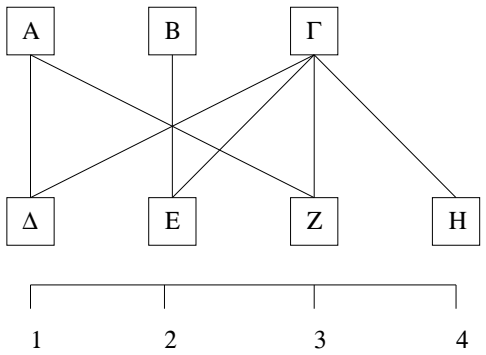
Ο Warfield στο [36] παρουσιάζει τη θεωρία των τομών ακμών (crossing theory) για ιεραρχικούς γράφους και περιγράφει έναν αλγόριθμο ο οποίος δοσμένης μιας ιεραρχίας δύο επιπέδων βρίσκει την αντίστοιχη ιεραρχία με τις ελάχιστες τομές ακμών. Αυτο επιτυγχάνεται βρίσκοντας τη μετάθεση των κόμβων των δύο επιπέδων η οποία αντιστοιχεί στον ελάχιστο αριθμό τομών ακμών. Ο Warfield δεν ασχολήθηκε ούτε με περισσότερα από δύο επίπεδα ιεραρχίας αλλά ούτε και με τη μελέτη της πολυπλοκότητας των προβλημάτων αυτών.

Ο Eades κ.α. [12] έδειξαν ότι το πρόβλημα της ελαχιστοποίησης των τομών ακμών σε ιεραρχίες με k επίπεδα είναι NP-hard ακόμα και όταν k = 2 και οι κόμβοι του δεύτερου επιπέδου έχουν τοποθετηθεί σε προκαθορισμένες σταθερές θέσεις. Αυτό το θεωρητικό αποτέλεσμα οδήγησε πολλούς ερευνητές [11] [13] στην κατασκευή ευρηματικών μεθόδων για την ελάττωση των τομών ακμών σε ιεραρχίες με k επίπεδα.

Οι Di Battista και Nardelli [3] παρουσίασαν έναν αλγόριθμο ο οποίος εξετάζει εάν μιά ιεραρχία με k επίπεδα και μία ρίζα μπορεί να σχεδιαστεί χωρίς τομές ακμών (αν η ιεραρχία είναι δηλαδή επίπεδος γράφος), και στη περίπτωση αύτή, ο αλγόριθμος σχεδιάζει την ιεραρχία χωρίς τομές ακμών.

Ο Delarche [9] ανέπτυξε μία γρήγορη ευρηματική μέθοδο για την ελάττωση των τομών ακμών σε ιεραρχίες με δύο επίπεδα, η οποία αναφέρεται επίσης από τον Carpano [5]. Αυτή η μέθοδος συνδυάζει την ιδέα του Warfield για την μετάθεση των κόμβων ενός επιπέδου και

ένα κριτήριο για ταξινόμηση κόμβων, το *βαρύκεντρο*, το οποίο είχε αναπτυχθεί από τον Tutte [32]. Ο Delarche διαφοροποιεί την έννοια του βαρυκέντρου σε σχέση με τον Tutte, και όταν αναφέρεται σε ιεραρχίες δύο επιπέδων εννοεί τον μέσον όρο των θέσεων των γειτόνων ενός κόμβου αν υποθέσουμε ότι όλοι έχουν τοποθετηθεί σε έναν οριζόντιο άξονα. Για παράδειγμα, στο σχήμα 2.4 το βαρύκεντρο του κόμβου Α είναι ο μέσος όρος των θέσεων των γειτόνων



Σχήμα 2.4: Υπολογισμός του βαρυκέντρου

του Δ και Ζ, δηλαδή 2, το βαρύκεντρο του κόμβου Γ είναι 2.5, του δε κόμβου Η είναι 3. Στη συνέχεια δίνεται η μέθοδος του Delarche για την ελάττωση των τομών ακμών. Οι κόμβοι του επάνω επιπέδου ανακατατάσσονται έτσι ώστε να είναι ταξινομημένοι σύμφωνα με τα βαρύκεντρά τους. Αν έστω και ένας κόμβος αλλάξει θέση λόγω της ανακατάταξης, τότε υπολογίζονται και τα βαρύκεντρα των κόμβων του κάτω επιπέδου και ανακατατάσσονται και αυτοί με τον ίδιο τρόπο. Κατόπιν, υπολογίζονται πάλι τα βαρύκεντρα των κόμβων του επάνω επιπέδου και η ίδια διαδικασία που περιγράφηκε πριν επαναλαμβάνεται έως ότου οι κόμβοι αποκτήσουν σταθερές θέσεις.

Κάθε φορά που δημιουργείται από την παραπάνω ευρηματική μέθοδο μία μετάθεση του τρέχοντος επιπέδου κόμβων όπου όλοι οι κόμβοι είναι ταξινομημένοι σύμφωνα με τα βαρύκεντρά τους, το σχέδιο που παράγεται βελτιώνεται γιατί αυτό που ουσιαστικά επιτυγχάνεται είναι:

α) η απόσταση μεταξύ κόμβων και των προγόνων και των απογόνων τους μικραίνει (η κάθετη λωρίδα χώρου που τους περιέχει είναι γενικά στενότερη σε σχέση με αυτή που τους περιείχε πριν την ταξινόμηση),

β) η μεγιστοποίηση του αριθμού των ακμών που είναι κάθετες (αν κάποιες από τις ακμές μπορούν να σχεδιαστούν κάθετες) και

γ) η ελαχιστοποίηση των τομών ακμών.

Η μέθοδος με βάση τα βαρύκεντρα είναι γρήγορη και είναι μιά ευρηματική προσέγγιση της εναλλακτικής μεθόδου του εξαντλητικού ελέγχου όλων των μεταθέσεων των κόμβων ενός επιπέδου προκειμένου να βρεθεί αυτή που αντιστοιχεί στις λιγότερες τομές ακμών. Ο Sugiyama κ.α. [29] βρήκε ικανοποιητική την αποδοτικότητα της μεθόδου χρησιμοποώντας την για το σχεδιασμό ενός συνόλου παραδειγμάτων ιεραρχιών.

Ο Carpano γενικεύει την παραπάνω ευρηματική μέθοδο έτσι ώστε να μπορεί να χρησιμοποιηθεί σε ιεραρχίες με μήκος k > 2. Στα πλαίσια τέτοιων ιεραρχιών η έννοια του βαρυκέντρου επεκτείνεται και ορίζονται δύο νέα κριτήρια ταξινόμησης, το *πάνω-βαρύκεντρο* και το *κάτω-βαρύκεντρο*. Το πάνω-βαρύκεντρο είναι ο μέσος όρος των θέσεων των προκατόχων ενός κόμβου, ενώ το κάτω-βαρύκεντρο είναι ο μέσος όρος των θέσεων των διαδόχων ενός κόμβου (οι κόμβοι του πρώτου επιπέδου δεν έχουν πάνω-βαρύκεντρο και οι κόμβοι του τελευταίου επιπέδου δεν έχουν κάτω-βαρύκεντρο). Στο σχήμα 2.5 περιγράφεται η μέθοδος του Carpano.

α. Ταξινόμησε τους κόμβους των επιπέδων 1 και 2 χρησιμοποιώντας τον αλγόριθμο του Delarche.

β. Για κάθε επίπεδο κόμβων i, όπου $3 \leq i \leq k$:

1. Υπολόγισε το πάνω-βαρύκεντρο κάθε κόμβου στο επίπεδο i.
2. Ταξινόμησε τους κόμβους του επιπέδου i σύμφωνα με το πάνω-βαρύκεντρο.
3. Υπολόγισε το κάτω-βαρύκεντρο όλων των κόμβων των επιπέδων j, όπου $1 \leq j \leq i-1$ σύμφωνα με τις καινούργιες θέσεις των κόμβων στο επίπεδο i και ταξινόμησε τα επίπεδα j σύμφωνα με το κάτω-βαρύκεντρο.

Σχήμα 2.5: Ο αλγόριθμος του Carpano για ιεραρχίες μήκους k.

Παράλληλα με τον Carpano, οι Sugiyama, Tagawa και Toda [29] ανέπτυξαν έναν αλγόριθμο, ο οποίος έχει πολλά κοινά σημεία με τον αλγόριθμο του Carpano, για το σχεδιασμό ιεραρχιών με k επίπεδα ο οποίος βασίζεται και αυτός στον αλγόριθμο του Delarche. Αυτός ο αλγόριθμος (τον οποίο θα αναφέρουμε από εδώ και στο εξής ως αλγόριθμο STT) είναι πιο ολοκληρωμένος σε σχέση με τον αλγόριθμο του Carpano και αποτελείται από τρεις κύριες φάσεις:

1. **Ανάθεση κόμβων σε επίπεδα.** Αυτή η φάση τοποθετεί τους κόμβους σε επίπεδα και μετατρέπει την ιεραρχία σε κανονική ιεραρχία. Αν υπάρχουν κύκλοι στο γράφο, τότε ακολουθείται η μέθοδος που περιέγραψε ο Warfield σχετικά, και δημιουργούνται οι ειδικοί κόμβοι που αντιπροσωπεύουν τους κύκλους.

2. **Ελαχιστοποίηση τομών ακμών.** Γίνεται μετάθεση των κόμβων ανά επίπεδο έτσι ώστε

να είναι ταξινομημένοι σύμφωνα με τις τιμές των βαρυκέντρων τους με παρόμοιο τρόπο με αυτόν της Carpano. Η σειρά με την οποία γίνονται αυτές οι μεταθέσεις είναι διαφορετική: πρώτα οι κόμβοι στα επίπεδα 2 έως k ταξινομούνται σύμφωνα με τη τιμή του πάνω-βαρύκεντρού τους και μετά οι κόμβοι των επιπέδων k-1 έως 1 ταξινομούνται σύμφωνα με τη τιμή του κάτω-βαρύκεντρού τους.

3. **Υπολογισμός της συντεταγμένης χ των κόμβων.** Σ'αυτή τη φάση δίνονται τελικές τιμές στην συντεταγμένη χ κάθε κόμβου. Η μέθοδος που ακολουθείται είναι παρόμοια με τη μέθοδο των βαρυκέντρων, όμως εδώ δεν επιτρέπεται να αλλάξει η σχετική τοποθέτηση των κόμβων (η οποία υπολογίστηκε κατά τη προηγούμενη φάση). Δίνεται προτεραιότητα στο καθορισμό της θέσης κόμβων που συνδέονται με μεγάλο αριθμό άλλων κόμβων καθώς και των τεχνητών κόμβων.

Πολλοί ερευνητές ασχολήθηκαν με τη βελτίωση και επέκταση του αλγόριθμου STT και ανάμεσά τους οι Meyer [22], Davis [8], Rowe [27] και Messinger [21]. Πρώτη επέκταση του αλγορίθμου ήταν η δημιουργία ενός ακόμα κριτηρίου για ταξινόμηση κόμβων, το *πάνω-κάτω-βαρύκεντρο* το οποίο δίνει το μέσον όρο των θέσεων των προκατόχων και διαδόχων ενός κόμβου. Στη δεύτερη φάση προστέθηκε μια τρίτη επίσκεψη των κόμβων κατά επίπεδα, κατά την οποία οι κόμβοι ταξινομούνται σύμφωνα με το νέο κριτήριο ταξινόμησης, το πάνω-κάτω-βαρύκεντρο. Μια δεύτερη επέκταση από τον Davis, ήταν ο χειρισμός των κύκλων όπως περιγράφηκε προηγουμένως. Ο Davis περιγράφει επίσης μια ευρηματική μέθοδο η οποία εφαρμόζεται στη τρίτη φάση του αλγορίθμου και πετυχαίνει να ισιώσει τμήματα των μακριών ακμών όπου αυτό είναι δυνατόν, έτσι ώστε να μειωθεί ο αριθμός των κάμψεων ακμών. Τέλος, ο Messinger έκανε μερικές επιπλέον επεκτάσεις ο οποίες αναφέρονται παρακάτω.

Αλλες εργασίες [11] [16], εξετάζουν εναλλακτικές μεθόδους ταξινόμησης των κόμβων, οι οποίες μπορούν να χρησιμοποιηθούν κατά τη δεύτερη φάση αντί της μεθόδου των βαρυκέντρων, χωρίς όμως να καταφέρνουν αισθητά μεγαλύτερη μείωση στον αριθμό των τομών ακμών από αυτή που πετυχαίνει ο STT χρησιμοποιώντας τη μέθοδο των βαρυκέντρων. Οι μέθοδες αυτές φαινεται πως έχουν εγκαταληφθεί.

Οι Gansner, North και Vo [16] επεκτείνουν τον STT εισάγοντας νέες μεθόδους και στις τρεις φάσεις του αλγορίθμου. Επίσης δημιουργούν και μια τέταρτη φάση κατά την οποία κάποιες ακμές σχεδιάζονται καμπύλες έτσι ώστε να αποφευχθούν οι απότομες γωνίες στα σημεία που κάμπτονται οι ακμές, και παράλληλα να μην υπάρχουν κόμβοι και ακμές που επικαλύπτονται. Ο αλγόριθμός τους χρησιμοποιείται στα πλαίσια του προγράμματος DAG, το οποίο είναι ένα εργαλείο σχεδίασης προσανατολισμένων γράφων. Το DAG προσφέρει στο χρήστη δυνατότητες σχεδίασης που δεν προσέφεραν προηγούμενα συστήματα, όπως

η δυνατότητα εισαγωγής περιορισμών σχετικά με τη τοποθέτηση κόμβων, η δυνατότητα επιλογής παραμέτρων σχεδιασμού όπως ο τύπος των κόμβων και ακμών, η απόσταση μεταξύ διαδοχικών επιπέδων κ.α.

Στη συνέχεια δίνονται οι κυριότερες βελτιώσεις του STT από τους Gansner, North και Vo. Στην πρώτη φάση χρησιμοποιούν τεχνικές ακέραιου προγραμματισμού για την ανάθεση κόμβων σε επίπεδα. Αυτή η τεχνική έχει το πλεονέκτημα ότι μπορούν οι χρήστες να εισάγουν βάρη σε ακμές έτσι ώστε να επηρεάσουν τη τελική τοποθέτηση των κόμβων, όπως και το μήκος ορισμένων ακμών. Στη δεύτερη φάση, χρησιμοποιούν το *σταθμισμένο μέσο* (weighted median) των κόμβων ως κριτήριο ταξινόμησής τους, αντί του βαρυκέντρου. Η μέθοδός τους παρουσιάζει κάποια μικρή βελτίωση στην ελάττωση των τομών ακμών σε σχέση με τον STT. Επίσης, περιγράφουν μια μέθοδο αντιμετάθεσης γειτονικών κόμβων χρησιμοποιώντας ως κριτήριο για την αντιμετάθεση είτε βαρύκεντρα είτε σταθμισμένους μέσους, καταφέρνοντας έτσι επιπλέον βελτίωση όσον αφορά τη μείωση των τομών ακμών. Στη δεύτερη φάση χρησιμοποιούν επίσης τεχνικές γραμμικού προγραμματισμού για τη τοποθέτηση των κόμβων στις τελικές συντεταγμένες τους. Επειδή η λύση ενός προβλήματος γραμμικού προγραμματισμού είναι χρονοβόρα, διατυπώνουν ευρηματικές μεθόδους προκειμένου να προσεγγίσουν τις λύσεις που θα έδινε ο γραμμικός προγραμματισμός.

Ενας άλλος αλγόριθμος που σχεδιάζει προσανατολισμένους γράφους σε ιεραρχική μορφή είναι αυτός του Robins [26]. Ο αλγόριθμος αυτός αποτελείται από δύο φάσεις. Στην πρώτη φάση δημιουργείται ένα δέντρο που περιέχει όλους τους κόμβους του γράφου (ένα spanning tree του γράφου). Στη δεύτερη φάση σχεδιάζεται το δέντρο που υπολογίστηκε πριν, με μία μέθοδο που χρησιμοποιεί δύο διασχίσεις οι οποίες χρειάζονται γραμμικό χρόνο εκτέλεσης. Εάν ο γράφος έχει κύκλους τότε δημιουργείται το αντίγραφο ενός κόμβου του κύκλου και έτσι ο κύκλος μπορεί πλέον να σχεδιαστεί σαν ένα μονοπάτι. Αυτά τα αντίγραφα κόμβων σχεδιάζονται εντονότερα από τους άλλους κόμβους έτσι ώστε να είναι ευδιάκριτο ότι αυτοί ο κόμβοι κλείνουν κάποιο κύκλο. Οι ακμές σχεδιάζονται ευθείες αγνοώντας τυχόν τομές μεταξύ κόμβων και ακμών.

Είναι φανερό ότι ο αλγόριθμος του Robins έχει αρκετά μειονεκτήματα. Πολλοί από τους αισθητικούς περιορισμούς που περιγράφηκαν προηγουμένως για δέντρα δεν ικανοποιούνται από αυτό τον αλγόριθμο όταν ο γράφος που σχεδιάζεται είναι ένα δέντρο. Ο αλγόριθμος δεν παίρνει υπόψη του ότι η επιφάνεια του σχεδίου πρέπει να είναι ελάχιστη και δεν κάνει επίσης καμιά προσπάθεια να ελαχιστοποιήσει τον αριθμό των τομών ακμών ή να αποφύγει τομές μεταξύ κόμβων και ακμών. Αυτό που ο αλγόριθμος πετυχαίνει είναι η σχεδίαση μεγάλων γράφων πάρα πολύ γρήγορα αφού η πλοκή του είναι γραμμική, έτσι ώστε να μπορεί να εφαρμοστεί όταν υπάρχουν απαιτήσεις για σχεδίαση πραγματικού χρόνου, όπως π.χ η επαφή

χρήσης ενός πληροφοριακού συστήματος.

Εκτός από τους αλγορίθμους που αναφέραμε προηγουμένως, υπάρχουν και άλλοι για το σχεδιασμό προσανατολισμένων γράφων σε ιεραρχική δομή, οι οποίοι όμως δεν έχουν να παρουσιάσουν σημαντικά αποτελέσματα και γενικά δεν χρησιμοποιήθηκαν από εφαρμογές. Είναι φανερό ότι κατά τη δεκαετία του '80 οι ερευνητές έδωσαν μεγάλο βάρος στη βελτίωση και επέκταση του αλγορίθμου STT. Η απόδοσή του επηρεάζεται σημαντικά από την αρχική διάταξη των κόμβων, τον αριθμο και το είδος των επαναλήψεων που κάνει. Οι δύο τελευταίες φάσεις του αλγορίθμου δεν επικοινωνούν μεταξύ τους με αποτελεσμα τη δημιουργία κακών σχεδίων που υπό άλλες συνθήκες θα είχαν σχεδιαστεί καλύτερα. Τέλος δημιουργεί αρκετές κάμψεις ακμών για την εξάλειψη των οποίων χρειάζεται μια τέταρτη φάση, η οποία καταναλώνει επιπλέον χρόνο χωρίς να εγγυάται καλά αποτελέσματα. Παρά τα προβλήματα αυτά, ο STT μπορεί να χρησιμοποιηθεί (αν πάρει κανείς υπόψη τις βελτιώσεις και επεκτάσεις) στην επαφή χρήσης ενός πληροφοριακού συστήματος. Μπορεί να θεωρηθεί δηλαδή ότι υπάρχει μια ικανοποιητική λύση για το πρόβλημα του σχεδιασμού μικρών και μεσαίων γράφων. Έτσι, μετά το 1987, οι ερευνητές στράφηκαν σε άλλα θέματα, όπως ο σχεδιασμός πολύ μεγάλων γράφων [21], η επέκταση των αλγορίθμων σχεδιασμού γράφων για την αντιμετώπιση περιορισμών που εισάγονται από το χρήστη [4], ο σχεδιασμός αλγορίθμων που θα επιτρέπει δυναμική τροποποίηση γράφων με αποδοτικό τρόπο, η σχεδίαση αλγορίθμων που θα μπορούν να εκτελούνται παράλληλα, κ.α.

Ο Messinger [21] εξέτασε διάφορες επεκτάσεις και βελτιώσεις στον αλγόριθμο STT τις οποίες συγκεντρώνει σ'έναν αλγόριθμο που ονομάζει GRAB. Βρήκε ότι η αρχική τοποθέτηση των κόμβων είναι πολύ σημαντική για την ποιότητα του παραγόμενου σχεδιού, αλλά δυστυχώς δεν υπάρχει τρόπος να καθοριστεί αν μία συγκεκριμένη αρχική τοποθέτηση κόμβων θα επηρεάσει θετικά ή αρνητικά τη ποιότητα του σχεδίου. Εξέτασε αν η χρησιμοποίηση ενός τροποποιημένου πάνω-κάτω-βαρύκεντρου, το οποίο θα χρησιμοποιεί βάρη και δεν θα στηρίζεται σε αυστηρά τοπική πληροφορία, μπορεί να επιφέρει βελτίωση, χωρίς όμως σημαντικά αποτελέσματα. Δημιούργησε μία νέα σειρά επισκέψεων των επιπέδων για την ταξινόμηση των κόμβων. Ο GRAB περιέχει μια αλληλουχία επισκέψεων επιπέδων, κάθε μία από τις οποίες εξετάζει πάνω-βαρύκεντρα, κάτω-βαρύκεντρα, ή ένα συνδυασμό πάνω-βαρυκέντρων και κάτω-βαρυκέντρων. Ακόμα, βελτίωσε την τρίτη φάση του αλγορίθμου STT έτσι ώστε να επιτρέπει μεταβλητές αποστάσεις ανάμεσα σε δύο επίπεδα με σκοπό την αποφυγή απότομων κλίσεων σε ορισμένες ακμές. Τέλος, παρουσίασε διάφορα παραδείγματα προβληματικών καταστάσεων, οι οποίες δημιουργούνται επειδή οι τρείς φάσεις του αλγορίθμου STT δεν επικοινωνούν μεταξύ τους. Έδειξε επίσης περιπτώσεις όπου η ευρηματική μέθοδος του Davis για την ελαχιστοποίηση των κάμψεων ακμών δεν δουλεύει καλά.

Τα κυριότερα συμπεράσματα του Messinger ήταν οτι η μέθοδος της χρησιμοποίησης των βαρυκέντρων για τη ταξινόμηση των κόμβων χρειάζεται βελτίωση και επίσης ότι η έλλειψη επικοινωνίας μεταξύ των τριών φάσεων του STT αλγορίθμου δημιουργεί διάφορα προβλήματα στη σχεδίαση του γράφου, τα οποία θα εξέλιπαν αν υπήρχε κάποιος τρόπος οι τρείς φάσεις του αλγορίθμου να ανταλλάσουν μεταξύ τους δεδομένα. Επίσης ο χρόνος εκτέλεσης του GRAB για το σχεδιασμό μεγάλων γράφων είναι μεγάλος (περισσότερο από 4 λεπτά για ένα γράφο 292 κόμβων και 558 ακμών σε ένα Sun-3/75) ώστε να μπορεί να χρησιμοποιηθεί σε εφαρμογές που απαιτούν διαλογική σχεδίαση και επεξεργάζονται πολύ μεγάλους γράφους.

Τα παραπάνω συμπεράσματα οδήγησαν τον Messinger στη σχεδίαση ενός αλγορίθμου για σχεδιασμό πολύ μεγάλων γράφων τον οποίο ονομάζει Compoze. Ο αλγόριθμός του χρησιμοποιεί την τεχνική διαίρει-και-βασίλευε για τη λύση του προβλήματος. Αρχικά βρίσκουμε έναν αριθμό υπογράφων του αρχικού γράφου. Ο αριθμός των υπογράφων που βρίσκονται εξαρτάται από το μέγεθος του γράφου. Με βάση τον τρόπο που συνδέονται αυτοί οι υπογράφοι μεταξύ τους, δημιουργείται ένας μεταγράφος του οποίου κόμβοι είναι οι υπογράφοι και οι ακμές του υποδηλώνουν το τρόπο σύνδεσης των υπογράφων. Ο GRAB χρησιμοποιείται αρχικά για το σχεδιασμό του μεταγράφου, και στο τέλος οι ακμές του μεταγράφου (κάθε μία από τις οποίες αντιστοιχεί σε ένα σύνολο ακμών του αρχικού γράφου) δρομολογούνται κατάλληλα. Διαιρώντας το αρχικό πρόβλημα σε πολλά μικρότερα προβλήματα, ο Messinger κατορθώνει να δημιουργήσει ένα πολυωνυμικό αλγόριθμο ο οποίος χρειάζεται λιγότερο χρόνο συνολικά για να λύσει όλα τα μικρότερα προβλήματα από όσο χρειάζεται για να λύσει το αρχικό. Η εργασία του Messinger είναι εντυπωσιακά απλή στη σύλληψή της και έχει πολύ καλά αποτελέσματα. Πάντως η χρησιμοποίηση κάποιου καλύτερου αλγορίθμου από τον GRAB για τη σχεδίαση των υπογράφων, θα βελτιώσει όπως είναι φυσικό την απόδοση του Compoze.

Κεφάλαιο 3

Περιγραφή του Αλγορίθμου

3.1 Εισαγωγή

Το κεφάλαιο αυτό περιγράφει τον αλγόριθμο που αναπτύξαμε για σχεδιασμό γράφων. Ο αλγόριθμος αυτός αναδιατάσσει τους κόμβους του γράφου έτσι ώστε το σχέδιο που θα προκύψει αν τοποθετήσουμε τους αναδιατεταγμένους κόμβους, να έχει λιγότερες τομές ακμών από ότι θα είχε αν δεν γινόταν η αναδιάταξη.

Το κίνητρο για την ανάπτυξη αυτού του αλγορίθμου ήταν η κακή απόδοση του αλγορίθμου STT στη περίπτωση μεγάλων γράφων. Για παράδειγμα, όσον αφορά το χρόνο σχεδίασης, χρειάστηκε 31.22 sec για να σχεδιάσει ένα γράφο με 178 κόμβους και 441 ακμές σε ένα Sun SparClassic 4.1.3 όπου εκτελούνταν μονάχα αυτή η διεργασία. Αν σκεφτούμε ότι αυτός ο χρόνος αποτελεί τμήμα μόνο του συνολικού χρόνου απόκρισης σε μια ερώτηση του χρήστη η οποία απαιτεί απάντηση με γραφικό τρόπο, τότε ένας αποδεκτός χρόνος σχεδίασης θα ήταν το πολύ 2-5 sec. Ο χρόνος σχεδίασης του ίδιου γράφου από τον αλγόριθμο που περιγράφουμε σ'αυτό το κεφάλαιο είναι μόλις 3.22 sec δηλαδή περίπου το ένα δέκατο του χρόνου του STT. Ακόμα, ο STT δημιουργεί αρκετές ανεπιθύμητες κάμψεις ακμών οι οποίες μειώνουν την ποιότητα του σχεδίου. Στη σχεδίαση του αλγόριθμού μας δόθηκε μεγάλη έμφαση στην επίτευξη μικρού χρόνου σχεδίασης. Επίσης, όπως θα φανεί από τις μετρήσεις και τη σύγκριση με τον αλγόριθμο STT στο επόμενο κεφάλαιο, ο αλγόριθμος μας έχει ικανοποιητικά αποτελέσματα και ως προς άλλα κριτήρια αξιολόγησης, όπως τομές ακμών, συνολικό μήκος ακμών, κ.α.

Όσο μεγαλώνει το μέγεθος του γράφου τόσο μεγαλώνει ο αριθμός των κόμβων ανά επίπεδο και ο αριθμός των επιπέδων, και επομένως τόσο περισσότερες είναι οι δυνατές μεταθέσεις αυτών των κόμβων, των οποίων ο αριθμός μεγαλώνει εκθετικά. Γι' αυτό οποιαδήποτε ευρηματική μέθοδος αποφασίσει να ακολουθήσει τη στρατηγική της ταξινόμησης κόμβων, θα είναι υποχρεωμένη είτε να αφιερώσει πάρα πολύ χρόνο για την εύρεση τοπικών ελαχίστων

ή θα πρέπει να είναι αρκετά ‘‘έξυπνη’’ ώστε να βρεί μία αρκετά καλή τοπικά βέλτιστη λύση του χώρου έρευνας ψάχνοντας όμως ένα μικρό μέρος του, διαφορετικά θα αντιμετωπίσει τα ίδια προβλήματα με τον αλγόριθμο STT. Για την αποφυγή παρόμοιων προβλημάτων, σ’αυτή την εργασία, προσεγγίζουμε το πρόβλημα από μία εντελώς διαφορετική σκοπιά. Το υπόλοιπο αυτής της παραγράφου περιγράφει σε γενικές γραμμές τον αλγόριθμο που σχεδιάστηκε σε αυτή την εργασία.

Ο αλγόριθμος δέχεται ως είσοδο μία περιγραφή του γράφου η οποία αποτελείται από τη λίστα των ακμών του γράφου. Για κάθε γράφο που του δίδεται ως είσοδος, ο αλγόριθμος δημιουργεί ένα σχέδιο στο οποίο ο προσανατολισμός των ακμών είναι από αριστερά προς τα δεξιά (η ρίζα του γράφου δηλαδή δε σχεδιάζεται στην κορυφή του σχεδίου αλλά στο αριστερότερο σημείο του). Είναι περισσότερο συνηθισμένο όταν σχεδιάζονται δέντρα ή και προσανατολισμένοι γράφοι με τη μορφή ιεραρχίας, να σχεδιάζονται με φορά ακμών από πάνω προς τα κάτω. Τα σχήματα που θα δωθούν στη συνέχεια σε αυτό το κεφάλαιο ακολουθούν αυτή τη σύμβαση, όχι όμως και τα σχήματα που παράγει ο αλγόριθμος. Όπως έχουμε αναφέρει, ο αλγόριθμος αυτός χρησιμοποιείται στα πλαίσια της επαφής χρήσης ενός πληροφοριακού συστήματος. Γενικά τα ονόματα των κόμβων γράφων σε τέτοιες περιπτώσεις τείνουν να είναι ορμαθοί χαρακτήρων με αρκετά μεγάλο μέγεθος και δεν είναι δόκιμο να αποκόπτουμε κάποιο τμήμα τους έτσι ώστε όλα τα ονόματα κόμβων να αποκτήσουν μικρότερο μήκος από κάποιο ανώτατο όριο. Επίσης δεν υπάρχει περιορισμός ως προς τον αριθμό των παιδιών που μπορεί να έχει ένας κόμβος. Έτσι αν επιλέγαμε τον σχεδιασμό με φορά ακμών από πάνω προς τα κάτω ίσως το πλάτος μίας οθόνης υπολογιστή να μην ήταν αρκετό για να σχεδιαστούν τα παιδιά ενός κόμβου.

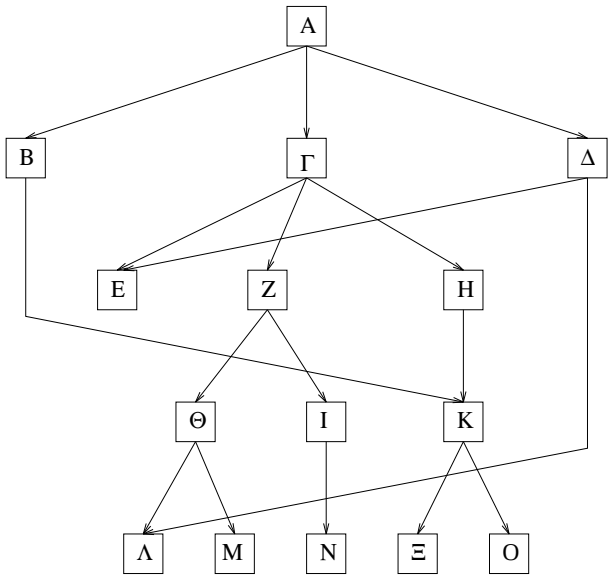
Η αναδιάταξη των κόμβων από τον αλγόριθμο δεν γίνεται έχοντας υπόψη τη δομή του γράφου αλλά ένα κατάλληλο *ζευγνύον δέντρο* του γράφου (spanning tree) (δηλαδή ένα δέντρο που περιέχει όλους τους κόμβους του γράφου και κάποιες από τις ακμές του έτσι ώστε στο γράφημα που προκύπτει να υπάρχει μονοπάτι από ένα κόμβο σε κάθε άλλο). Οι ακμές του γράφου που περιέχονται στο δέντρο ονομάζονται *ακμές δέντρου* ενώ αυτές που δεν περιέχονται *ακμές γράφου*.

Πριν αρχίσει η εκτέλεση του αλγορίθμου, πρέπει να προηγηθεί ένα στάδιο προεπεξεργασίας του γράφου. Στο στάδιο αυτό, ο γράφος μετατρέπεται σε έναν άκυκλο γράφο (στη περίπτωση που είχε κύκλους), υπολογίζονται τα επίπεδα των κόμβων, επιλέγεται το ζευγνύον δέντρο κ.α. Η παράγραφος 3.2 περιγράφει με περισσότερες λεπτομέρειες αυτό το στάδιο.

Οι ακμές γράφου είναι σημαντικές όσον αφορά το πρόβλημα της σχεδίασης του γράφου. Αν αυτές δεν υπήρχαν τότε ο γράφος θα ήταν δέντρο και ο σχεδιασμός του θα ήταν απλός (θα χρησιμοποιούσαμε έναν από τους υπάρχοντες αλγόριθμους σχεδιασμού δέντρων). Επιπλέον,

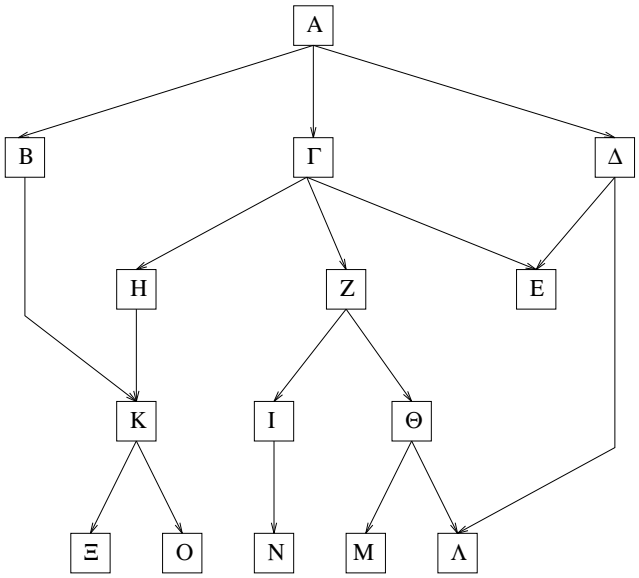
όσο μεγαλώνει ο αριθμός των ακμών γράφου τόσο πιο πολύπλοκος γίνεται ο γράφος (περιέχει περισσότερες τομές ακμών και γενικά το σχέδιό του γίνεται περισσότερο μπερδεμένο). Οι ακμές γράφου όμως, δεν θα αποτελούσαν σημαντικό πρόβλημα στη σχεδίαση αν είχαμε κάποιο τρόπο να περιορίζαμε την απόσταση που διασχίζουν στο τελικό σχέδιο. Έτσι, θα τέμνονταν με λιγότερες ακμές (αν υποθέσουμε ομοιόμορφη διάταξη των κόμβων στο επίπεδο) και το σχέδιο θα ήταν πιο καθαρό. Παρατηρήσαμε [10] ότι με κατάλληλη αναδιάταξη των κόμβων ενός γράφου, είναι δυνατόν ορισμένες φορές να πετύχουμε τον παραπάνω στόχο. Ακόμα όμως και αν δεν τον πετύχουμε εντελώς, η αναδιάταξη των κόμβων επιφέρει ούτως ή άλλως σημαντική βελτίωση στην εμφάνιση του γράφου.

Στο σχήμα 3.1 δίνουμε ένα παράδειγμα γράφου με απλή σχετικά δομή - έχει μόνο τρεις



Σχήμα 3.1: Σχέδιο ενός προσανατολισμένου γράφου πριν την αναδιάταξη των κόμβων

ακμές γράφου, τις: (Β, Κ), (Δ, Ε) και (Δ, Λ). Με τη πρώτη ματιά είναι φανερό ότι κάτι δεν πάει καλά. Γιατί τα Β και Κ είναι τόσο μακριά και γιατί τα Δ και Λ είναι τοποθετημένα σχεδόν διαμετρικά αντίθετα; Δημιουργείται λοιπόν το ερώτημα αν θα μπορούσε κάποια αναδιάταξη των κόμβων (η οποία θα φέρει τους κόμβους που αποτελούν άκρα των ακμών γράφου πιο κοντά) να βελτιώσει το σχήμα. Το σχήμα 3.1 έχει 8 τομές ακμών. Ο ίδιος γράφος μπορεί όντως να σχεδιαστεί με λιγότερες τομές ακμών και επομένως καλύτερα. Ένα σχήμα του ίδιου γράφου χωρίς καθόλου τομές ακμών φαίνεται στο σχήμα 3.2. Για τη βελτίωση του σχήματος χρειάστηκε να αναδιαταχθούν τα παιδιά του κόμβου Γ, δηλαδή οι κόμβοι Ε, Ζ, και Η. Επίσης, τα παιδιά του κόμβου Ζ (δηλαδή οι Θ και Ι) και τα παιδιά του κόμβου



Σχήμα 3.2: Ο γράφος του σχήματος 3.1 μετά την αναδιάταξη των κόμβων

Θ (δηλαδή οι Λ και Μ). Στη περίπτωση του σχήματος 3.1, η αναδιάταξη κόμβων είχε τελικά καλά αποτελέσματα. Αυτό δεν είναι πάντα δυνατό. Δεν μπορεί πάντα να έχουμε τα ίδια καλά αποτελέσματα, για δύο κυρίως λόγους. Ο πρώτος είναι ότι υπάρχει ένας αριθμός τομών ακμών σε κάθε σχήμα ο οποίος δεν μπορεί να εξαληφθεί, και ο δεύτερος έχει να κάνει με τη πολυπλοκότητα του γράφου. Ένας τρόπος να εκφράσουμε την πολυπλοκότητα είναι μέσω της πυκνότητας του γράφου. Στο παράδειγμα που δόθηκε προηγουμένως, κάθε ακμή γράφου έθετε μία απαίτηση για μια σειρά από αναδιάτάξεις κόμβων. Όσο αυξάνει η πυκνότητα ενός γράφου μεγαλώνει αντίστοιχα και ο αριθμός των ακμών γράφου, άρα και η πιθανότητα οι απαιτήσεις που τίθενται από τις ακμές γράφου να συγκρούονται μεταξύ τους (για παράδειγμα μία ακμή γράφου να απαιτεί να αναδιαταχθούν οι κόμβοι του γράφου έτσι ώστε ένα μονοπάτι κόμβων να βρεθεί στο αριστερό σύνορο του υποδέντρου που το περιέχει ενώ κάποια άλλη ακμή να απαιτεί το ίδιο μονοπάτι να βρεθεί στο δεξί σύνορο). Πάντως, αν ικανοποιηθεί το μεγαλύτερο μέρος από τις απαιτήσεις που θέτουν οι ακμές γράφου για αναδιάταξη, μπορούμε να περιμένουμε βελτίωση του σχήματος.

Επικεντρώνοντας λίγο περισσότερο τη προσοχή μας στις ακμές γράφου, μπορούμε να εξάγουμε και άλλα χρήσιμα συμπεράσματα μέσα από τα οποία αρχίζει να διαφαίνεται διαισθητικά η προσέγγισή μας στη λύση του προβλήματος. Κατ'αρχήν, για κάθε ακμή γράφου ορίζουμε δύο μονοπάτια κόμβων. Τα δύο αυτά μονοπάτια έχουν κοινό τέλος τον προορισμό της ακμής γράφου και κοινή αρχή τον κοινό πρόγονο της αφετηρίας και του προορισμού της ακμής γράφου (θεωρώντας το ζευγνύον δέντρο για την εύρεση των

προγόνων). Παραδείγματος χάριν, στο σχήμα 3.1, για την ακμή (Δ, Λ) ορίζονται τα μονοπάτια Α, Γ, Ζ, Θ, Λ και Α, Δ, Λ. Όμοια, για την (Β, Κ) ορίζονται τα μονοπάτια Α, Γ, Η, Κ και Α, Β, Κ και για την (Δ, Ε) τα Α, Γ, Ε και Α, Δ, Ε.

Δεν είναι τυχαίο το γεγονός ότι τα μονοπάτια αυτά, στο τελικό σχήμα, βρέθηκαν εξ'ολοκλήρου στα σύνορα των υποδέντρων που τα περιέχουν, ή πολύ κοντά σ'αυτά. Στο σχήμα 3.2, με εξαίρεση τον κόμβο Ε, το μονοπάτι Γ, Ζ, Θ, Λ συμπίπτει με το δεξί σύνορο του υποδέντρου με ρίζα το Γ, επομένως αυτό το μονοπάτι έχει μετακινηθεί κοντά στο δεξί σύνορο. Αν αυτό δεν συνέβαινε, τότε η ακμή (Δ, Λ) θα δημιουργούσε αναγκαστικά κάποιες τομές ακμών. Αυτό που επιχειρεί να καταφέρει ο αλγόριθμος είναι να φέρει ''κοντά'' τα δύο χαρακτηριστικά μονοπάτια κάθε μακριάς ακμής. Θεωρώντας ένα σχέδιο του γράφου το ένα μονοπάτι βρίσκεται δεξιά του άλλου. Μία μείωση στην απόσταση μεταξύ των δύο μονοπατιών ισοδυναμεί με πιθανή ελάττωση των τομών ακμών, όπως διαισθητικά φάνηκε από τα παραπάνω. Η μέγιστη μείωση της απόστασης μεταξύ των δύο μονοπατιών αντιστοιχεί στην περίπτωση κατά την οποία το μονοπάτι που βρίσκεται στα αριστερά μετακινείται στο δεξί σύνορο του υποδέντρου που το περιέχει και το μονοπάτι που βρίσκεται στα δεξιά μετακινείται στο αριστερό σύνορο του υποδέντρου που το περιέχει.

Στην παράγραφο 3.3 περιγράφεται η πρώτη φάση του αλγορίθμου. Σ'αυτήν τη φάση συγκεντρώνονται πληροφορίες οι οποίες αφορούν τα μονοπάτια της μορφής που περιγράψαμε προηγουμένως, καθώς και άλλες, οι οποίες χρησιμοποιούνται κατά τη δεύτερη φάση του αλγορίθμου. Η παράγραφος 3.4 αναφέρεται στη δεύτερη φάση, η οποία αναδιατάσσει τους κόμβους του ζευγνύοντος δέντρου φροντίζοντας να ικανοποιηθούν οι απαιτήσεις που τέθηκαν από τις ακμές γράφου σχετικά με τη τοποθέτηση των χαρακτηριστικών τους μονοπατιών.

Μετά το τέλος της δεύτερης φάσης, μένει να εισαχθούν στο γράφο οι τεχνητοί κόμβοι που αντιστοιχούν σε κάθε μακριά ακμή, και να υπολογιστούν και οι τελικές συντεταγμένες των κόμβων. Οι τεχνητοί κόμβοι δεν εισάγονται στη δομή του γράφου, αλλά στη δομή του δέντρου στο οποίο δουλεύαμε μέχρι και το τέλος της δεύτερης φάσης. Για κάθε ακμή γράφου έχουμε ήδη αναφέρει ότι χρειάζεται να εισάγουμε $l - 1$ τεχνητούς κόμβους όπου l είναι το μήκος της. Έτσι, επεκτείνουμε το δέντρο προσθέτοντας για κάθε αρχική μακριά ακμή γράφου e_i με μήκος l_{e_i}, τα $l_{e_i} - 1$ τμήματα που της αντιστοιχούν. Είναι προφανές ότι δεν εισάγουμε στο δέντρο το τελευταίο τμήμα που αντιστοιχεί σε κάθε ακμή γράφου, γιατί τότε δε θα ήταν πλέον δέντρο. Για τη τοποθέτηση του πρώτου τεχνητού κόμβου για κάθε ακμή γράφου χρειάζεται κάποια ειδική διαδικασία προκειμένου να αποφασιστεί σε ποια θέση μεταξύ των αδελφιών του θα τοποθετηθεί. Η τοποθέτηση των υπολοίπων τεχνητών κόμβων είναι τετριμμένη. Στη παράγραφο 3.5 δίνονται περισσότερες λεπτομέρειες σχετικά με την

εισαγωγή των τεχνητών κόμβων.

Εφόσον έχουν εισαχθεί όλοι οι τεχνητοί κόμβοι στο δέντρο, είναι γνωστές οι σχετικές θέσεις των κόμβων ανά επίπεδο όχι όμως και οι συντεταγμένες τους. Σ'αυτή τη φάση μπορούμε να υπολογίσουμε τις συντεταγμένες των κόμβων χρησιμοποιώντας έναν αλγόριθμο σχεδιασμού δέντρων. Επιλέξαμε τον αλγόριθμο του S. Moen (περιγράφεται στο παράρτημα Α) για αυτό το σκοπό. Η διαδικασία της εισαγωγής των τεχνητών κόμβων στο δέντρο και η χρησιμοποίηση ενός αλγορίθμου σχεδιασμού δέντρων για την εύρεση των συντεταγμένων των κόμβων του γράφου βοηθάει στην ελλάτωση των κάμψεων των ακμών. Συγκεκριμένα δημιουργούνται το πολύ 2 κάμψεις ανά μακριά ακμή γράφου. Είναι δυνατόν όμως να αυξηθεί το συνολικό πλάτος του σχεδίου (ή το ύψος του σχεδίου αν η φορά των ακμών είναι από αριστερά προς τα δεξιά). Αφού τοποθετηθούν οι κόμβοι στις συντεταγμένες που υπολογίστηκαν και δρομολογηθούν οι ακμές του δέντρου, στη συνέχεια εισάγουμε για κάθε αρχική ακμή γράφου το τελευταίο τμήμα που της αντιστοιχεί - αυτό που θα δημιουργήσει τελικά τις τομές μεταξύ ακμών. Σ'αυτό το σημείο έχει ολοκληρωθεί η σχεδίαση του γράφου.

3.2 Προεπεξεργασία του Γράφου

3.2.1 Χειρισμός των κύκλων

Ο αλγόριθμος χρειάζεται έναν άκυκλο προσανατολισμένο γράφο ως είσοδο. Στη περίπτωση που ο γράφος που πρόκειται να σχεδιαστεί έχει κύκλους εφαρμόζουμε την ίδια τεχνική με προηγούμενες εργασίες. Κατά τη διάρκεια μιας κατά-βάθος-πρώτα διάσχισης του γράφου, αν βρεθεί ότι κάποια ακμή κλείνει έναν κύκλο, τότε η φορά της αντιστρέφεται. Όταν ο γράφος έχει σχεδιαστεί, η φορά των ακμών που είχαν αρχικά αντιστραφεί, αντιστρέφεται ξανά, έτσι ώστε να αποκτήσουν την πραγματική τους φορά.

Η παραπάνω μέθοδος δεν εγγυάται ότι θα βρει το ελάχιστο σύνολο των ακμών οι οποίες πρέπει να αλλάξουν φορά προκειμένου να γίνει ο γράφος άκυκλος. Ο Sugiyama σε μια εργασία του '82 περιγράφει πως είναι δυνατόν να υπολογιστεί το ελάχιστο σύνολο τέτοιων ακμών, αλλά κρίναμε ως μη σημαντική την υλοποίηση αυτής της μεθόδου αυτής.

3.2.2 Τοποθέτηση κόμβων σε επίπεδα

Για την τοποθέτηση των κόμβων σε επίπεδα κάνουμε μία τοπολογική ταξινόμηση των κόμβων του γράφου. Αρχικά οι κόμβοι οι οποίοι δεν έχουν προηγούμενους κόμβους (οι ρίζες του γράφου) τοποθετούνται στο πρώτο επίπεδο, δηλαδή στο επίπεδο 0. Αν έχουμε παραπάνω από μία ρίζες δημιουργούμε μία τεχνητή ρίζα η οποία γίνεται προηγούμενος κόμβος όλων

των ριζών που βρέθηκαν. Σε κάθε ένα από τα επόμενα επίπεδα, τοποθετούμε μόνο κόμβους των οποίων έχουν ήδη τοποθετηθεί όλοι οι προηγούμενοί τους κόμβοι.

Η τοποθέτηση των κόμβων με βάση τοπολογική ταξινόμηση έχει αποτέλεσμα να δείχνουν όλες οι ακμές του γράφου προς την ίδια κατεύθυνση, αλλά δεν κάνει τη βέλτιστη τοποθέτηση κόμβων, εφόσον τείνει να δημιουργεί μεγαλύτερο αριθμό επιπέδων κόμβων από το ελάχιστο δυνατόν, το οποίο έχει σαν συνέπεια την αύξηση του συνολικού μήκους ακμών. Οι Gansner et. al. στο [16] χρησιμοποιούν μια διαφορετική τεχνική για την τοποθέτηση κόμβων σε επίπεδα, η οποία προσπαθεί να μειώσει το συνολικό μήκος των ακμών του γράφου. Σ'αυτή την εργασία το πρόβλημα της τοποθέτησης των κόμβων αντιμετωπίζεται σαν ένα πρόβλημα γραμμικού προγραμματισμού στο οποίο ελαχιστοποιείται η αντικειμενική συνάρτηση που εκφράζει το συνολικό μήκος ακμών. Για να δείξουν τα αποτελέσματα της μεθόδου τους στο σχεδιασμό γράφων, παρουσιάζουν ένα παράδειγμα γράφου με 43 κόμβους και 64 ακμές το οποίο χρησιμοποιώντας την μέθοδο τους χρειάζεται 1.63 sec για να σχεδιαστεί και έχει 20 τομές ακμών ενώ χρησιμοποιώντας τη μέθοδο της τοπολογικής ταξινόμησης για την τοποθέτηση των κόμβων χρειάζεται 3.68 sec για να σχεδιαστεί και έχει 44 τομές ακμών.

Η μέθοδος των Gansner et. al. εγγυάται την τοποθέτηση των κόμβων σε ελάχιστο αριθμό επιπέδων αλλά δεν εξασφαλίζει ίδια κατεύθυνση σε όλες τις ακμές γράφου. Είναι φανερό ότι η επιλογή της μεθόδου που χρησιμοποιείται για την τοποθέτηση των κόμβων σε επίπεδα, εξαρτάται από το γράφο που σχεδιάζεται και τις απαιτήσεις του χρήστη σχετικά με τη φορά των ακμών. Στη δική μας εφαρμογή θέλουμε όλες τις ακμές προς την ίδια κατεύθυνση, και γι'αυτό χρησιμοποιούμε τη μέθοδο της τοπολογικής ταξινόμησης.

3.2.3 Τοποθέτηση των ριζών στο μεγαλύτερο δυνατόν επίπεδο

Κατά την τοποθέτηση των κόμβων σε επίπεδα, οι ρίζες του γράφου τοποθετήθηκαν στο επίπεδο 0. Για μερικές από αυτές τις ρίζες είναι δυνατόν όλοι οι απόγονοί τους να τοποθετήθηκαν σε επίπεδα αρκετά μετά το επίπεδο 0. Αν θεωρήσουμε μια τέτοια ρίζα r, όπου το ελάχιστο επίπεδο στο οποίο έχει τοποθετηθεί απόγονός της είναι το επίπεδο min_l_r, τότε αυτή η ρίζα μεταφέρεται στο επίπεδο $min_l_r - 1$. Έτσι, εξασφαλίζουμε ότι κάθε ρίζα βρίσκεται ένα επίπεδο πάνω από τον απόγονό της με το μικρότερο επίπεδο και όχι πολύ μακριά από αυτόν. Παρόμοια μετακίνηση μπορεί να προκληθεί και σε ρίζες υποδέντρων λόγω της αρχικής μετακίνησης των ριζών.

Η εύρεση του μέγιστου επιπέδου στο οποίο μπορεί να τοποθετηθεί κάποια ρίζα γίνεται σε δύο φάσεις:

α) *Εύρεση των κόμβων οι οποίοι δεν θα μετακινηθούν σε μεγαλύτερο επίπεδο.* Προφανώς οι κόμβοι οι οποίοι τοποθετήθηκαν από την τοπολογική ταξινόμηση προηγουμένως στο

τελευταίο επίπεδο, δεν μπορούν να μετακινηθούν σε μεγαλύτερο επίπεδο. Έστω ότι αυτοί οι κόμβοι αποτελούν το σύνολο S. Για τους κόμβους του συνόλου S σημειώνεται ότι δεν θα μετακινηθούν σε μεγαλύτερο επίπεδο. Το νέο σύνολο S αποτελείται από τους προηγούμενους κόμβους των κόμβων του συνόλου οι οποίοι βρίσκονται ακριβώς ένα επίπεδο πιο πάνω. Επαναλαμβάνουμε την ίδια διαδικασία για τους κόμβους του νέου συνόλου, έως ότου πάρουμε ένα σύνολο το οποίο είναι κενό. Κάθε φορά που σημειώνουμε για κάποιο κόμβο v του συνόλου S, ότι δεν θα μετακινηθεί σε μεγαλύτερο επίπεδο, αναδρομικά σημειώνουμε το ίδιο για κάθε απόγονό του ο οποίος έχει διαφορά επιπέδου τόση όση και το μήκος του μονοπατιού μέσω του οποίου φτάσαμε σ'αυτόν από τον v.

β) *Εύρεση του μέγιστου επιπέδου στο οποίο μπορεί να τοποθετηθεί κάθε ρίζα.* Η προηγούμενη φάση αποκλείει τη μετακίνηση κάποιων από τις ρίζες σε μεγαλύτερο επίπεδο και το επιτρέπει σε κάποιες άλλες. Για κάθε μία από τις τελευταίες, βρίσκεται αναδρομικά το μέγιστο επίπεδο στο οποίο μπορούν να τοποθετηθούν οι απόγονοί της και τοποθετείται ένα επίπεδο πιο πάνω. Η αναδρομή διακόπτεται σε κόμβους για τους οποίους έχει σημειωθεί ότι δεν θα μετακινηθούν σε μεγαλύτερο επίπεδο και αυτοί οι κόμβοι συνεισφέρουν στην εύρεση του μέγιστου επιπέδου, το επίπεδο στο οποίο βρίσκονται τοποθετημένοι.

3.2.4 Κατασκευή του ζευγνύοντος δέντρου

Η κατασκευή του ζευγνύοντος δέντρου μπορεί να γίνει με μία κατά-πλάτος-πρώτα (BFS) διάσχιση του γράφου. Κατά τη διάσχιση αυτή δημιουργούνται οι δείκτες που δείχνουν στον πατέρα, το παιδί και τον αδελφό κάθε κόμβου, και επιλέγονται οι ακμές του γράφου οι οποίες θα ανήκουν στο δέντρο και αυτές οι οποίες δεν θα ανήκουν. Πειραματικά αποτελέσματα έδειξαν ότι το σύνολο των ακμών που επιλέγεται να ανήκει στο δέντρο (ή αντίστοιχα το σύνολο των ακμών που επιλέγεται να είναι το σύνολο των ακμών γράφου) επηρεάζει το τελικό αποτέλεσμα, αφού συνήθως η τυχαία επιλογή οδηγεί σε σχέδια με περισσότερες τομές ακμών.

Η κατασκευή ενός ζευγνύοντος δέντρου που θα επιτρέπει στον αλγόριθμο σχεδιασμού του γράφου να έχει καλά αποτελέσματα δεν είναι τετριμμένη και αξίζει να αποτελέσει αντικείμενο μελλοντικής έρευνας. Στα πλαίσια αυτής της εργασίας δημιουργούμε το ζευγνύον δέντρο κάνοντας μία κατά πλάτος πρώτα διάσχιση του γράφου όπου κάθε κόμβος επιλέγει τα παιδιά του (άρα και τις ακμές δέντρου) χρησιμοποιώντας μια διαδικασία επιλογής η οποία περιγράφεται παρακάτω. Πειραματικά αποτελέσματα έδειξαν ότι στο μεγαλύτερο ποσοστό των παραδειγμάτων η επιλογή των ακμών δέντρου με βάση αυτό τον κανόνα είχε καλύτερα αποτελέσματα έναντι της τυχαίας επιλογής.

Η διαδικασία επιλογής στην οποία αναφερθήκαμε προηγουμένως δεν επιτρέπει σε κάποιο

κόμβο να επιλέξει για παιδιά του πολλούς κόμβους οι οποίοι έχουν έσω-βαθμό (fan-in) μεγαλύτερο από 1. Έτσι, επιτρέπει σε κάθε κόμβο να έχει το πολύ δύο παιδιά με έσω-βαθμό μεγαλύτερο από 1. Κάθε κόμβος επιλέγει παιδιά από το σύνολο των επόμενών του κόμβων, οι οποίοι έχουν τοποθετηθεί στο επόμενο επίπεδο από αυτόν. Αν ένας κόμβος v έχει ήδη δύο παιδιά με έσω-βαθμό μεγαλύτερο από 1, μπορεί να απορρίψει για υποψήφιο παιδί του ένα τρίτο επόμενο του κόμβο w με έσω-βαθμό μεγαλύτερο από 1, υπό την προϋπόθεση ότι υπάρχει κάποιος άλλος κόμβος u προηγούμενος του w στο ίδιο επίπεδο με τον v, που μπορεί να θεωρήσει τον u παιδί του. Αν αυτό δεν είναι δυνατό, τότε αναγκαστικά ο v αποκτάει τον w σαν παιδί. Στη περίπτωση που τα δύο παιδιά με έσω-βαθμό μεγαλύτερο από 1, που επέλεξε κάποιος κόμβος τυχαίνει να έχουν αρκετά μεγάλο έσω-βαθμό (μεγαλύτερο από κάποια σταθερά η οποία πειραματικά επιλέχθηκε να είναι 5), τότε επιτρέπουμε να κρατήσει μόνο το ένα από αυτά. Για το παιδί, έστω c, που δεν μπορεί να κρατήσει αυτός ο κόμβος, προσπαθούμε να βρούμε κάποιον από τους προηγούμενους κόμβους του c, οι οποίοι βρίσκονται ένα ή δύο επίπεδα πιο πάνω από τον c και δεν είναι βεβαρυμένοι όσον αφορά το αριθμό παιδιών με έσω-βαθμό μεγαλύτερο από 1. Αν βρούμε κάποιο τέτοιο κόμβο τότε κάνουμε τον c παιδί του, διαφορετικά αφήνουμε τον c στον παλιό του πατέρα.

Όσον αφορά τις ρίζες του γράφου έχουμε δύο εναλλακτικές λύσεις ως προς το ποιον θα θεωρήσουμε πατέρα τους. Η προφανής λύση είναι να δημιουργήσουμε ένα μονοπάτι από τεχνητούς κόμβους του οποίου το μήκος εξαρτάται από το επίπεδο στο οποίο μετακινήσαμε τη συγκεκριμένη ρίζα και να θεωρήσουμε σαν πατέρα της ρίζας τον τελευταίο τεχνητό κόμβο του μονοπατιού. Η δημιουργία αυτών των τεχνητών μονοπατιών (αν και δεν σχεδιάζονται στο τελικό σχέδιο) κάνουν περισσότερο εμφανή τη θέση της ρίζας μέσα στο σύνολο των κόμβων του γράφου. Το μειονέκτημα είναι ότι αυξάνουν το χρόνο εκτέλεσης του αλγορίθμου ο οποίος επηρεάζεται ορισμένες φορές από το αριθμό των παιδιών της ρίζας του γράφου.

Προκειμένου να αποφευχθεί αυτό το επιπλέον κόστος σε χρόνο ακολουθήθηκε μια διαφορετική μέθοδος ως προς την επιλογή του πατέρα των ριζών που μετακινούνται σε μεγαλύτερο επίπεδο. Για κάθε ρίζα r που μετακινούμε στο επίπεδο $max\lrcorner_r$, προσπαθούμε να βρούμε έναν κόμβο v στο επίπεδο $max\lrcorner_r - 1$ ο οποίος σχετίζεται σε μεγαλύτερο βαθμό από ότι οι άλλοι κόμβοι του επιπέδου του με τη ρίζα r και κάνουμε την r παιδί του v. Ο βαθμός συσχέτισης εξαρτάται από τον αριθμό των ακμών γράφου που ξεκινούν από το υποδέντρο S_v και τελειώνουν στο υποδέντρο S_r, ή το αντίστροφο. Το υποδέντρο S_v είναι το υποδέντρο με ρίζα τον κόμβο v και το S_r το υποδέντρο με ρίζα τον κόμβο r.

3.3 Συγκέντρωση Πληροφορίας κατά την Πρώτη Φάση

Στην παράγραφο αυτή περιγράφουμε τις ενέργειες που γίνονται κατά τη πρώτη φάση του αλγορίθμου. Σ'αυτή τη φάση συγκεντρώνεται πληροφορία η οποία είναι σχετική με τις ακμές γράφου και τα χαρακτηριστικά μονοπάτια κόμβων τα οποία συσχετίζονται μέσω αυτής της ακμής γράφου. Ο τρόπος με τον οποίο αξιοποιείται η πληροφορία αυτή από τον αλγόριθμο περιγράφεται στην επόμενη παράγραφο.

Έχουμε ήδη αναφέρει ότι οι ακμές γράφου είναι σημαντικές στα πλαίσια της πρώτης φάσης. Οι ακμές ενός προσανατολισμένου γράφου χωρίζονται σε 4 κατηγορίες:

α) *ακμές δέντρου* (tree edges), είναι όλες οι ακμές του γράφου που ανήκουν στο επιλεγέν ζευγνύον δέντρο του,

β) *ακμές κύκλων* (back edges), είναι οι ακμές οι οποίες κλείνουν κάποιο κύκλο,

γ) *πρόσθιες ακμές* (forward edges), είναι οι ακμές e = (v, w) όπου ο v είναι πρόγονος του w στο επιλεγέν ζευγνύον δέντρο αλλά όχι προηγούμενός του κόμβος, και

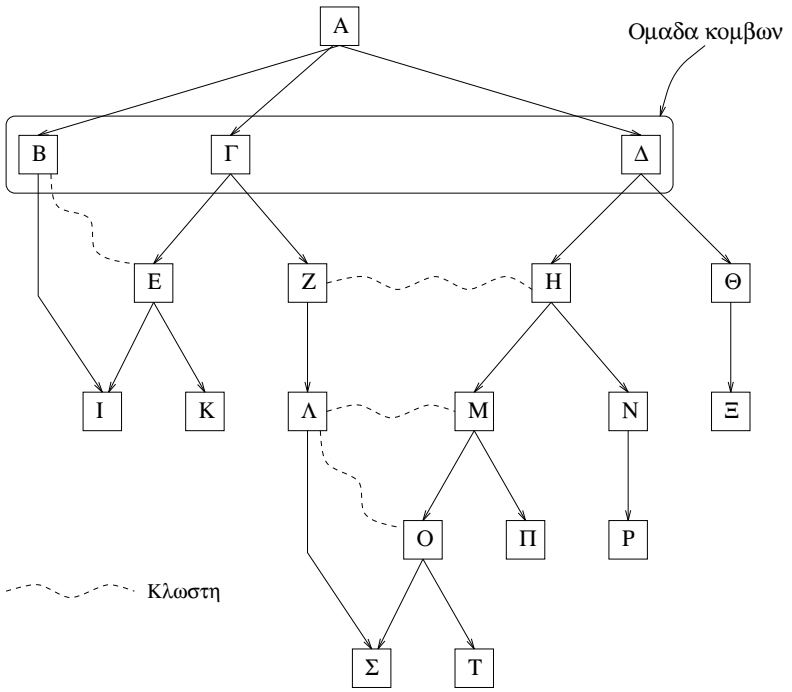
δ) *ακμές διασταύρσης* (cross edges), είναι οι ακμές e = (v, w) όπου ο v και ο w δεν έχουν μεταξύ τους σχέση προγόνου-απογόνου.

Οι ακμές γράφου που θα μας απασχολήσουν εδώ, είναι είτε πρόσθιες ακμές είτε ακμές διασταύρωσης. Οι ακμές κύκλων έχουν αντιστραφεί κατά τη διάρκεια του χειρισμού των κύκλων και έχουν μετατραπεί σε πρόσθιες ακμές ή ακμές διασταύρωσης.

Στην εισαγωγή αναφερθήκαμε στα δύο μονοπάτια κόμβων που σχετίζονται με κάθε ακμή γράφου. Στην περίπτωση των πρόσθιων ακμών το ένα μονοπάτι από τα δύο είναι τετριμμένο αφού περιέχει μονάχα τους 2 κόμβους που αποτελούν την αφετηρία και προορισμό της ακμής γράφου. Η οργάνωση του προβλήματος μας απαιτεί και τα δύο μονοπάτια να είναι μη τετριμμένα. Γι'αυτό, εισάγουμε σε κάθε πρόσθια ακμή τον πρώτο από τους τεχνητούς κόμβους που της αντιστοιχεί (μια πρόσθια ακμή έχει πάντα μήκος μεγαλύτερο ή ίσο του 2 σύμφωνα με τον ορισμό). Έτσι, ο τεχνητός κόμβος θα αποτελέσει τμήμα του μονοπατιού που περιείχε προηγουμένως την αφετηρία και το προορισμό της πρόσθιας ακμής γράφου.

3.3.1 Ακμές γράφου και μονοπάτια κόμβων

Στη παράγραφο αυτή θα προσπαθήσουμε να περιγράψουμε με τυπικό τρόπο τα μονοπάτια κόμβων που σχετίζονται με κάθε ακμή γράφου. Τα μονοπάτια αυτά τα ονομάζουμε *χαρακτηριστικά μονοπάτια*. Τα χαρακτηριστικά μονοπάτια P_1 και P_2 για κάθε ακμή γράφου $e = (s, t)$ έχουν την εξής μορφή:



Σχήμα 3.3: Κλωστές μεταξύ χαρακτηριστικών μονοπατιών ακμών

$$P_1 = (A, v_1, \ldots, v_k, t) \quad P_2 = (A, w_1, \ldots, w_l, t) \quad \text{όπου} \begin{cases} k \leq l,\ v_k = s, \\ w_l \text{ είναι ο πατέρας του } t \text{ στο ζευγνύον δέντρο, και} \\ \text{A είναι ο κοινός πρόγονος των } v_k \text{ και } w_l \end{cases}$$

Το P_1 δηλαδή, είναι το μονοπάτι που περιέχει την αφετηρία και τον προορισμό της e.

Υπάρχουν δύο τύποι τέτοιων μονοπατιών. Για το πρώτο τύπο χαρακτηριστικών μονοπατιών ισχύει $k = 1$, ενώ για το δεύτερο τύπο $k > 1$. Η διαφορά δηλαδή μεταξύ των δύο τύπων χαρακτηριστικών μονοπατιών είναι ότι το μονοπάτι P_1 στον πρώτο τύπο έχει μόνο τρεις κόμβους ενώ στο δεύτερο τύπο περισσότερους. Το σχήμα 3.3 έχει ειδικά σχεδιαστεί ώστε να περιέχει μονοπάτια και των δύο τύπων και θα χρησιμοποιηθεί στη συνέχεια για να επεξηγηθούν κάποιες έννοιες που παρουσιάζονται σ'αυτήν και επόμενες παραγράφους. Το ζευγάρι μονοπατιών του πρώτου τύπου είναι στο σχήμα 3.3 τα (Α, Β, Ι), (Α, Γ, Ε, Ι), ενώ του δεύτερου τύπου τα (Α, Γ, Ζ, Λ, Σ), (Α, Δ, Η, Μ, Ο, Σ).

Στόχος του αλγορίθμου είναι να μειώσει την απόσταση στο τελικό σχέδιο μεταξύ των χαρακτηριστικών μονοπατιών P_1 και P_2 κάθε ακμής γράφου. Προκειμένου να γίνει αυτό, θα πρέπει να έχουμε τη δυνατότητα να συσχετίζουμε τα δύο μονοπάτια. Όπως έχουμε ήδη αναφέρει στην παράγραφο 3.1, οι απαιτήσεις που θέτουν οι ακμές γράφου σχετικά με τα χαρακτηριστικά μονοπάτια τους, είναι ορισμένες φορές συγκρουόμενες

και επομένως ο αλγόριθμος καταφέρνει να πληρεί τελικά μέρος αυτών των απαιτήσεων, δηλαδή να μετακινεί κοντά σε σύνορα υποδέντρων τμήματα μόνο των χαρακτηριστικών μονοπατιών. Γι'αυτό επιλέγουμε να συσχετίσουμε τους κόμβους των χαρακτηριστικών μονοπατιών μεταξύ τους και όχι τα ίδια τα μονοπάτια. Τα μονοπάτια P_1 και P_2 δεν έχουν εν γένει τον ίδιο αριθμό κόμβων, επομένως δεν μπορούμε να κάνουμε μία-προς-μία αντιστοίχιση των κόμβων τους. Μπορούμε όμως να αντιστοιχίσουμε ένα προς ένα τους κόμβους τους με ίδιο επίπεδο. Ονομάζουμε αυτές τις σχέσεις *κλωστές*, επειδή η δημιουργία τους δηλώνει την επιθυμία μας οι δύο κόμβοι που αυτές συσχετίζουν να έχουν κοντινές αποστάσεις στο τελικό σχέδιο, όπως παραδείγματος χάριν αν τους συνέδεε μία κλωστή με περιορισμένο μήκος ή με μικρή ελαστικότητα. Στο σχήμα 3.3 οι κλωστές φαίνονται με τη μορφή κυματιστών διακεκομμένων γραμμών. Για το ζεύγος μονοπατιών (P_1, P_2) όπου $P_1 = (A, v_1, \ldots, v_k, t)$ και $P_2 = (A, w_1, \ldots, w_l, t)$ δημιουργούμε το εξής σύνολο κλωστών $K = \{(v_2,\ w_2),\ \ldots\ ,\ (v_k,\ w_k),\ (v_k,\ w_{k+1}),\ \ldots\ ,\ (v_k,\ w_l)\}$. Παρατηρούμε ότι δεν δημιουργούμε τη κλωστή που αντιστοιχεί στους δεύτερους κόμβους των P_1 και P_2, για λόγους που θα δούμε αργότερα. Ξεκινώντας από τους τρίτους κόμβους των μονοπατιών, δημιουργούμε κλωστές έως ότου φτάσουμε στο επίπεδο που αντιστοιχεί στον κόμβο v_k. Το μονοπάτι P_1 δεν έχει κόμβους στα επόμενα επίπεδα (με εξαίρεση τον κόμβο t), γι'αυτό συνδέουμε με κλωστές τον v_k με όλους τους υπόλοιπους κόμβους του μονοπατιού P_2. Λόγω του τρόπου κατασκευής των μονοπατιών, είναι δυνατόν κατά τη κατασκευή των κλωστών να προσπαθήσουμε να κατασκευάσουμε κάποια ήδη υπάρχουσα κλωστή. Οι κλωστές θεωρούνται μη διατεταγμένα ζεύγη κόμβων, επομένως δεν κατασκευάζουμε την $(u,\ v)$ αν υπάρχει η $(v,\ u)$. Κάθε κλωστή έχει ένα βάρος το οποίο είναι μια αριθμητική ποσότητα. Αν ζητήσουμε τη κατασκευή μιας ήδη υπάρχουσας κλωστής, τότε δεν γίνεται κατασκευή νέας κλωστής, αλλά απλώς προσθέτουμε το βάρος της νέας κλωστής στο βάρος της προυπάρχουσας.

Σαν αρχικό βάρος κλωστών δοκιμάσαμε να δίνουμε την ποσότητα ($\frac{\text{μήκος ακμής γράφου}}{\text{αριθμός κλωστών}}$). Σ'αυτή τη περίπτωση, κάθε κλωστή έχει βάρος το οποίο εξαρτάται από το μήκος της ακμής γράφου από την οποία προήλθε. Επίσης, δοκιμάσαμε να θέσουμε σαν βάρος το (C_1 x # κλωστής), όπου C_1 είναι μία σταθερά, έτσι ώστε μεγαλύτερο βάρος να έχουν οι κλωστές οι οποίες δημιουργούνται τελευταίες για κάθε ακμή γράφου. Ακόμη, δοκιμάσαμε ως βάρος την ποσότητα ($\frac{\text{\# κλωστής}}{\text{αριθμός κλωστών}}$), η οποία εξαρτά το βάρος κάθε κλωστής από το μήκος της ακμής γράφου, αλλά και από τη σειρά της στη διαδικασία κατασκευής των κλωστών στα πλαίσια κάθε ακμής γράφου. Τέλος, δοκιμάσαμε όλες οι κλωστές να ξεκινάνε με το ίδιο βάρος, C. Οι τρεις πρώτοι τρόποι υπολογισμού του βάρους κάθε κλωστής, απαιτούν περισσότερο υπολογιστικό χρόνο σε σχέση με τον τελευταίο και επιπλέον δεν φάνηκε να δίνουν καλύτερα αποτελέσματα. Γι'αυτό αποφασίσαμε να χρησιμοποιήσουμε τον τελευταίο τρόπο για τον

υπολογισμό των βαρών των κλωστών.

Η τελευταία κλωστή $(v_k,\ t)$ δεν κατασκευάζεται γιατί θεωρούμε ότι οι κόμβοι v_k και t είναι ήδη συσχετισμένοι μέσω της ακμής γράφου την οποία μπορούμε εδώ να την φανταστούμε σαν μία επιπλέον κλωστή. Επίσης, δεν κατασκευάζουμε την πρώτη κλωστή $(v_1,\ w_1)$. Αντί γι'αυτό, το ζεύγος $(v_1,\ w_1)$ συμμετέχει στη διαδικασία κατασκευής ομάδων συσχετιζόμενων κόμβων, την οποία περιγράφουμε πιο κάτω.

3.3.2 Ομάδες συσχετιζόμενων κόμβων

Όταν έχει τελειώσει η διαδικασία κατασκευής των κλωστών τα ζεύγη (u, v), όπου u και v είναι οι δεύτεροι κόμβοι ενός ζεύγους χαρακτηριστικών μονοπατιών κάποιας ακμής γράφου, χρησιμοποιούνται για την κατασκευή ομάδων συσχετιζόμενων κόμβων. Κάθε τέτοιο ζεύγος μπορεί να δημιουργήσει μια νέα ομάδα, να προσθέσει ένα νέο μέλος σε μια ήδη υπάρχουσα ή να οδηγήσει σε ένωση ομάδων.

Για κάθε ζεύγος κόμβων $(u,\ v)$ που συσχετίζεται μ'αυτό το τρόπο, έχουμε να παρατηρήσουμε τα εξής:

α) ο v και ο w είναι αδέλφια και παιδιά του πρώτου κόμβου των χαρακτηριστικών μονοπατιών που τους αντιστοιχούν,

β) ο v και ο w είναι ρίζες των υποδέντρων $T(v)$ και $T(w)$ αντίστοιχα, για τα οποία ισχύει: μια ή περισσότερες ακμές γράφου έχει για αφετηρία κάποιο κόμβο του $T(v)$ και για προορισμό κάποιο κόμβο του $T(w)$ ή αντίστροφα, και

γ) οι v και ο w ονομάζονται *Κύριοι* ή *Δευτερεύοντες* ανάλογα με τον τύπο του ζεύγους χαρακτηριστικών μονοπατιών $(P_1,\ P_2)$. Αν το $(P_1,\ P_2)$ είναι του πρώτου τύπου, τότε ο ένας κόμβος ονομάζεται Κύριος και ο άλλος Δευτερεύων, ενώ αν είναι του δεύτερου τύπου τότε και οι δύο κόμβοι είναι Κύριοι κόμβοι.

Κάθε ομάδα O αποτελείται από ένα σύνολο κόμβων $M_{\mathrm{O}} = \{v_1, \ldots, v_m\}, m \geq 2$ οι οποίοι ονομάζονται *μέλη* της ομάδας και ένα σύνολο ζευγών P_O. Τα στοιχεία του M_{O} είναι αδέλφια, ανήκουν στο ίδιο επίπεδο κόμβων και επιπλέον είναι δεύτεροι κόμβοι σε κάποιο ζεύγος χαρακτηριστικών μονοπατιών. Τα στοιχεία του P_O είναι μή διατεταγμένα ζεύγη στοιχείων του M_O. Με κάθε ζεύγος p που ανήκει στο P_O συσχετίζουμε ένα βάρος w_p και μία λίστα κόμβων L_p. Τα μέλη κάθε ζεύγους p είναι οι δεύτεροι κόμβοι ενός ή περισσοτέρων ζευγών χαρακτηριστικών μονοπατιών $(P_1,\ P_2)$ ακμών γράφου. Το w_p είναι το άθροισμα των μηκών αυτών των ακμών γράφου, ενώ η λίστα L_p είναι οι η λίστα που περιέχει τους προορισμούς αυτών των ακμών. Ορισμένοι από τους κόμβους μέλη της ομάδας, αυτοί που ονομάζονται Κύριοι, θεωρούνται περισσότερο σημαντικοί από τους υπόλοιπους, ενώ οι υπόλοιποι δηλαδή οι Δευτερεύοντες, είναι λιγότερο σημαντικοί για λόγους τους οποίους θα δούμε στη συνέχεια.

Στό σχήμα 3.4 περιγράφουμε σε ψευδογλώσσα την κατασκευή των ομάδων συσχετιζόμενων κόμβων.

Κύριοι και Δευτερεύοντες κόμβοι

Οι Κύριοι κόμβοι K_i κάθε ομάδας είναι αυτοί για τους οποίους υπάρχουν ακμές γράφου οι οποίες ξεκινάνε από το υποδέντρο $T(K_i)$ και καταλήγουν σε κάποιο υποδέντρο $T(K_j)$, αλλά και ακμές γράφου που καταλήγουν στο $T(K_i)$ σε κάποιο επίπεδο l μεγαλύτερο από το επίπεδο του κόμβου K_i και ξεκινάνε από το υποδέντρο $T(K_k)$, όπου $K_i,\ K_j$καιK_k είναι όλοι Κύριοι κόμβοι της ίδιας ομάδας. Οι Δευτερεύοντες κόμβοι είναι αυτοί από τους οποίους ξεκινάει κάποια ακμή γράφου, η οποία καταλήγει στο υποδέντρο ενός Κύριου κόμβου της ίδιας ομάδας. Στο σχήμα 3.3 ο κόμβος Α έχει μία μόνο ομάδα συσχετιζόμενων κόμβων, η οποία έχει τρία μέλη, τους κόμβους Β, Γ και Δ και δύο ζεύγη συσχετιζόμενων κόμβων, τα (Β, Γ) και (Γ, Δ). Οι κόμβοι Γ και Δ είναι Κύριοι ενώ ο Β Δευτερεύων. Η διάκριση των κόμβων σε Κύριους και Δευτερεύοντες (δηλαδή σημαντικούς και μη) γίνεται έχοντας υπόψη την αναδιάταξη των κόμβων του γράφου. Κάθε Κύριος κόμβος K_i, ο οποίος βρίσκεται στο επίπεδο l_{k_i} είναι ρίζα ενός υποδέντρου $T(K_i)$ του οποίου οι κόμβοι πρέπει να αναδιαταχθούν προκειμένου να ικανοποιηθούν οι απαιτήσεις που θέτουν οι ακμές γράφου που έχουν αφετηρία στο $T(K_i)$ και προορισμό στο $T(K_j)$ ή προορισμό στο $T(K_i)$ και αφετηρία στο $T(K_j)$ όπου K_j είναι κύριος κόμβος της ομάδας στην οποία ανήκει ο K_i. Αντίθετα, από έναν Δευτερεύοντα κόμβο Δ_i ξεκινούν μία ή περισσότερες ακμές γράφου αλλά δεν τίθεται θέμα αναδιάταξης του υποδέντρου $T(\Delta_i)$ λόγω αυτών των ακμών γράφου. Κάθε Δευτερεύων κόμβος σχετίζεται με έναν ή περισσότερους Κύριους κόμβους.

Ο τρόπος οργάνωσης των κόμβων σε ομάδες συσχετιζόμενων κόμβων, έχει ένα πλεονέκτημα το οποίο δεν είναι ίσως άμεσα φανερό. Μεταξύ των υποδέντρων με ρίζες τους κόμβους μέλη μιας ομάδας υπάρχει ένας συγκεκριμένος αριθμός ακμών γράφου που πηγαίνουν από ένα υποδέντρο σε κάποιο άλλο. Στο σχήμα 3.5 ο κόμβος Α που είναι και ρίζα του γράφου, έχει δύο ομάδες συσχετιζόμενων κόμβων, τις O_1 και O_2 με τρία και δύο μέλη αντίστοιχα. Στο σχήμα φαίνονται τα υποδέντρα με ρίζες τα μέλη των ομάδων καθώς και οι ακμές γράφου.

Η διακεκομμένη γραμμή είναι μια νοητή γραμμή η οποία χωρίζει τους απογόνους των μελών της O_1 από τους απογόνους των μελών της O_2. Δεν υπάρχει ακμή γράφου η οποία να τέμνει αυτή τη νοητή γραμμή. Αν υπήρχε, τότε το ένα της άκρο θα είχε πρόγονο στην O_1 και το άλλο στην O_2 και κατά τη φάση της κατασκευής των ομάδων, οι O_1 και O_2 θα είχαν συγχωνευθεί. Το ότι δεν υπάρχουν τέτοιες ακμές γράφου δεν μπορεί παρά να μας οδηγήσει στη σκέψη ότι αναδιατάσσοντας τα παιδιά του Α έτσι ώστε να έχουν τελικά μια αντίστοιχη διάταξή με αυτή του σχήματος 3.5 (όλοι οι κόμβοι της ομάδας O_1 να είναι δεξιά ή αριστερά

Για κάθε ζεύγος μονοπατιών $P_1 = (A, v_1, \ldots, v_k, T)$, $P_2 = (A, w_1, \ldots, w_l, T)$
που αντιστοιχεί σε μια ακμή γράφου e = (S, T) {

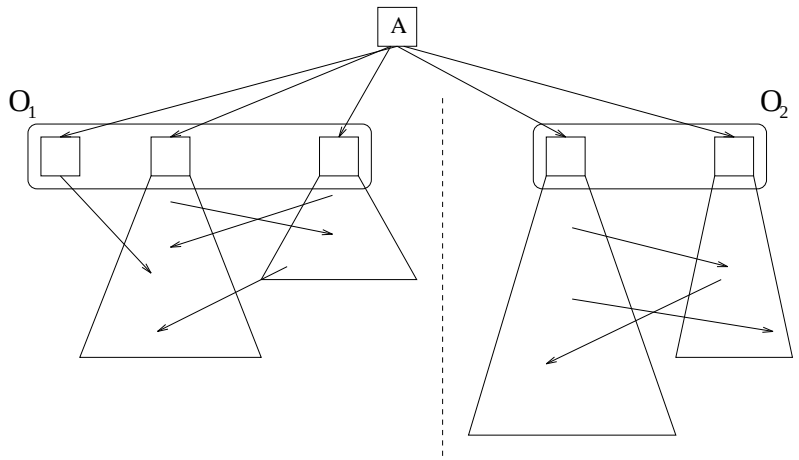
Υπαρχει_v_1 = Υπαρχει_w_1 = FALSE;
Ομαδα_v_1 = Ομαδα_w_1 = NULL;

if ($v_1 \in M_{O_i}$ για κάποια ομάδα M_{O_i} του A) {
Υπαρχει_v_1 = TRUE; Ομαδα_v_1 = O_i;
}
if ($w_1 \in M_{O_j}$ για κάποια ομάδα M_{O_j} του A) {
Υπαρχει_w_1 = TRUE; Ομαδα_w_1 = O_j;
}
Ομαδα = Ομαδα_v_1;
NZεύγος = $(v_1,\ w_1)$; $w_{\text{NZεύγος}}$ = μήκος της e; $L_{\text{NZεύγος}}$ = (T);

if (Ομαδα_v_1 == Ομαδα_w_1 && Ομαδα_v_1 != NULL) {
Πρόσθεσε το μήκος της e στο $w_{(v_1,\ w_1)}$;
Πρόσθεσε το T στο $L_{(v_1,\ w_1)}$;
} else if (Υπαρχει_v_1 && Υπαρχει_w_1) {
Συγχώνευσε την O_j στην O_i;
Διέγραψε την O_j;
Βάλε το NZεύγος στο P_{O_i};
} else if (Υπαρχει_v_1) {
Βάλε τον w_1 στο M_{O_i};
Βάλε το NZεύγος στο P_{O_i};
} else if (Υπαρχει_w_1) {
Βάλε τον ω_1 στο MO_j;
Βάλε το NZεύγος στο P_{O_j};
Ομαδα = Ομαδα_w_1;
} else {
Δημιούργησε μια νέα ομάδα O_k στον A;
$M_{O_k} = \{v_1, w_1\}$; $P_{O_k} = \{\text{NZεύγος}\}$; Ομαδα = Ομαδα$_k$;
}

Κάνε τον w_1 Κύριο κόμβο στην Ομαδα
if ((P_1, P_2) είναι του πρώτου τύπου) {
Κάνε τον v_1 Δευτερέων κόμβο στην Ομαδα
} else {
Κάνε τον v_1 Κύριο κόμβο στην Ομαδα
}
}

Σχήμα 3.4: Κατασκευή των ομάδων συσχετιζόμενων κόμβων



Σχήμα 3.5: Δύο διαφορετικές ομάδες συσχετιζόμενων κόμβων, παιδιών του A

από όλους τους κόμβους της ομάδας O_2) δεν μπορεί παρά να έχει καλά αποτελέσματα. Όντας διαταγμένοι με αυτό το τρόπο οι κόμβοι παιδιά του A, αποφεύγονται τουλάχιστον οι τομές ακμών γράφου με υποδέντρα τα οποία έχουν ρίζες σε άλλη ομάδα κόμβων από αυτήν που ανήκουν οι πρόγονοι της αφετηρίας και του προορισμού αυτών των ακμών γράφου. Η παραπάνω σκέψη μπορεί να γενικευτεί και να εφαρμοστεί σε γράφους στους οποίους έχουμε υπολογίσει ομάδες κόμβων.

Δεν είναι πάντα δυνατόν τα παιδιά ενός κόμβου να ανήκουν σε ομάδες συσχετιζόμενων κόμβων. Παραδείγματος χάριν, οι κόμβοι οι οποίοι είναι φύλλα δεν μπορούν να ανήκουν σε ομάδες. Στο εξής θα αναφερόμαστε σε αυτούς τους κόμβους ως *μη ομαδοποιημένους κόμβους*.

3.4 Αναδιάταξη των Κόμβων κατά τη Δεύτερη Φάση

Κατά τη διάρκεια της δεύτερης φάσης γίνεται η αναδιάταξη των κόμβων του ζευγνύοντος δέντρου ξεκινώντας απο τη ρίζα του δέντρου και εξετάζοντας κάθε φορά τους κόμβους ενός επιπέδου πριν προχωρήσουμε στους κόμβους του επόμενου επιπέδου. Γι'αυτή την αναδιάταξη θα στηριχθούμε σε πληροφορία που έχουμε συλλέξει ως τώρα η οποία αποτελείται όπως είδαμε από το σύνολο των ακμών γράφου, το σύνολο των κλωστών οι οποίες συνδέουν κόμβους, και το σύνολο των ομάδων συσχετιζόμενων κόμβων. Εκτός από τη δομή του ζευγνύοντος δέντρου, θα χρησιμοποιήσουμε και άλλες δομές, οι σημαντικότερες από τις οποίες θα περιγραφούν στη συνέχεια.

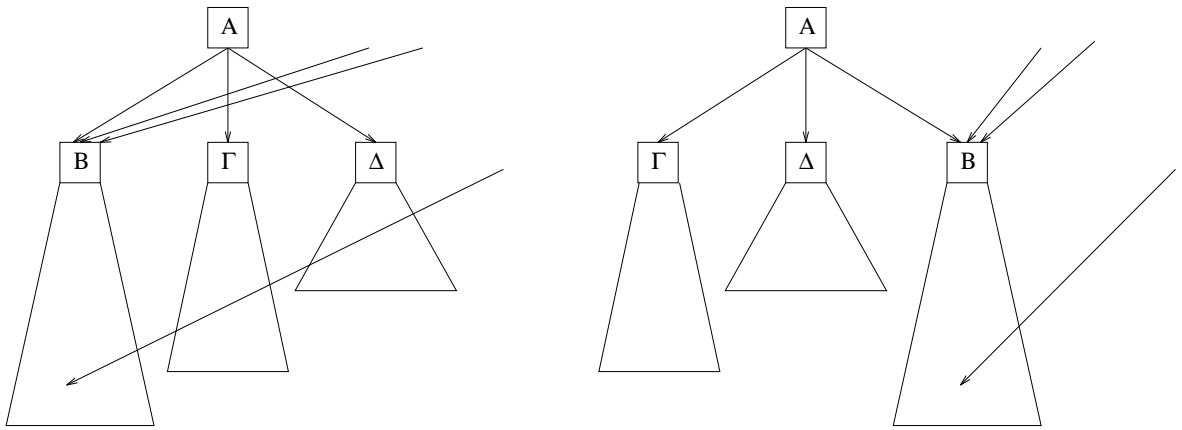
Σε κάθε κόμβο του δέντρου που επισκεπτόμαστε, στόχος μας είναι να αναδιατάξουμε τα παιδιά του έτσι ώστε μετά από αυτή την αναδιάταξη αλλά και άλλες οι οποίες θα

ακολουθήσουν, να βελτιωθεί το αρχικό σχέδιο του γράφου. Ήδη η πρώτη φάση μας έχει δώσει κάποιες ενδείξεις για το τρόπο αναδιάταξης των κόμβων. Αν σε κάθε κόμβο τοποθετήσουμε τα μέλη κάθε ομάδας συσχετιζόμενων κόμβων μετά ή πρίν από όλα τα μέλη κάποιας άλλης ομάδας και τέλος τοποθετήσουμε και όλους τους μη ομαδοποιημένους κόμβους σε τυχαίες θέσεις αλλά όχι μεταξύ των μελών ομάδων, έχουμε κάνει μια θετική κίνηση για την αναδιάταξη των κόμβων σύμφωνα με τα συμπεράσματά μας της προηγούμενης παραγράφου. Αν επιστρέψουμε όμως στα απλά παραδείγματα γράφων που δώσαμε στην αρχή του κεφαλαίου, θα δούμε ότι υπάρχουν περιπτώσεις όπου η αναδιάταξη των παιδιών ενός κόμβου με το τρόπο που περιγράψαμε προηγουμένως δεν μας βοηθάει να επιτύχουμε το στόχο μας. Παραδείγματος χάριν, στο σχήμα 3.1, ο κόμβος Z έχει δύο μη ομαδοποιημένους κόμβους παιδιά, τους κόμβους Θ και Ι. Ο στόχος μας είναι δεδομένης για παράδειγμα της θέσης που έχουν στο σχήμα οι κόμβοι Γ και Δ, οι Θ και Ι να αντιστρέψουν τελικά τις θέσεις τους.

3.4.1 Η κατεύθυνση ενός κόμβου

Ας ξεχάσουμε για λίγο τις ομάδες συσχετιζόμενων κόμβων και ας εξετάσουμε κάθε κόμβο του γράφου ανεξάρτητα από την ομάδα στην οποία αυτός μπορεί να ανήκει. Κάθε κόμβος μπορεί να ανήκει ή να μην ανήκει σε ένα ζεύγος χαρακτηριστικών μονοπατιών $(P_1,\ P_2)$ κάποιας ακμής γράφου. Αν δεν ανήκει, τότε όπως θα δούμε στη συνέχεια η τοποθέτηση αυτού του κόμβου μεταξύ των αδελφιών του δεν παρουσιάζει δυσκολία. Αν όμως ανήκει, τότε τα χαρακτηριστικά μονοπάτια, αλλά και ο ίδιος ο κόμβος πρέπει να μετακινηθούν όσο το δυνατόν πλησιέστερα στα σύνορα των υποδέντρων που τα περιέχουν έτσι ώστε να βρεθεί τελικά ο συγκεκριμένος κόμβος κοντά στους κόμβους με τους οποίους συνδέεται μέσω κλωστών και ακμών γράφου. Επειδή κάθε κόμβος έχει την ελευθερία να κινηθεί μόνο μέσα στα πλαίσια της λίστας των αδελφιών του, η παραπάνω κίνηση προς σύνορα υποδέντρων μεταφράζεται για κάθε κόμβο σε κίνηση προς την αρχή ή το τέλος της λίστας των αδελφιών του ή αντίστοιχα σε τοποθέτηση στην πρώτη ή τελευταία θέση αυτής της λίστας. Φυσικά δεν μπορούν όλοι οι κόμβοι να τοποθετηθούν στην πρώτη ή στη τελευταία θέση της λίστας των αδελφιών τους. Επομένως για να ορίσουμε τελικά μια διάταξη της παραπάνω λίστας, χρειάζεται να γνωρίζουμε την τάση κίνησης, ή *κατεύθυνση* του κόμβου (προς την αρχή ή το τέλος της λίστας) καθώς και ένα μέτρο αυτής της κατεύθυνσης. Έτσι, για τον κόμβο v, μεγαλύτερη τάση κίνησης προς την αρχή της λίστας έναντι των αδελφιών του έχει ως αποτέλεσμα την τοποθέτησή του εκεί.

Η κατεύθυνση ενός κόμβου v αλλά και το μέτρο της είναι άμεσα εξαρτημένα από τις σχέσεις που έχει ο κόμβος v με άλλους κόμβους του γράφου. Όσον αφορά την αναδιάταξη



(α) (β)

Σχήμα 3.6: Δύο δυνατές τοποθετήσεις του ίδιου κόμβου μέσα στη λίστα των αδελφιών του.

των κόμβων, έχουμε εκφράσει αυτές τις σχέσεις μέσω των κλωστών και των εισερχόμενων ακμών γράφου για κάθε κόμβο. Έτσι, η κατεύθυνση ενός κόμβου μπορεί να εκτιμηθεί από τον αριθμό των κλωστών που τον συνδέουν με άλλους κόμβους, και τον αριθμό των εισερχόμενων σ'αυτόν ακμών γράφου. Τη χρονική στιγμή t_v κατά την οποία επισκεπτόμαστε τον κόμβο v, ο οποίος βρίσκεται στο επίπεδο l_v, έχουμε ήδη επισκευθεί τους κόμβους των επιπέδων 0 έως l_{v-1} και έχουμε ήδη αποφασίσει για τις σχετικές θέσεις των κόμβων στα επίπεδα αυτά. Όπως θα δούμε αργότερα, υπάρχουν ορισμένοι κόμβοι για τους οποίους δεν έχουμε βρεί τη σχετική τους θέση προς τα αδέλφια τους, αλλά αυτό δεν θα έπρεπε να μας απασχολεί τώρα.

Κάθε κόμβος v έχει τη δυνατότητα να υπολογίσει τη σχετική του θέση ως προς κάθε κόμβο w με τον οποίο σχετίζεται και ο οποίος βρίσκεται σε μικρότερο ή ίσο επίπεδο από τον v. Αρκεί να βρεθούν οι πρόγονοι των v και w οι οποίοι είναι αδέλφια και να συγκριθούν οι σχετικές θέσεις των προγόνων αυτών. Δεδομένου του τρόπου λειτουργίας του αλγορίθμου σχεδιασμού δέντρων του S. Moen και της εισαγωγής των τεχνητών κόμβων στο ζευγνύον δέντρο πριν το σχεδιασμό του, οι σχετικές θέσεις των προγόνων αυτών είναι ισοδύναμες με τις σχετικές θέσεις των v και w. Η κατεύθυνση του v επηρεάζεται από τις θέσεις των κόμβων που σχετίζονται με αυτόν. Έτσι, παραδείγματος χάριν, όλοι οι κόμβοι οι οποίοι σχετίζονται με τον κόμβο Β του σχήματος 3.6 (β) δείχνουν να είναι στα δεξιά του αν πάρουμε υπόψη τη φορά των τριών ακμών γράφου οι οποίες φαίνονται στο σχήμα. Η θέση των σχετικών με τον Β κόμβων υπαγορεύουν για τον Β μια κατεύθυνση προς το τέλος της λίστας των αδελφιών του όσον αφορά τη τελική τοποθέτησή του. Στο 3.6 (α) βλέπουμε τις τομές ακμών που θα μπορούσαν να δημιουργηθούν αν επιλέγαμε να τοποθετήσουμε τον Β στην αρχή της λίστας.

Μπορούμε να θεωρήσουμε ότι η κατεύθυνση κάθε κόμβου v είναι ένα άθροισμα προσημασμένων ποσοτήτων κάθε μία από τις οποίες αντιστοιχεί σε μία κλωστή με άκρο τον v ή μία εισερχόμενη στον v ακμή γράφου. Το πρόσημο κάθε μιας από τις παραπάνω ποσότητες εξαρτάται από τη σχετική θέση ως προς τον v του άλλου άκρου της κλωστής ή της ακμής γράφου. Αυθαίρετα δεχόμαστε το πρόσημο $+$ να αντιστοιχεί στις ποσότητες που αντιστοιχούν σε κόμβους δεξιά του v. Έτσι, η κατεύθυνση ενός κόμβου v είναι το άθροισμα:

$$\text{Κατεύθυνση}(v) = \sum_{i=1}^{n} \text{ΣχετικήΘέση}(v,\ w_i) \times \text{Βάρος}((v,\ w_i)) + \sum_{j=1}^{m} \text{ΣχετικήΘέση}(u_j,\ v) \times I$$

Το $\{(v,\ w_1),\ \ldots\ , (v, w_n)\}$ είναι το σύνολο κλωστών με άκρο τον κόμβο v. Οι κόμβοι $w_1,\ \ldots\ , w_n$ βρίσκονται όλοι σε επίπεδα μικρότερα ή ίσα του επιπέδου του v. Η συνάρτηση Βάρος επιστρέφει το βάρος της κλωστής που δέχεται ως είσοδο. Το $\{u_1,\ \ldots\ , u_m\}$ είναι το σύνολο των αφετηριών των εισερχόμενων στον v ακμών γράφου. Τέλος το I είναι μια σταθερή ποσότητα. Η ποσότητα C που αντιστοιχεί στο βάρος το οποίο δίνεται σε κάθε νέα κλωστή που κατασκευάζεται, όπως και η ποσότητα I παίζουν μεγάλο ρόλο στον υπολογισμό της κατεύθυνσης των κόμβων. Πειραματικά βρήκαμε ότι μια σχέση $I = 50 * C$ δίνει καλά αποτελέσματα. Δίνοντας στο I μια τόσο μεγαλύτερη τιμή από το C, αναγκάζουμε τους κόμβους οι οποίοι έχουν εισερχόμενες ακμές γράφου, να μετακινηθούν προς μία από τις δύο άκρες της λίστας των αδελφιών τους, έτσι ώστε να μη δημιουργηθούν τομές ακμών μεταξύ αυτών των εισερχομενων ακμών γράφου και άλλων ακμών εισερχομενων στα αδέλφια αυτών των κόμβων. Θα έπρεπε όμως να εξεταστεί με περισσότερη προσοχή πως επηρεάζει η σχέση μεταξύ I και C τον υπολογισμό της κατεύθυνσης κόμβων και κατ'επέκταση την ποιότητα των αποτελεσμάτων του αλγορίθμου.

Η ΣχετικήΘέση είναι μια συνάρτηση η οποία βρίσκει τη σχετική θέση μεταξύ των δύο κόμβων που δέχεται ως είσοδο και επιστρέφει 1 αν ο πρώτος είναι αριστερά του δεύτερου, διαφορετικά επιστρέφει -1. Ο τρόπος με τον οποίο η ΣχετικήΘέση() υπολογίζει τα αποτελέσματά της περιγράφεται στην επόμενη παράγραφο.

Η δομή Tree

Στη παράγραφο αυτή περιγράφουμε μια δομή η οποία ονομάζεται **Tree** και χρησιμοποιείται αρκετά συχνά κατά την εκτέλεση του αλγορίθμου για την αποθήκευση των αποτελεσμάτων της αναδιάταξης των κόμβων, αλλά και τον υπολογισμό των κατευθύνσεων κόμβων.

Η δομή αυτή ειναι ένας μονοδιάστατος πίνακας L θέσεων. Το L είναι ο αριθμός των επιπέδων του γράφου. Το στοιχείο Tree[i] είναι μια λίστα με τους κόμβους του επιπέδου i. Μέσα στο Tree[i] υπάρχουν κατάλληλα διαχωριστικά που χωρίζουν μία λίστα αδελφιών από μία άλλη. Για κάθε κόμβο v ο οποίος βρίσκεται στο επίπεδο l_v, σημειώνουμε το σημείο στο

Tree[l_v+1] στο οποίο αρχίζει η λίστα των παιδιών του. Όταν έχουμε επισκευθεί όλους τους κόμβους του επιπέδου i τότε η διάταξη των κόμβων στις λίστες Tree[0], . . . , Tree[i] είναι αυτή που θα υπάρχει στο τελικό σχέδιο, ενώ οι λίστες Tree[i+1], . . . , Tree[L-1] είναι κενές.

Η δομή Tree δεν χρησιμοποιείται μόνο για την αποθήκευση των αποτελεσμάτων της αναδιάταξης, αφού θα μπορούσαμε να έχουμε τα ίδια αποτελέσματα αντανακλώντας την αναδιάταξη στους δείκτες οι οποίοι δείχνουν τον επόμενο κάθε κόμβου και το παιδί κάθε κόμβου. Η κύρια προσφορά αυτής της δομής είναι στην εύρεση των σχετικών θέσεων μεταξύ κόμβων η οποία γίνεται από τη συνάρτηση ΣχετικήΘέση() κατά τον υπολογισμό της Κατεύθυνσης ενός κόμβου. Προηγουμένως αναφέραμε ότι προκειμένου να βρούμε τη σχετική θέση δύο κόμβων αρκεί να βρούμε τους προγόνους τους οι οποίοι είναι αδέλφια και να κοιτάξουμε τις σχετικές θέσεις των προγόνων. Η συνάρτηση ΣχετικήΘέση( ) υπολογίζει το αποτέλεσμά της ως εξής: αν τα ορίσματά της έχουν το ίδιο επίπεδο, τα αντικαθιστά με τους προγόνους τους στο προηγούμενο επίπεδο, ενώ αν έχουν διαφορετικό επίπεδο, αντικαθιστά το όρισμα με το μικρότερο επίπεδο με τον πρόγονό του, ο οποίος βρίσκεται στο ίδιο επίπεδο με το άλλο της όρισμα. Στη συνέχεια χρησιμοποιώντας τη δομή Tree, εξετάζει αν τα δυο ορίσματά της βρίσκονται στη δομή αυτή. Αν ναί, τότε επιτρέφει τις σχετικές τους θέσεις στη δομή, διαφορετικά αντικαθιστά κάθε όρισμά της με τον πατέρα του και επαναλαμβάνει την ίδια διαδικασία έως ότου βρεί τη σχετική θέση των ορισμάτων της ή τα ορίσματά της γίνουν ίδια.

3.4.2 Αναδιάταξη κόμβων με βάση την κατεύθυνσή τους

Έχοντας υπολογίσει την Κατεύθυνση κάθε κόμβου έχουμε ουσιαστικά αντιστοιχίσει κάθε κόμβο σε ένα πραγματικό αριθμό ο οποίος ανάλογα με το πρόσημό του δείχνει τάση του κόμβου να τοποθετηθεί στην αρχή ή στο τέλος της λίστας των αδελφιών του. Αν ταξινομήσουμε τα παιδιά ενός κόμβου με βάση την Κατεύθυνσή τους, θα έχουμε πετύχει μία αναδιάταξη κόμβων η οποία ικανοποιεί όσο το δυνατόν περισσότερο τις τάσεις κόμβων να καταλάβουν συγκεκριμένες θέσεις. Θα έχουμε πετύχει δηλαδή το ζητούμενο.

Έχουμε ήδη πει ότι οι κόμβοι που ανήκουν σε ομάδες θα πρέπει τελικά να βρεθούν σε διαδοχικές θέσεις μέσα στη λίστα των αδελφιών τους. Επομένως η διαδικασία της τακινόμησης θα πρέπει να θεωρήσει κάθε ομάδα κόμβων σαν έναν ακόμη κόμβο προς ταξινόμηση. Αν κάθε κόμβος της ομάδας έπαιρνε μέρος στην ταξινόμηση, τότε θα ήταν δυνατόν τα μέλη κάθε ομάδας να βρεθούν σε μη διαδοχικές θέσεις μετά την ταξινόμηση.

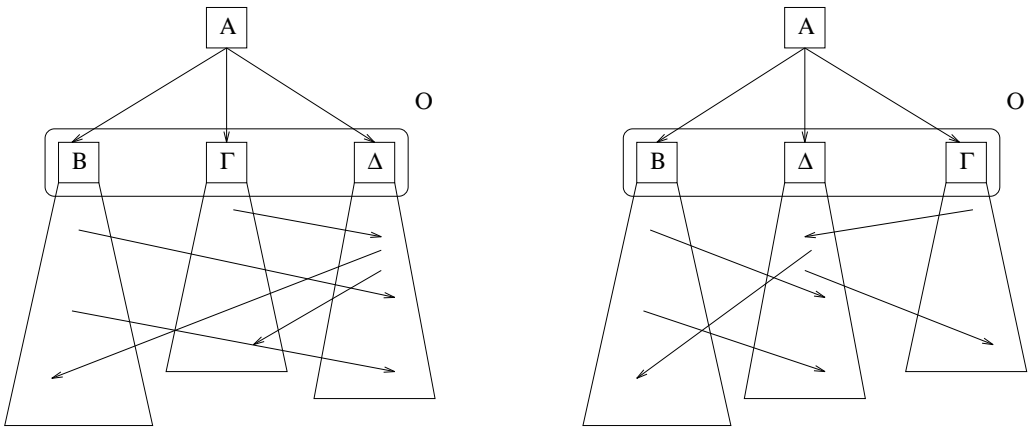
Κατεύθυνση ομάδας κόμβων

Για τους λόγους που αναφέραμε προηγουμένως, η Κατεύθυνση ενός κόμβου δεν είναι χρήσιμη εντός ομάδων. Γι'αυτό υπολογίζουμε και Κατευθύνσεις Ομάδων εκτός από Κατευθύνσεις κόμβων. Ο υπολογισμός της Κατεύθυνσης μιας ομάδας δεν είναι το ίδιο χρονοβόρα διαδικασία όσο ο υπολογισμός της Κατεύθυνσης ενός κόμβου, αφού ορίζεται να είναι το άθροισμα των Κατευθύνσεων των μελών της. Έτσι, στα πλαίσια της ταξινόμησης των παιδιών ενός κόμβου με βάση τις κατευθύνσεις τους, κάθε ομάδα θεωρείται σαν ένας υποθετικός κόμβος ο οποίος έχει Κατεύθυνση την Κατεύθυνση της ομάδας.

Όπως θα δούμε στη συνέχεια, η διάταξη των μελών κάθε ομάδας είναι σημαντική όσον αφορά την ποιότητα των αποτελεσμάτων του αλγορίθμου. Αφού βρούμε λοιπόν μια καλή διάταξη (σχετική θέση) των μελών θα πρέπει να τη διατηρήσουμε. Πρέπει να ληφθούν όμως υπόψη και οι Κατευθύνσεις κάθε μέλους της ομάδας. Γι'αυτό ακολουθούμε την παρακάτω διαδικασία: βρίσκουμε μια διάταξη των μελών της ομάδας. Θεωρούμε το σύνολο το οποίο αποτελείται από το πρώτο μισό των μελών της ομάδας και υπολογίσουμε τη συνολική του Κατεύθυνση. Κάνουμε το ίδιο και για το δεύτερο μισό των μελών της ομάδας. Αν βρούμε ότι η συνολική Κατεύθυνση του πρώτου μισού είναι μεγαλύτερη από του δεύτερου, αντιστρέφουμε τη διάταξη των μελών της ομάδας (ο πρώτος κόμβος στη διάταξη γίνεται τελευταίος). Έτσι, δίνουμε σε κάθε κόμβο μια θέση που αντιστοιχεί περισσότερο στη Κατεύθυνσή του και ταυτόχρονα διατηρούμε τη σχετική του θέση με τα άλλα μέλη της ομάδας. Για παράδειγμα, αν έχουμε μια ομάδα Ο όπου η διάταξη των μελών της είναι η (Α, Β, Γ, Δ) όπου Κατεύθυνση(Α) = 5, Κατεύθυνση(Β) = Κατεύθυνση(Γ) = 0 και Κατεύθυνση(Δ) = -15, τότε η συνολική Κατεύθυνση της ομάδας είναι -10 επομένως οι κόμβοι της θα τοποθετηθούν πριν από όλους τους κόμβους με Κατεύθυνση μεγαλύτερη από -10. Ακόμη η διάταξη των μελών της Ο πρέπει να γίνει (Δ, Γ, Β, Α) έτσι ώστε ο Δ (και αντίστοιχα ο Α) να βρεθεί πλησιέστερα στην άκρη της λίστας των αδερφιών του που υποδηλώνεται από τη Κατεύθυνσή του.

3.4.3 Διάταξη των κόμβων κάθε ομάδας

Όπως είδαμε στην παράγραφο 3.3.2, οι Κύριοι κόμβοι μιας ομάδας είναι σημαντικοί επειδή αποτελούν ρίζες δέντρων τα οποία πρέπει να αναδιαταχθούν. Οι Δευτερεύοντες κόμβοι είναι αφετηρίες ακμών γράφου οι οποίες έχουν τους προορισμούς τους στα υποδέντρα ενός ή περισσοτέρων Κύριων κόμβων. Η θέση των Δευτερεύοντων κόμβων όπως θα δούμε στη συνέχεια εξαρτάται από τη θέση των Κύριων. Επομένως, για να διατάξουμε τους κόμβους κάθε ομάδας, πρέπει πρώτα να διατάξουμε τους Κύριους κόμβους και στη συνέχεια να βρούμε τις σχετικές θέσεις των Δευτερεύοντων κόμβων ως προς τους Κύριους.



(α) (β)

Σχήμα 3.7: Δύο δυνατές τοποθετήσεις των Κύριων κόμβων μιας ομάδας συσχετιζόμενων κόμβων

Η διάταξη των Κύριων κόμβων είναι ιδιαίτερα σημαντική και μπορεί να επηρεάσει σε μεγάλο βαθμό την ποιότητα των παραγόμενων σχεδίων. Αναφερόμαστε σε ομάδες οι οποίες έχουν τουλάχιστον 3 Κύριους κόμβους, αφού η διάταξη δύο κόμβων είναι τετριμμένη. Σε μεγάλους γράφους (οι οποίοι έχουν περισσότερες πιθανότητες να έχουν ομάδες με αρκετούς Κύριους κόμβους) βρέθηκε ότι μία κακή διάταξη των Κύριων κόμβων μπορεί να δημιουργήσει μέχρι και 25 % περισσότερες τομές ακμών. Στο σχήμα 3.7 φαίνεται πώς μπορούμε να οδηγηθούμε σε μία τέτοια κακή κατάσταση. Η ομάδα Ο έχει τρεις Κύριους κόμβους, τους Β, Γ και Δ. Η κακή διάταξη των κόμβων στο (α) δεν έχει πάρει υπόψη ότι ολόκληρο σχεδόν το υποδέντρο του Γ παρεμβάλλεται στο χώρο στον οποίο δρομολογούνται οι ακμές γράφου οι οποίες έχουν αφετηρία στο υποδέντρο του Β και προορισμό στο υποδέντρο του Δ ή αντίστροφα. Στο 3.7 (β) το σχέδιο έχει φανερά βελτιωθεί.

3.4.4 Διάταξη των Κύριων κόμβων

Για τη διάταξη των Κύριων κόμβων μιας ομάδας πρέπει να γνωρίζουμε πώς αυτοί σχετίζονται μεταξύ τους. Αυτή η πληροφορία έχει αποθηκευθεί με τη μορφή ζευγών στο σύνολο ζευγών της ομάδας κατά τη πρώτη φάση του αλγορίθμου. Το βάρος που σχετίζεται με κάθε ζεύγος Κύριων κόμβων είναι ενδεικτικό του είδους των ακμών γράφου που δημιούργησαν αυτό το ζεύγος.

Διάταξη σύμφωνα με τα βάρη των ζευγών

Στη συνέχεια περιγράφουμε μια αρχική προσπάθεια να διατάξουμε τους Κύριους κόμβους μιας ομάδας η οποία δεν περιέχεται στη τρέχουσα έκδοση του αλγορίθμου λόγω αντικατάστασής της από άλλη καλύτερη.

Στα πλαίσια αυτής της προσπάθειας φανταζόμαστε τους Κύριους κόμβους σαν μέρος ενός μηχανικού συστήματος. Οι κόμβοι κάθε ζεύγους θεωρούμε ότι συνδέονται με ένα ελατήριο του οποίου η σταθερά είναι ίση με το βάρος που αντιστοιχεί στο ζεύγος. Προσπαθούμε να βρούμε τη διάταξη των κόμβων η οποία αντιστοιχεί στην ελάχιστη ενέργεια του συστήματος. Σ'αυτή τη διάταξη, οι κόμβοι οι οποίοι συνδέονται με ελατήρια με μεγάλη σταθερά (δηλαδή το βάρος του αντίστοιχου ζεύγους είναι μεγάλο) θα βρεθούν κοντά.

Για την ελαχιστοποίηση της ενέργειας του συστήματος χρησιμοποιούμε μια ευρηματική μέθοδο η οποία βρίσκει τη σχετική τοποθέτηση των Κύριων κόμβων μιας ομάδας ως εξής: Αρχικά διατάσσουμε τους δύο κόμβους οι οποίοι αντιστοιχούν στο ζεύγος με το μεγαλύτερο βάρος. Στη συνέχεια παίρνουμε έναν κόμβο v από αυτούς οι οποίοι δεν έχουν διαταχθεί ακόμα, και ο οποίος σχετίζεται με το μεγαλύτερο συνολικό βάρος με ήδη τοποθετημένους κόμβους. Για κάθε μία από τις πιθανές θέσεις όπου μπορεί να τοποθετηθεί ο v υπολογίζουμε τη συνολική ενέργεια που της αντιστοιχεί. Τοποθετούμε τελικά τον v στη θέση με την ελάχιστη ενέργεια. Ο υπολογισμός της ενέργειας γίνεται χρησιμοποιώντας τον τύπο $\sum_{i=1}^{n} k_i x_i^2$, όπου n είναι ο αριθμός των διατεταγμένων κόμβων με τους οποίους σχετίζεται ο v, k_i είναι το βάρος που αντιστοιχεί στο ζεύγος του v και του $i_{\text{οστού}}$ κόμβου στη διάταξη και x_i είναι η απόσταση μεταξύ του v και του $i_{\text{οστού}}$ κόμβου (για διαδοχικές θέσεις δεχόμαστε $x_i = 1$). Συνεχίζουμε την ίδια διαδικασία έως ότου διατάξουμε όλους τους Κύριους κόμβους.

Η παραπάνω διαδικασία διάταξης εξασφαλίζει ότι κόμβοι οι οποίοι σχετίζονται με μεγάλο αριθμό ακμών γράφου ή με λίγες ακμές γράφου με μεγάλο μήκος θα βρεθούν τελικά κοντά με αποτέλεσμα αυτές οι ακμές να τέμνονται με όσο το δυνατόν λιγότερα ενδιάμεσα υποδέντρα. Διαισθητικά φαίνεται ότι ο παραπάνω τρόπος έχει πλεονεκτήματα έναντι μιας τυχαίας σχετικής τοποθέτησης και υπάρχουν περιπτώσεις όπου έχει καλά αποτελέσματα. Υπάρχουν όμως και περιπτώσεις γράφων όπου δεν είχε καλά αποτελέσματα. Παρ'όλο που η παραπάνω διαδικασία παίρνει υπόψη της τις σχέσεις μεταξύ Κύριων κόμβων αποτυχαίνει τελικά γιατί δεν κάνει καμιά εκτίμηση για τη ποιότητα των ενδιάμεσων βημάτων που κάνει για την τοποθέτηση κάθε κόμβου. Η ποιότητα κάθε ενδιάμεσου βήματος μπορεί να εκτιμηθεί μέσω των τομών ακμών οι οποίες δημιουργούνται. Σε μια τέτοια προσέγγιση στόχος θα ήταν η τοποθέτηση κάθε νέου κόμβου στη θέση η οποία αντιστοιχεί στις λιγότερες προκαλούμενες τομές ακμών γράφου.

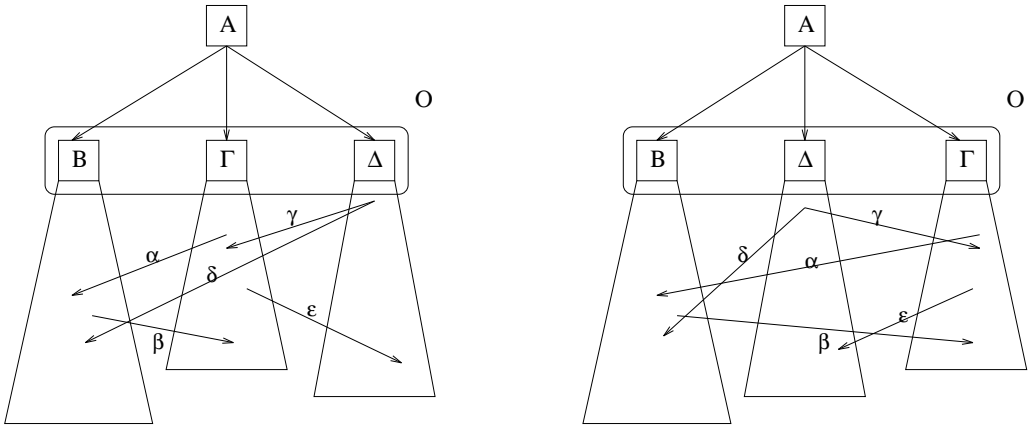
Διάταξη με υπολογισμό τομών ακμών

Η διαδικασία εύρεσης της σχετικής τοποθέτησης των Κύριων κόμβων που περιγράφεται εδώ έχει αρκετά κοινά σημεία με αυτή που περιγράψαμε προηγουμένως και είναι αυτή η οποία χρησιμοποιείται στη τρέχουσα έκδοση του αλγορίθμου. Και εδώ ξεκινάμε διατάσσοντας δύο Κύριους κόμβους και στη συνέχεια τοποθετούμε έναν-έναν τους υπόλοιπους. Για την τοποθέτηση καθενός από τους υπόλοιπους κόμβους ελέγχουμε ένα σύνολο από πιθανές θέσεις για τοποθέτηση και επιλέγουμε αυτή που αντιστοιχεί στις λιγότερες προκαλούμενες τομές ακμών.

Η επιλογή των δύο πρώτων Κύριων κόμβων που θα τοποθετηθούν μπορεί να γίνει με τρεις τρόπους. Ο πρώτος τρόπος είναι να κάνουμε μία τυχαία επιλογή δύο κόμβων. Ο δεύτερος είναι να επιλέξουμε τους δύο κόμβους οι οποίοι αντιστοιχούν στο ζεύγος με το μεγαλύτερο βάρος. Τέλος, ο τρίτος τρόπος είναι να επιλέξουμε τους δύο κόμβους οι οποίοι έχουν το μεγαλύτερο συνολικό βάρος στα ζεύγη στα οποία συμμετέχουν. Επιλέξαμε τον τρίτο τρόπο διότι αυτός ελλατώνει το πλήθος των πράξεων στα επόμενα βήματα: τοποθετούμε πρώτα τους κόμβους εκείνους που αν τους αφήναμε για μετά, θα είχαμε περισσότερη δουλειά να κάνουμε, εφόσον αυτοί έχουν τις περισσότερες ακμές γράφου που ξεκινούν ή καταλήγουν στα υποδέντρα τους.

Τα υπόλοιπα βήματα είναι επαναλήψεις του εξής: Έστω ότι βρισκόμαστε στη χρονική στιγμή κατά την οποία έχουμε βρεί τη σχετική τοποθέτηση Μ από τους Κύριους κόμβους και προσπαθούμε να βρούμε τη σχετική θέση του κόμβου v ως προς τους Μ κόμβους. Οι Μ κόμβοι δημιουργούν Μ+1 δυνατές θέσεις για να τοποθετήσουμε τον v. Μπορούμε να αποφύγουμε να εξετάσουμε τη περίπτωση τοποθέτησης και για τις Μ+1 θέσεις ως εξής: Έστω ότι ο πρώτος (κατά σειρά) διατεταγμένος κόμβος με τον οποίο σχετίζεται ο v είναι ο $i_{\text{οστός}}$ κόμβος και ο τελευταίος ο $j_{\text{οστός}}$, όπου $1 \leq i, j \leq$Μ. Τότε, αρκεί να εξετάσουμε τη περίπτωση τοποθέτησης του v στις θέσεις i έως j+1. Για κάθε μια από αυτές τις θέσεις μετράμε τις τομές ακμών οι οποίες θα δημιουργούνταν αν τοποθετούσαμε τον v σ'αυτή τη θέση. Οι τομές αυτές είναι τομές οι οποίες δημιουργούνται από ακμές γράφου οι οποίες τέμνουν υποδέντρα κάποιων από τους μέχρι αυτή τη στιγμή διατεταγμένους κόμβους.

Υπάρχουν δύο περιπτώσεις να δημιουργηθούν τέτοιες τομές ακμών. Στη πρώτη περίπτωση, ανάλογα με τη πιθανή θέση που εξετάζουμε, είναι δυνατόν οι ακμές γράφου οι οποίες σχετίζουν τον v μέ κόμβους στά δεξιά και αριστερά του να τέμνουν ενδιάμεσα υποδέντρα. Στη δεύτερη περίπτωση δημιουργούνται τομές επειδή το υποδέντρο του v βρίσκεται ανάμεσα σε δύο Κύριους κόμβους οι οποίοι σχετίζονται μεταξύ τους. Το σχήμα 3.8 είναι ένα παράδειγμα δημιουργίας τομών ακμών στις δύο περιπτώσεις. Υποθέτουμε εδώ ότι οι Β, Γ και Δ είναι Κύριοι κόμβοι, ότι έχουμε ήδη διατάξει τους κόμβους Β και Γ και ότι εξετάζουμε



(α) (β)

Σχήμα 3.8: Τομές ακμών δημιουργούμενες κατά την τοποθέτηση των Κύριων κόμβων

δυνατές θέσεις για τον Δ. Στο σχήμα 3.8 (α) έχουμε τοποθετήσει τον Δ στην τρίτη από τις πιθανές θέσεις. Η ακμή γράφου δ (η οποία τον συσχετίζει με τον Β) τέμνει το υποδέντρο με ρίζα τον Γ (πρώτη περίπτωση). Στο σχήμα 3.8 (β) έχουμε τοποθετήσει τον Δ στην δεύτερη από τις πιθανές θέσεις. Εδώ, το υποδέντρο με ρίζα τον Δ βρίσκεται μεταξύ των υποδέντρων των κόμβων Β και Γ οι οποίοι σχετίζονται με τις ακμές α και β, επομένως αυτές οι ακμές τέμνουν το υποδέντρο με ρίζα τον Δ (δεύτερη περίπτωση).

Δημιουργείται εδώ εύλογα η απορία πώς είναι δυνατόν να μετρήσουμε αυτές τις τομές ακμών. Παρατηρούμε ότι στην πρώτη αλλά και στη δεύτερη περίπτωση παραπάνω αρκεί να έχουμε τον τρόπο να υπολογίζουμε τις τομές που δημιουργεί μια συγκεκριμένη ακμή γράφου e με ένα συγκεκριμένο υποδέντρο $T(v)$ για να μπορούμε να υπολογίζουμε και τις συνολικές τομές ακμών κάθε φορά. Αν θυμηθούμε το τρόπο με τον οποίο σχεδιάζουμε τελικά τις ακμές γράφου, θα δούμε ότι μόνο το τελευταίο τμήμα της ακμής γράφου δημιουργεί τομές ακμών. Αύτο είναι το τμήμα μεταξύ του τελευταίου τεχνητού κόμβου και του προορισμού της ακμής e. Κατά την κατασκευή των ομάδων έχουμε συσχετίσει τους προορισμούς των ακμών με τα ζεύγη Κύριων κόμβων. Έτσι, θα έχουμε λύσει το πρόβλημά μας αν ξέρουμε τον αριθμό των ακμών (ακμές δέντρου και ακμές γράφου) οι οποίες υπάρχουν στον υπογράφο $G(v)$ μεταξύ των επιπέδων l και l+1 όπου l+1 είναι το επίπεδο του προορισμού της e. Αυτή η πληροφορία υπολογίζεται αναδρομικά για κάθε κόμβο του γράφου πριν την εκτέλεση του αλγορίθμου και αποθηκεύεται για κάθε κόμβο σε ένα πίνακα ακεραίων L το πολύ θέσεων, όπου L+1 είναι ο αριθμός των επιπέδων του γράφου.

3.4.5 Διάταξη των Δευτερεύοντων κόμβων

Όπως έχουμε ήδη δει, οι Δευτερεύοντες κόμβοι σχετίζονται με έναν ή περισσότερους Κύριους κόμβους μέσω μίας ή περισσότερων ακμών γράφου. Ας θεωρήσουμε το απλό παράδειγμα ενός Δευτερεύοντα κόμβου Δ ο οποίος σχετίζεται μόνο με τον Κύριο κόμβο Κ μέσω δύο ακμών γράφου. Η παραπάνω σχέση, καθώς και η λογική της τοποθέτησης κόμβων που σχετίζονται μεταξύ τους σε κοντινές θέσεις, υπαγορεύει την τοποθέτηση του Δ δίπλα στον Κ, αφού οποιαδήποτε άλλη τοποθέτηση είναι δυνατόν να δημιουργήσει επιπλέον τομές ακμών.

Η Διάταξη κάθε Δευτερεύοντα κόμβου εξαρτάται από τις σχέσεις που έχει με τους Κύριους κόμβους της ομάδας. Διακρίνουμε δύο περιπτώσεις Δευτερεύοντων κόμβων: αυτούς που σχετίζονται με ένα μόνο Κύριο κόμβο, και αυτούς που σχετίζονται με περισσότερους. Οι χειρισμοί μας σε καθε περίπτωση είναι διαφορετικοί. Παρακάτω περιγράφουμε το τρόπο με τον οποίο χειριζόμαστε τη διάταξη των Δευτερευόντων κόμβων μεταξύ των Κύριων.

Δευτερεύοντες κόμβοι που σχετίζονται με περισσότερους απο έναν Κύριο κόμβο.

Αυτοί οι Δευτερεύοντες κόμβοι είναι συνηθώς ένα μικρό ποσοστό του συνόλου των Δευτερεύοντων κόμβων (περίπου 5 %). Για κάθε τέτοιο κόμβο v, υπάρχει ένας αριθμός $n > 1$ Κύριων κόμβων με τους οποίους σχετίζεται. Σύμφωνα με τα όσα είπαμε στην αρχή αυτής της παραγράφου η πιο κατάλληλη θέση για να τοποθετήσουμε τον v είναι κάποια θέση μεταξύ των n Κύριων κόμβων.

Η εύρεση της καλύτερης θέσης μεταξύ αυτών μας θυμίζει τη διαδικασία της διάταξης των Κύριων κόμβων. Στην πραγματικότητα, το κριτήριο για την εκτίμηση της καταλληλότερης θέσης είναι και πάλι το ίδιο, δηλαδή ο αριθμός των δημιουργούμενων τομών ακμών. Η διαδικασία είναι εντελώς ανάλογη, μόνο που εδώ όταν θεωρούμε τις δυνατές θέσεις για την τοποθέτηση του Δευτερεύοντα κόμβου δεν παίρνουμε υπόψη τους Δευτερεύοντες κόμβους οι οποίοι έχουν ήδη τοποθετηθεί μεταξύ των Κύριων (ας θυμηθούμε ότι δεν υπάρχουν σχέσεις μεταξύ Δευτερεύοντων κόμβων).

Δευτερεύοντες κόμβοι που σχετίζονται με έναν Κύριο κόμβο.

Οι Δευτερεύοντες κόμβοι αυτής της κατηγορίας είναι αυτοί οι οποίοι παρουσιάζουν μεγαλύτερο ενδιαφέρον. Είναι το μεγαλύτερο ποσοστό των Δευτερεύοντων κόμβων, γιατί ανάμεσά τους συγκαταλέγονται και όλοι οι τεχνητοί κόμβοι που εισαγάγαμε για κάθε πρόσθια ακμή γράφου. Συνήθως το 95 % περίπου αυτών των κόμβων είναι τεχνητοί κόμβοι. Η διάταξη αυτών των Δευτερεύοντων κόμβων κάνει πιο πολύπλοκους τους χειρισμούς του αλγορίθμου όπως θα δούμε στη συνέχεια.

Στο τελικό σχέδιο, αυτοί οι κόμβοι τοποθετούνται κοντά στον Κύριο κόμβο με τον οποίο σχετίζονται. Το σημαντικό πρόβλημα εδώ είναι ότι δεν μπορούμε να αποφασίσουμε σ'αυτή τη φάση τη σχετική θέση κάθε τέτοιου κόμβου Δ_i με τον αντίστοιχο Κύριο κόμβο. Μια τέτοια απόφαση προϋποθέτει γνώση για τη τελική τοποθέτηση του συνόλου των προορισμών των ακμών γράφου T_{Δ_i} που σχετίζουν τους δύο κόμβους, αφού μια τοποθέτηση των κόμβων του T_{Δ_i} πιο κοντά σε ένα από τα δύο σύνορα του υποδέντρου του Κύριου κόμβου είναι ένδειξη και για τη σχετική θέση του Δ_i ως προς τον Κύριο κόμβο. Εφόσον τη δεδομένη χρονική στιγμή που αποφασίζουμε τη σχετική τοποθέτηση των Δευτερεύοντων κόμβων δεν έχουμε πληροφορία για τη τοποθέτηση κόμβων σε επόμενα επίπεδα (δεν έχει δημιουργηθεί καν ακόμα), μπορούμε είτε να αποφασίσουμε αυθαίρετα, ή να προσπαθήσουμε να πάρουμε μία απόφαση σύμφωνα με τη πληροφορία που έχουμε μέχρι τώρα, ή να αναβάλλουμε την τοποθέτηση αυτών των κόμβων.

Πριν περιγράψουμε τη διαδικασία στην οποία καταλήξαμε για την τοποθέτηση τέτοιων Δευτερεύοντων κόμβων θα αναφερθούμε σε μία αρχική προσπάθεια, η οποία απέτυχε μεν στο να κάνει μια καλή τοποθέτηση των Δευτερεύοντων κόμβων, αλλά μας βοήθησε στην εξαγωγή χρήσιμων συμπερασμάτων.

Αρχικά δοκιμάσαμε μια αυθαίρετη διάταξη των Δευτερεύοντων κόμβων, η οποία φρόντιζε όμως να μοιράζει τους Δευτερεύοντες κόμβους μεταξύ των δύο πλευρών του Κύριου κόμβου. Το μόνο θετικό που είχε αυτή η τοποθέτηση κόμβων ήταν ότι δεν επιβάρυνε το συνολικό χρόνο εκτέλεσης του αλγορίθμου. Είχε όμως ένα σοβαρό μειονέκτημα. Τοποθετώντας άμεσα κάθε τέτοιο κόμβο Δ_i επηρεάζουμε αυτομάτως και την τοποθέτηση των κόμβων T_{Δ_i}, αφού καθορίζουμε ένα μέρος της κατεύθυνσής τους, βασισμένοι όμως σε τυχαίες επιλογές. Αν υποθέσουμε ότι δεν διατάσσουμε με τυχαίο τρόπο τους Δευτερεύοντες κόμβους αλλά έχουμε έναν τρόπο να εξάγουμε πληροφορία χρήσιμη για την διάταξη τους, τότε θα είχαμε και πάλι δύο βασικά προβλήματα:

α) θα βασίζαμε σημαντικές αποφάσεις σε πληροφορία η οποία θα ήταν εκτίμηση και όχι βεβαιότητα και

β) δεν θα χρησιμοποιούσαμε ποσοτικές εκτιμήσεις για να κρίνουμε τις αποφάσεις μας, έτσι αποφάσεις οι οποίες προς στιγμήν φαίνονται καλές θα μπορούσαν να οδηγήσουν σε μεγάλο αριθμό τομών ακμών.

Τα παραπάνω συμπεράσματα μας ώθησαν στη σκέψη ότι ίσως η καλύτερη τακτική για την αντιμετώπιση του προβλήματος της τοποθέτησης τέτοιων κόμβων θα ήταν να αναβάλλουμε προσωρινά την τοποθέτησή τους. Αφού έχουμε τοποθετήσει όλους τους κόμβους του συνόλου T_{Δ_i} που αντιστοιχεί σε κάποιο Δευτερεύοντα κόμβο Δ_i μπορούμε να τοποθετήσουμε και τον Δ_i. Μία εύκολη λύση θα ήταν να τοποθετήσουμε τον Δ_i στη σχετική

θέση ως προς τον αντίστοιχο Κύριο κόμβο K_i η οποία αντιστοιχεί στη συνολική Κατεύθυνση των κόμβων του συνόλου T_{Δ_i}. Για παράδειγμα, μία θετική συνολική Κατεύθυνση δείχνει ότι η πλειοψηφία των κόμβων του T_{Δ_i} έχει μετακινηθεί προς το δεξιό σύνορο του υποδέντρου με ρίζα τον K_i και επομένως ο Δ_i θα έπρεπε να τοποθετηθεί στα δεξιά του K_i. Ορισμένες φορές όμως, η παραπάνω λογική αποτυγχάνει, όπως όταν όλα τα παιδιά ενός κόμβου έχουν θετική Κατεύθυνση, οπότε μόνο λίγα από αυτά θα είναι κοντά στο σύνορο του αντίστοιχου υποδέντρου ενώ τα άλλα όχι. Έτσι, για άλλη μια φορά αποφασίσαμε να χρησιμοποιήσουμε τον αριθμό των δημιουργούμενων τομών ακμών ως κριτήριο τοποθέτησης, το οποίο οδηγεί βέβαια σε πιο χρονοβόρες λύσεις αλλά δεν έχει τα μειονεκτήματα της προηγούμενης μεθόδου.

Στη συνέχεια περιγράφουμε διάφορα θέματα σχετικά με την τοποθέτηση των Δευτερεύοντων κόμβων καθώς και διάφορα προβλήματα που ανακύπτουν.

Γεγονότα αναμονής και επίλυσή τους. Για κάθε Δευτερεύοντα κόμβο Δ_i του οποίου αναβάλλουμε την τοποθέτηση, δημιουργούμε μια σειρά γεγονότων αναμονής WE_i. Αυτά τα γεγονότα έχουν σκοπό να εκφράσουν την εξάρτηση του Δ_i από τους κόμβους του συνόλου T_{Δ_i}. Συγκεκριμένα, ο Δ_i περιμένει την τοποθέτηση κάθε κόμβου του T_{Δ_i} (γεγονότα πρώτου τύπου) ενώ κάθε κόμβος του T_{Δ_i} σημειώνει ότι ο Δ_i περιμένει γι'αυτόν (γεγονότα δεύτερου τύπου). Δημιουργούμε τόσα γεγονότα από κάθε τύπο όσα και το πλήθος των στοιχείων του T_{Δ_i}. Έστω $W(\Delta_i)$ το σύνολο των γεγονότων πρώτου τύπου για τον Δ_i. Ο Δ_i μπορεί να τοποθετηθεί όταν το $W(\Delta_i)$ γίνει κενό. Για να διαγραφεί ένα γεγονός αναμονής από το $W(\Delta_i)$ πρέπει να τοποθετηθεί ο κόμβος t του T_{Δ_i} που σχετίζεται μ'αυτό. Όταν έχουμε ολοκληρώσει την τοποθέτηση όλων των κόμβων ενός επιπέδου l, εξετάζουμε για κάθε κόμβο v του l, το σύνολο των γεγονότων αναμονής δεύτερου τύπου που έχουν δημιουργηθεί για αυτόν, οπότε μπορούμε να κάνουμε τις αντίστοιχες διαγραφές από τα σύνολα $W(\Delta)$.

Αναδιάταξη υποδέντρων Δευτερεύοντων κόμβων. Στη περίπτωση που ένας Δευτερεύων κόμβος Δ_i για τον οποίο αποφασίσαμε να αναβάλλουμε την τοποθέτησή του, έχει απογόνους, τότε κατά την τοποθέτηση του κόμβου Δ_i μέσα στο γράφο θα πρέπει να μεριμνήσουμε και για την εισαγωγή στο γράφο του υποδέντρου με ρίζα τον Δ_i.

Οι κόμβοι αυτού του δέντρου θα πρέπει να αναδιαταχθούν και αυτοί, οπότε εκτελούμε τη δεύτερη φάση του αλγορίθμου αναδρομικά ξεκινώντας από τα παιδιά του Δ_i. Τα αποτελέσματα και αυτών των αναδιατάξεων αποθηκεύονται στη δομή Tree.

Κόμβοι που δεν μπορούν να υπολογίσουν Κατεύθυνση. Όταν αναφερθήκαμε στον υπολογισμό της Κατεύθυνσης ενός κόμβου στην παράγραφο 3.4.1, δεν είχαμε πάρει υπόψη το γεγονός ότι κάποιοι κόμβοι μένουν προσωρινά ατοποθέτητοι.

Κατά τον υπολογισμό της Κατεύθυνσής του, κάθε κόμβος ελέγχει τις σχετικές θέσεις των προγόνων του και των κόμβων με τους οποίους σχετίζεται μέσω ακμών γράφου και κλωστών, οι οποίοι ενδεχομένως να είναι Δευτερεύοντες κόμβοι που δεν έχουν τοποθετηθεί ακόμα. Οι κόμβοι οι οποίοι δεν έχουν τοποθετηθεί δεν μπορούν να συνεισφέρουν μια προσημασμένη ποσότητα στον υπολογισμό της Κατεύθυνσης άλλων κόμβων. Έτσι, για όλους τους κόμβους για τους οποίους χρειάζεται να υπολογίσουμε την Κατεύθυνσή τους, εκτός από τη προσημασμένη ποσότητα στην οποία αναφερθήκαμε στη παράγραφο 3.4.1, υπολογίζουμε και μια άλλη ποσότητα την οποία ονομάζουμε *Δ_Κατεύθυνση*, η οποία δεν έχει πρόσημο, και για τον υπολογισμό της οποίας χρησιμοποιούμε τις σχέσεις με κόμβους οι οποίοι δεν έχουν τοποθετηθεί. Αντίστοιχα πράγματα ισχύουν και για ομάδες κόμβων

Η Δ_Κατεύθυνση, προκειμένου να είναι χρήσιμη (να μπορεί να αθροιστεί στην υπολογισμένη Κατεύθυνση) θα πρέπει να βρεθεί το πρόσημό της. Έχουμε ήδη αναφέρει ότι αφού υπολογίσουμε Κατευθύνσεις κόμβων και ομάδων κόμβων προχωρούμε σε ταξινόμηση των κόμβων σύμφωνα με τις Κατευθύνσεις τους, η οποία οδηγεί και στη τελική διάταξή τους. Πριν προχωρήσουμε στην ταξινόμηση των κόμβων, επιλέγουμε το πρόσημο των Δ_Κατευθύνσεων με τρόπο ώστε τελικά ο αριθμός των κόμβων με θετική Κατεύθυνση να είναι ίσος περίπου με τον αριθμό κόμβων με αρνητική Κατεύθυνση. Αφού επιλέξουμε το πρόσημο της Δ_Κατεύθυνσης, την προσθέτουμε στην Κατεύθυνση του κόμβου. Παρακάτω περιγράφουμε πως γίνεται η επιλογή των προσήμων των Δ_Κατευθύνσεων κόμβων και ομάδων κόμβων. Αν κάποιος κόμβος ή ομάδα κόμβων έχει μη μηδενική Κατεύθυνση και μη μηδενική Δ_Κατεύθυνση, τότε αποφασίζουμε να δώσουμε στη Δ_Κατεύθυνση το πρόσημο της Κατεύθυνσης. Επομένως μένει να δώσουμε πρόσημο στη Δ_Κατεύθυνση των κόμβων οι οποίοι έχουν μηδενική Κατεύθυνση. Αρχικά όλοι οι κόμβοι αλλά και οι ομάδες κόμβων οι οποίοι έχουν μηδενική Δ_Κατεύθυνση χρησιμοποιούνται για τον υπολογισμό δύο μετρητών ΚΘΚ (κόμβοι με θετική Κατεύθυνση) και ΚΑΚ (κόμβοι με αρνητική Κατεύθυνση). Μία ομάδα με θετική συνολική Κατεύθυνση και πέντε μέλη αυξάνει τον ΚΘΚ κατά πέντε. Για κάθε κόμβο ή ομάδα κόμβων με μή μηδενική Δ_Κατεύθυνση κάνουμε τα εξής:

(α) Αν ΚΘΚ < ΚΑΚ, δίνουμε θετικό πρόσημο στη Δ_Κατεύθυνση και αυξάνουμε τον ΚΘΚ κατά ένα ή κατά τον αριθμό των μελών της ομάδας αν πρόκειται για ομάδα κόμβων. Αλλιώς δίνουμε αρνητικό πρόσημο και αυξάνουμε ανάλογα τον ΚΑΚ.

(β) Ενημερώνουμε κάθε ατοποθέτητο κόμβο ο οποίος είχε συνεισφέρει στον υπολογισμό της Δ_Κατεύθυνσης για το πρόσημο που δόθηκε.

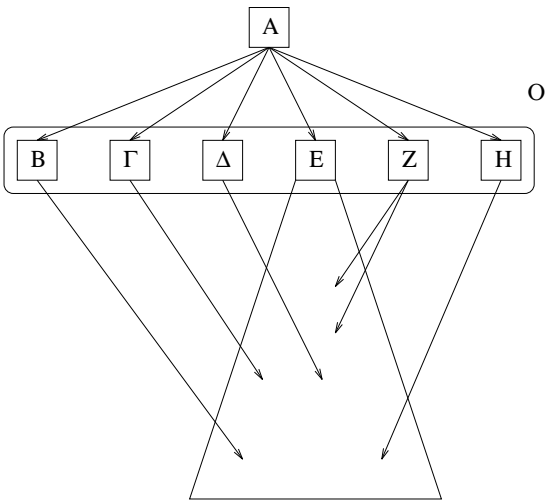
(γ) Μηδενίζουμε την Δ_Κατεύθυνση.

Στο βήμα (β) αναφέραμε ότι ενημερώνουμε τους ατοποθέτητους κόμβους. Με κάθε τέτοιο κόμβο έχουμε ήδη συσχετίσει μία ποσότητα την οποία ονομάζουμε *ΥποθετικήΣχετικήΘέση*,

η οποία αλλάζει τιμή όταν γίνεται η παραπάνω ενημέρωση. Όπως δείχνει και το όνομά της, αυτή η ποσότητα δείχνει τη σχετική θέση ως προς τον Κύριο κόμβο στην οποία υποθέτουμε ότι θα τοποθετηθεί ο Δευτερευών κόμβος. Την πρώτη φορά που ενημερώνεται αυτή η ποσότητα, παίρνει μία τιμή η οποία αντιστοιχεί στο πρόσημο της Δ_Κατεύθυνσης (θετικό πρόσημο αντιστοιχεί σε τοποθέτηση στα δεξιά του Κύριου κόμβου). Στις επόμενες εκτελέσεις του (β) η ποσότητα αυτή δεν αλλάζει τιμή. Η ποσότητα αυτή χρησιμοποιείται κατά τον υπολογισμό της Κατεύθυνσης των κόμβων. Αν η ΥποθετικήΣχετικήΘέση έχει πάρει τιμή, τότε παίρνουμε υπόψη μας αυτή τη τιμή υποθέτοντας ότι ο Δευτερεύων κόμβος έχει τοποθετηθεί στη συγκεκριμένη σχετική θέση, και έτσι αντί να αλλάξουμε την τιμή της Δ_Κατεύθυνσης αλλάζουμε την τιμή της Κατεύθυνσης του κόμβου. Η παραπάνω διαδικασία έχει σκοπό να επηρεάσει τελικά την τοποθέτηση όλων των κόμβων οι οποίοι σχετίζονται με έναν ατοποθέτητο Δευτερεύοντα κόμβο έτσι ώστε να βρεθούν όλοι κοντά στο ίδιο σύνορο του υποδέντρου που τους περιέχει. Δεν εξασφαλίζει όμως ότι ο Δευτερεύων κόμβος θα τοποθετηθεί στη σχετική θέση που υποδηλώνεται. Αυτό είναι και το επιθυμητό, αλλά όπως θα δούμε στη συνέχεια μπορεί η σχετική θέση που υποδηλώνει η ποσότητα ΥποθετικήΣχετικήΘέση να δημιουργεί περισσότερες τομές ακμών, οπότε και δεν θα προτιμηθεί.

Φωλιασμένη τοποθέτηση Δευτερεύοντων κόμβων. Επειδή κάθε Κύριος κόμβος μπορεί να σχετίζεται με περισσότερους από έναν Δευτερεύοντες κόμβους ο προσδιορισμός ‘‘τοποθετώ τον Δ_i δεξιά του K_i’’ δεν είναι αρκετός. Ενδέχεται δεξιά του K_i να υπάρχουν και άλλοι σχετιζόμενοι μ’αυτόν Δευτερεύοντες κόμβοι. Η θέση στην οποία επιλέγουμε να τοποθετήσουμε προσωρινά τον Δ_i, ώστε να μετρήσουμε τις δημιουργούμενες τομές ακμών είναι αυτή η οποία αντιστοιχεί σε φωλιασμό των ακμών γράφου που ξεκινούν από τον Δ_i και τους άλλους σχετιζόμενους Δευτερεύοντες κόμβους. Η έννοια του φωλιασμού εδώ έχει σχέση με τα μήκη αυτών των ακμών γράφου. Όπως θα φανεί και από τη διαδικασία υπολογισμού της τοποθέτησης Δευτερεύοντων κόμβων ώστε να είναι φωλιασμένοι, η οποία περιγράφεται στη συνέχεια, στόχος της είναι το να μη δημιουργηθούν επιπλέον τομές ακμών οι οποίες μπορούν να αποφευχθούν με αυτό το τρόπο.

Για κάθε Δευτερεύοντα κόμβο Δ_i, υπολογίζουμε τη μέση τιμή μήκους των ακμών γράφου οι οποίες έχουν αφετηρία Δ_i και προορισμό κάποιον απόγονο του Κύριου κόμβου με τον οποίο σχετίζεται ο Δ_i. Για να εξασφαλιστεί ο φωλιασμός, πρέπει κόμβοι με μικρότερο μέσο μήκος ακμών γράφου να τοποθετηθούν πιο κοντά στον Κύριο κόμβο. Σε περιπτώσεις ίσων τέτοιων μέσων τιμών, χρησιμοποιούμε το μέσον όρο των θέσεων των προορισμών των ακμών γράφου στη δομή Tree προκειμένου να αποφασίσουμε ποιός κόμβος τοποθετείται πιο κοντά στον Κύριο κόμβο. Στο σχήμα 3.9 δείχνουμε μια τοποθέτηση των Δευτερεύοντων κόμβων B,

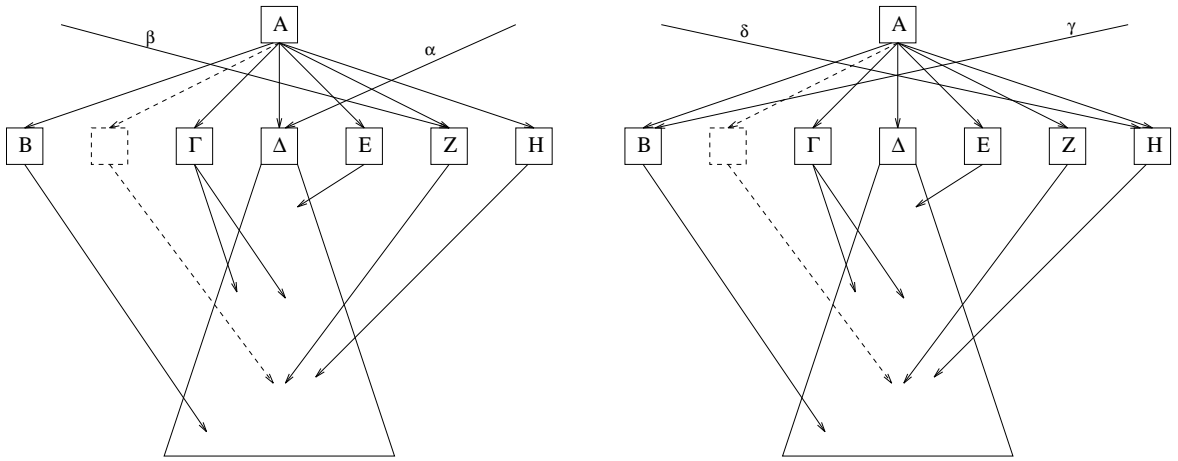


Σχήμα 3.9: Φωλιασμένη τοποθέτηση Δευτερεύοντων κόμβων γύρω από τον Κύριο κόμβο με τον οποίο σχετίζονται

Γ, Δ, Z και H γύρω από το Κύριο κόμβο E η οποία εξασφαλίζει φωλιασμό των ακμών γράφου. Παρατηρούμε ότι ο κόμβος Z έχει μικρότερο μέσο όρο μήκους ακμών γράφου από τον H, οπότε τοποθετείται σε πιο εσωτερική θέση. Οι κόμβοι Γ και Δ έχουν ίδιο μέσο όρο μηκών ακμών γράφου, αλλά ο μέσος όρος των θέσεων των προορισμών των ακμών γράφου του Δ στη δομή Tree είναι μεγαλύτερος από αυτόν του Γ, οπότε δεδομένης της σχετικής τους θέσης ως προς τον E, ο Δ τοποθετείται σε πιο εσωτερική θέση.

Δημιουργούμενες τομές ακμών από την τοποθέτηση ενός Δευτερεύοντα κόμβου. Στην παράγραφο αυτή περιγράφουμε με ποιο τρόπο μετράμε τις δημιουργούμενες τομές από την τοποθέτηση ενός Δευτερεύοντα κόμβου. Έστω ότι οι δύο σχετικές θέσεις ως προς τον Κύριο κόμβο για ένα Δευτερεύοντα κόμβο Δ_i είναι πρίν τον κόμβο N_1 και μετά τον κόμβο N_2. Ονομάζουμε S_{Δ_i} το σύνολο που περιέχει τους κόμβους N_1, N_2, καθώς και τους ενδιάμεσούς τους.

Εξετάζουμε τρείς περιπτώσεις δημιουργούμενων τομών ακμών. Η πρώτη περίπτωση αφορά τομές ακμών οι οποίες δημιουργούνται από ακμές γράφου που έχουν προορισμό στο επίπεδο l_{Δ_i}. Η δεύτερη περίπτωση αφορά τομές μεταξύ ακμών γράφου και του υποδέντρου του Δ_i. Τέλος, οι τομές ακμών της τρίτης περίπτωσης δημιουργούνται όταν το τελευταίο τμήμα των ακμών γράφου που έχουν αφετηρία τον Δ_i τέμνει ορισμένες από τις εισερχόμενες ακμές σε απογόνους κόμβων του S_{Δ_i}. Κάθε ακμή γράφου της οποίας το επίπεδο προορισμού είναι είτε πριν το επίπεδο του Δ_i είτε μετά το μέγιστο επίπεδο απογόνου του Δ_i δεν είναι δυνατόν να δημιουργήσει τομές ακμών. Η περίπτωση να εμπλακούν ακμές δέντρου στη



(α) (β)

Σχήμα 3.10: Δημιουργούμενες τομές ακμών κατά την τοποθέτηση του Δευτερεύοντα κόμβου Z - Πρώτη περίπτωση.

δημιουργία τομών ακμών εξετάζεται στην τελευταία από τις πιο πάνω περιπτώσεις. Σε κάθε άλλη περίπτωση, δεν είναι δυνατόν να δημιουργηθούν τομές ακμών στις οποίες εμπλέκονται ακμές δέντρου.

Τομές της πρώτης περίπτωσης. Από τις ακμές γράφου που το επίπεδο προορισμού τους είναι ίσο με το επίπεδο του Δ_i εξετάζουμε μόνο αυτές που έχουν προορισμό τον Δ_i ή κάποιο κόμβο του συνόλου S_{Δ_i}. Όπως θα δούμε στη συνέχεια, όλες οι υπόλοιπες ακμές είτε δεν δημιουργούν τομές ακμών, είτε οι τομές που δημιουργούν δεν μπορούν να αποφευχθούν εφόσον τοποθετήσουμε τον Δ_i σε μία από τις δύο θέσεις.

Για κάθε ακμή γράφου $e = (s, N_i)$, όπου $N_i \in S_{\Delta_i}$, είναι γνωστή η σχετική θέση $\Sigma\Theta_1$ του Δ_i ως προς τον N_i. Προκειμένου να βρούμε αν η e τέμνεται με την ακμή $(p,\ \Delta_i)$, όπου p είναι ο πατέρας του Δ_i, εξετάζουμε τη σχετική θέση $\Sigma\Theta_2$ του Δ_i ως προς τον s. Αν $\Sigma\Theta_1 \neq \Sigma\Theta_2$ τότε η e τέμνεται με την $(p,\ \Delta_i)$. Ο υπολογισμός της $\Sigma\Theta_2$ γίνεται από τη συνάρτηση ΣχετικηΘεση() που περιγράψαμε στην παράγραφο 3.4.1.

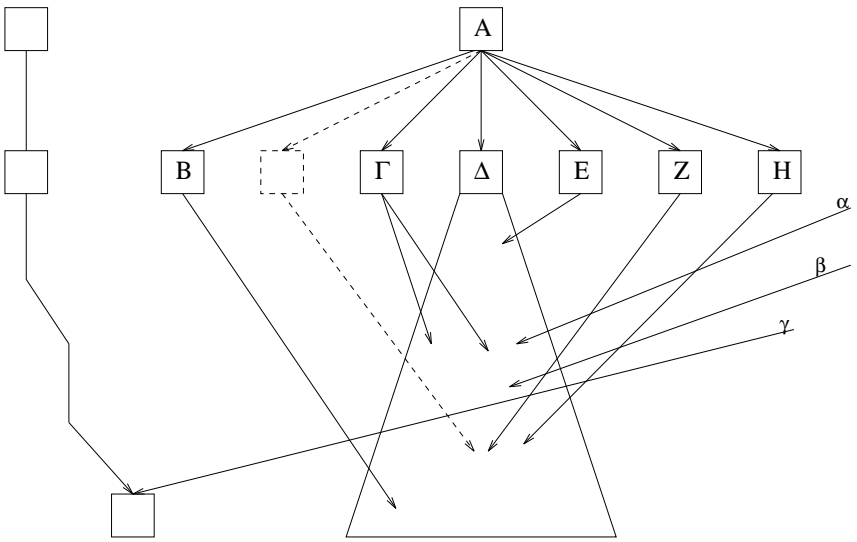
Τομές ακμών δημιουργούνται και από τις ακμές γράφου οι οποίες έχουν προορισμό τον Δ_i και τέμνουν ακμές εισερχόμενες σε κόμβους του συνόλου S_{Δ_i}. Για να δούμε αν δημιουργούνται τομές μεταξύ δύο τέτοιων ακμών ελέγχουμε αν η σχετική θέσή της αφετηρίας της πρώτης ως προς αυτήν της δεύτερης είναι αντίθετη από τη σχετική θέση του προορισμού της πρώτης ως προς αυτόν της δεύτερης, η οποία είναι γνωστή.

Στο σχήμα 3.10, υποθέτουμε ότι ο Δ είναι Κύριος κόμβος ενώ οι υπόλοιποι του ίδιου επιπέδου είναι Δευτερεύοντες, και υπολογίζουμε τις τομές ακμών από την τοποθέτηση του Z.

Με διακεκομμένες γραμμές φαίνεται η εναλλακτική θέση για τον Z. Στο 3.10 (α), η ακμή α δημιουργεί μια τομή με την ακμή (A, E) και ανήκει στη πρώτη υποπερίπτωση πιο πάνω, ενώ η ακμή β δημιουργεί 4 τομές με τις ακμές (A, Γ), (A, Δ), (A, E) και α, σύμφωνα με τη δεύτερη υποπερίπτωση πιο πάνω. Αν ο Z τοποθετηθεί στη άλλη σχετική θέση, τότε οι τομές αυτές δεν δημιουργούνται. Στο σχήμα 3.10 (β) εξηγούμε με ένα παράδειγμα γιατί δεν εξετάζουμε τομές με ακμές οι οποίες έχουν προορισμό εκτός του συνόλου S_{Z}. Όπως φαίνεται και στο σχήμα, οι τομές ακμών που δημιουργούνται απο τις ακμές γ και δ, θα δημιουργούνταν ούτως ή αλλως σε όποια θέση και αν τοποθετούσαμε τον Z.

Τομές της δεύτερης περίπτωσης. Οι τομές της δεύτερης περίπτωσης δημιουργούνται από ακμές γράφου των οποίων ο προορισμός έχει επίπεδο μεγαλύτερο από το επίπεδο του Δ_i και μικρότερο ή ίσο με το μεγαλύτερο επίπεδο του υποδέντρου του Δ_i. Από αυτές τις ακμές εξετάζουμε αν δημιουργούν τομές ακμών με το υποδέντρο του Δ_i μόνον εκείνες οι οποίες έχουν ως προορισμό κάποιο απόγονο κόμβου που ανήκει στο S_{Δ_i}. Αν κάποιες από τις υπόλοιπες δημιουργούν τομές ακμών, τότε αυτές οι ακμές δημιουργούνται ούτως ή άλλως, ανεξάρτητα από την τοποθέτηση του Δ_i. Για την εύρεση των σχετικών θέσεων που υποδηλώνουν τομή ακμών χρησιμοποιούμε τη συνάρτηση ΣχετικηΘεση() όπως και προηγουμένως.

Στο σχήμα 3.11 οι ακμές α και β δημιουργούν τομές ακμών που ανήκουν σ'αυτήν την



Σχήμα 3.11: Δημιουργούμενες τομές ακμών κατά την τοποθέτηση του Δευτερεύοντα κόμβου Z - Δεύτερη περίπτωση

περίπτωση, ενώ η ακμή γ τέμνει το υποδέντρο του κόμβου Z ανεξάρτητα από την τοποθέτηση

του κόμβου Z, επομένως δεν είναι χρήσιμο να μετρήσουμε αυτές τις τομές ακμών.

Τομές της τρίτης περίπτωσης. Οι τομές αυτής της περίπτωσης δημιουργούνται από τα τελευταία τμήματα των ακμών γράφου που έχουν αφετηρία τον Δ_i και τέμνουν εισερχόμενες ακμές σε απογόνους κόμβων του S_{Δ_i} οι οποίοι βρίσκονται στο ίδιο επίπεδο με τον προορισμό αυτών των ακμών γράφου.

Έστω t ο προορισμός μιας ακμής γράφου με αφετηρία τον Δ_i. Εξετάζουμε αν η ακμή $(\Delta_i,\ t)$ τέμνει εισερχόμενες ακμές σε κόμβους οι οποίοι είναι τοποθετημένοι πριν και μετά τον t και είναι απόγονοι των κόμβων του S_{Δ_i}. Ο έλεγχος για τομή ακμών γίνεται με ανάλογο τρόπο όπως στις δύο προηγούμενες περιπτώσεις χρησιμοποιώντας τη συνάρτηση ΣχετικηΘεση() για να βρούμε τις σχετικές θέσεις των άκρων δύο ακμών.

3.5 Τελική Επεξεργασία και Εισαγωγή των Τεχνητών Κόμβων

Μετά το τέλος της φάσης της αναδιάταξης των κόμβων έχουν βρεθεί οι σχετικές θέσεις για όλους τους κόμβους του γράφου και αυτή η πληροφορία έχει αποθηκευθεί στη δομή Tree. Πριν προχωρήσουμε στο σχεδιασμό του δέντρου χρησιμοποιώντας τον αλγόριθμο του S. Moen πρέπει να εισάγουμε τους τεχνητούς κόμβους στις ακμές γράφου οι οποίες έχουν μήκος μεγαλύτερο από 1. Σ'αυτή τη διαδικασία ενδιαφέρον παρουσιάζει η εισαγωγή του πρώτου τεχνητού κόμβου σε κάθε μακριά ακμή επειδή πρέπει να βρούμε τη σχετική του θέση ως προς τα αδέλφια του. Δεν εξετάζουμε τις πρόσθιες ακμές γράφου γιατί σε αυτές τις περιπτώσεις η τοποθέτηση του πρώτου τεχνητού κόμβου είναι τετριμμένη (δεν έχει αδέλφια). Όπως θα φανεί και στη συνέχεια, φροντίζουμε με την τοποθέτηση καθενός από τους τεχνητούς κόμβους να δημιουργηθούν όσο το δυνατόν λιγότερες τομές ακμών. Η εισαγωγή όλων των υπόλοιπων τεχνητών κόμβων μιας μακριάς ακμής είναι τετριμμένη. Αυτό συμβαίνει γιατί κάθε τέτοιος κόμβος δεν έχει αδέλφια και επομένως δεν τίθεται θέμα σχετικής του τοποθέτησης ως προς τους άλλους.

Αφού εισάγουμε τον πρώτο τεχνητό κόμβο στη δομή Tree, αντικατοπτρίζουμε την πληροφορία της αναδιάταξης, η οποία είναι αποθηκευμένη σ'αυτή τη δομή, στη δομή του δέντρου η οποία αποτελείται από τους δείκτες που δείχνουν από ένα κόμβο προς τον πατέρα, το επόμενο αδέλφι του και το παιδί του. Για οικονομία χρόνου, όλους τους υπόλοιπους τεχνητούς κόμβους μετά τον πρώτο, τους εισάγουμε απευθείας στη δομή του δέντρου.

Η εισαγωγή των τεχνητών κόμβων έχει στόχο να αναγκάσει τον αλγόριθμο σχεδιασμού του δέντρου να κρατήσει χώρο για το σχεδιασμό τους. Επειδή όμως οι τεχνητοί κόμβοι δεν σχεδιάζονται, ουσιαστικά ο χώρος αυτός χρησιμοποιείται για τη δρομολόγηση της ακμής γράφου στην οποία αντιστοιχούν. Η δημιουργία κάμψεων ακμών γενικά είναι ανεπιθύμητη,

εκτός από την περίπτωση που ο χρήστης αποφασίζει ότι θέλει ελαχιστοποίηση του χώρου που καταλαμβάνει το σχέδιο του γράφου, οπότε ο ανάλογος αλγόριθμος σχεδιασμού θα πρέπει να δημιουργήσει κάμψεις σε ακμές για να πετύχει αυτό το στόχο. Στα πλαίσια αυτής της εργασίας αποφεύγουμε τη δημιουργία περιττών κάμψεων ακμών, δημιουργώντας όμως τις περισσότερες φορές σχέδια με μεγαλύτερο εμβαδό από αυτό των σχεδίων που δημιουργεί ο αλγόριθμος STT.

Στο τέλος της παραγράφου 3.1 αναφέραμε ότι ο αλγόριθμος που σχεδιάσαμε δημιουργεί το πολύ 2 κάμψεις ανά μακριά ακμή γράφου. Αυτές οι κάμψεις δημιουργούνται στον πρώτο και στον τελευταίο τεχνητό κόμβο που εισάγουμε για κάθε μακριά ακμή γράφου. Σε κάθε έναν από τους ενδιάμεσους τεχνητούς κόμβους δεν δημιουργούνται κάμψεις ακμών, αφού αυτοί αποτελούν ένα υποδέντρο με ένα μόνο κόμβο σε κάθε επίπεδο, και ο αλγόριθμος σχεδιασμού του δέντρου δίνει σε όλους αυτούς τους κόμβους την ίδια συντεταγμένη-y στην περίπτωση που η φορά σχεδιασμού των ακμών είναι από αριστερά προς τα δεξιά.

Εισαγωγή του πρώτου τεχνητού κόμβου σε μακριές ακμές γράφου. Ο πρώτος τεχνητός κόμβος που εισάγουμε σε κάθε μακριά ακμή γράφου, ανήκει στους μη ομαδοποιημένους κόμβους του πατέρα του. Μια πρώτη απλή σκέψη για την τοποθέτηση αυτού του κόμβου είναι να υπολογίσουμε τη Κατεύθυνση αυτού του κόμβου και να τη χρησιμοποιήσουμε για την τοποθέτησή του. Δυστυχώς η Κατεύθυνση κάθε τέτοιου κόμβου δεν μας βοηθάει για την τοποθέτησή του. Η Κατεύθυνση αυτή είναι $\pm C$ (παράγραφος 3.4.1) για όλους τους πρώτους τεχνητούς κόμβους κάθε μακριάς ακμής και το πρόσημο εξαρτάται από τη σχετική του θέση ως προς τον προορισμό της αντίστοιχης ακμής.

Έστω ένας κόμβος v ο οποίος είναι αφετηρία τριών μακριών ακμών γράφου των οποίων οι πρώτοι τεχνητοί κόμβοι έχουν την ίδια Κατεύθυνση. Τότε, σύμφωνα με τη λογική της ταξινόμησης των κόμβων ως προς την Κατεύθυνσή τους η σχετική τους θέση στη τελική τοποθέτηση δεν είναι σημαντική και μπορεί να είναι οποιαδήποτε. Στη πραγματικότητα η σχετική θέση των τριών αυτών κόμβων είναι σημαντική όσον αφορά τη δημιουργία τομών ακμών. Αν υποθέσουμε για παράδειγμα ότι και οι τρείς αυτοί κόμβοι έχουν θετικό πρόσημο στη Κατεύθυνσή τους, τότε οι προορισμοί των ακμών γράφου που τους αντιστοιχούν - σύμφωνα με τα παραδείγματα που έχουμε δώσει σ'αυτό το κεφάλαιο - βρίσκονται στα δεξιά τους και προκειμένου να μην δημιουργηθούν τομές μεταξύ αυτών των ακμών θα έπρεπε οι τρείς κόμβοι να είναι τοποθετημένοι σε μια συγκεκριμένη σειρά (ο κόμβος που αντιστοιχεί στη κοντότερη ακμή πιο δεξιά από τους άλλους δύο). Η λογική του φωλιασμού των πρώτων τεχνητών κόμβων για τις μακριές ακμές γράφου που έχουν αφετηρία ένα κοινό κόμβο είναι παρόμοια με τη λογική του φωλιασμού Δευτερεύοντων κόμβων, και λαμβάνεται υπόψη κατά

την τοποθέτηση του πρώτου τεχνητού κόμβου κάθε μακριάς ακμής.

Για την εύρεση της κατάλληλης θέσης μεταξύ των αδελφιών του για κάθε τεχνητό κόμβο ακολουθούμε και πάλι μια διαδικασία υπολογισμού τομών ακμών. Εδώ, οι θέσεις που έχουμε να εξετάσουμε και να υπολογίσουμε τομές ακμών είναι συνήθως περισσότερες από τις 2 που αντιστοιχούν σε κάθε Δευτερεύοντα κόμβο. Εφόσον δεν μπορούμε να χρησιμοποιήσουμε τη Κατεύθυνση του κόμβου για την τοποθέτησή του, θα πρέπει να ελέγξουμε έναν αριθμό από δυνατές θέσεις. Ξεκινούμε τον έλεγχο από τη πιο ακραία θέση η οποία αντιστοιχεί στο πρόσημο της Κατεύθυνσης του κόμβου, για παράδειγμα από την αριστερότερη θέση αν ο κόμβος έχει αρνητικό πρόσημο. Εξετάζουμε θέσεις έως ότου βρούμε κάποιο κόμβο με Κατεύθυνση 0 ή Κατεύθυνση με αντίθετο πρόσημο. Δεν εξετάζουμε θέσεις οι οποίες είναι μεταξύ μελών ομάδων. Επίσης, δεν εξετάζουμε θέσεις οι οποίες είναι μεταξύ κόμβων με αντίθετο πρόσημο Κατεύθυνσης ή Κατεύθυνση ίση με 0. Αυτό γίνεται για δύο λόγους. Ο πρώτος είναι προφανής και είναι η ελλάτωση του συνολικού χρόνου υπολογισμού της τοποθέτησης. Ο δεύτερος είναι ότι η τοποθέτηση μεταξύ κόμβων με αντίθετο πρόσημο Κατεύθυνσης ή Κατεύθυνση 0 είναι πιο πιθανό να οδηγήσει σε αύξηση των τομών ακμών, παρά σε ελλάτωσή τους. Αυτό γίνεται γιατί είναι πιθανόν οι ακμές γράφου που ξεκινάνε από τα υποδέντρα τέτοιων κόμβων να τέμνονται με την ακμή γράφου που ξεκινάει απο τον τεχνητό κόμβο, μιάς και έχουν αντίθετες σχετικές θέσεις προορισμών και αφετηριών. Επίσης, σε κάθε θέση p που εξετάζουμε πέρα από την αρχική, μεγαλώνει η πιθανότητα να δημιουργηθούν τομές ακμών μεταξύ της ακμής γράφου που ξεκινάει από τον τεχνητό κόμβο και των υποδέντρων με ρίζες κόμβους στις θέσεις μεταξύ της αρχικής και της θέσης p.

Ο υπολογισμός των τομών ακμών που δημιουργούνται από την τοποθέτηση του τεχνητού κόμβου σε κάθε θέση που εξετάζουμε, γίνεται με αντίστοιχο τρόπο με τον υπολογισμό των τομών ακμών στη περίπτωση τοποθέτησης Δευτερεύοντων κόμβων, με τη διαφορά ότι εδώ σε κάθε νέα θέση αναπροσαρμόζουμε κατάλληλα το αντίστοιχο σύνολο του S_{Δ_i}, προσθέτοντας τον κόμβο που βρίσκεται μεταξύ της νέας και της προηγούμενης θέσης.

3.6 Συνοπτική παρουσίαση των βημάτων του αλγορίθμου

Στο σχήμα 3.12 παρουσιάζουμε συνοπτικά τις φάσεις και τα βήματα του αλγορίθμου στα οποία αναφερθήκαμε με περισσότερες λεπτομέρειες σε προηγούμενες παραγράφους. Δίπλα σε κάθε βήμα δίνουμε και τον αριθμό της παραγράφου στην οποία αναπτύσσεται.

1. Προεπεξεργασία του γράφου. (§ 3.2)
2. Φάση 1. (§ 3.3)
3. Για κάθε επόμενο κόμβο v του ζευγνύοντος δέντρου στο τρέχον επίπεδο:

   α. Για κάθε ομάδα παιδιών του v:
   - Βρες τη σχετική θέση των Κύριων κόμβων μεταξύ τους. (§ 3.4.4)
   - Βρες τη σχετική θέση όσων από τους Δευτερεύοντες κόμβους δεν αναβάλλεται η τοποθέτηση. (§ 3.4.5)
   - Δημιούργησε τα γεγονότα αναμονής για όσους από τους Δευτερεύοντες κόμβους αναβάλλεται η τοποθέτηση. (§ 3.4.5)
   - Υπολόγισε τη Κατεύθυνση όλων των μελών και τη συνολική Κατεύθυνση. (§ 3.4.2)
   - Αντίστρεψε τη διάταξη των μελών αν χρειάζεται. (§ 3.4.2)

   β. Για κάθε μη ομαδοποιημένο κόμβο w παιδί του v:
   - Υπολόγισε τη Κατεύθυνση του w. (§ 3.4.1)

   γ. Ταξινόμησε κόμβους και ομάδες κόμβων ως προς τις Κατευθύνσεις τους. (§ 3.4.2)

   δ. Τοποθέτησε τους ταξινομημένους κόμβους στη δομή Tree. (§ 3.4.1)

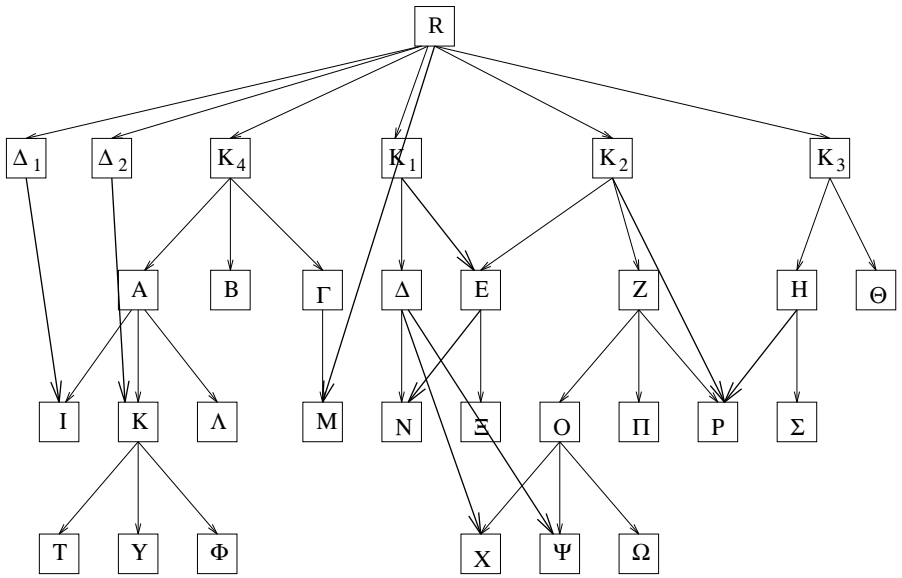
   ε. Αν ο v είναι ο τελευταίος κόμβος στο τρέχον επίπεδο τότε: Για κάθε κόμβο w του τρέχοντος επιπέδου και για κάθε παιδί u του w διευθέτησε τα γεγονότα αναμονής και αν κάποιος κόμβος x που περίμενε μπορεί πλέον να τοποθετηθεί, επανάλαβε το βήμα 3 με επόμενο κόμβο τον x. (§ 3.4.5)
4. Εισαγωγή του πρώτου τεχνητού κόμβου σε κάθε μακριά ακμή γράφου. (§ 3.5)
5. Αναδιάταξη των κόμβων του δέντρου χρησιμοποιώντας τη δομή Tree. (§ 3.5)
6. Τοποθέτηση στο δέντρο των υπόλοιπων τεχνητών κόμβων. (§ 3.5)
7. Σχεδιασμός του δέντρου. (Παραρτ. Α)
8. Σχεδιασμός των τελευταίων τμημάτων κάθε ακμής γράφου. (§ 3.5)

Σχήμα 3.12: Τα βήματα του αλγορίθμου

3.7 Ένα Παράδειγμα Δημιουργίας ενός Σχεδίου από τον Αλγόριθμο

Στη παράγραφο αυτή δείχνουμε μία βήμα προς βήμα κατασκευή ενός σχεδίου γράφου από τον αλγόριθμο. Ο γράφος G του σχήματος 3.13 είναι αυτός που επιλέξαμε για να δείξουμε την ακολουθία βημάτων που εκτελεί ο αλγόριθμος. Το μέγεθος του παραδείγματος που δίνουμε είναι μικρό για να περιέχει λεπτομέρειες σχετικά με όλα τα θέματα που περιγράψαμε παραπάνω, δηλαδή τις ομάδες με αρκετούς Κύριους κόμβους και τη διάταξή τους, τις διάφορες περιπτώσεις δημιουργούμενων τομών ακμών κατά την τοποθέτηση Δευτερεύοντων κόμβων και των πρώτων τεχνητών κόμβων των μακριών ακμών γράφου.

Ο G έχει 31 κόμβους και 39 ακμές από τις οποίες οι 30 που σχεδιάζονται με λεπτές



Σχήμα 3.13: Ενα παράδειγμα γράφου

γραμμές είναι οι ακμές που ανήκουν στο ζευγνύον δέντρο που επιλέχθηκε, ενώ οι υπόλοιπες 9 που σχεδιάζονται με έντονες γραμμές είναι οι ακμές γράφου. Στο σχήμα υπάρχουν 4 τομές ακμών. Το ίδιο σχήμα μπορεί να σχεδιαστεί με μία τομή ακμών, αν αναδιατάξουμε τους κόμβους του. Σημειώνουμε ότι η αρχική κατάσταση που δείχνουμε στο σχήμα 3.13 είναι τυχαία. Προσπαθήσαμε να σχεδιάσουμε μια καλή αρχική κατάσταση έτσι ώστε να είναι φανερή η δομή του γράφου που προσπαθούμε να αναδιατάξουμε εξ'αρχής. Το σχέδιο θα ήταν πολύ χειρότερο αν για παράδειγμα παρεμβάλλαμε το υποδέντρο του κόμβου K_4 μεταξύ των υποδέντρων των κόμβων K_1 και K_2.

3.7.1 Προεπεξεργασία.

Στο στάδιο προεπεξεργασίας του γράφου γίνεται η ανάθεση των κόμβων σε επίπεδα και επιλέγεται το ζευγνύον δέντρο που φαίνεται στο σχήμα 3.13 αν υποθέσουμε ότι αφαιρούμε τις ακμές που έχουν σχεδιαστεί έντονα. Επίσης, υπολογίζεται το σύνολο των ακμών γράφου $GE = \{(\Delta_1,\ \mathrm{I}),\ (\Delta_2,\ \mathrm{K}),\ (R,\ \mathrm{M}),\ (\Delta,\ \mathrm{X}),\ (\Delta,\ \Psi),\ (\mathrm{E},\ \mathrm{N}),\ (\mathrm{K}_1,\ \mathrm{E}),\ (\mathrm{K}_2,\ \mathrm{P}),\ (\mathrm{H},\ \mathrm{P})\}$. Από αυτές τις ακμές οι $(R,\ \mathrm{M})$ και $(\mathrm{K}_2,\ \mathrm{P})$) είναι πρόσθιες ακμές λόγω της επιλογής του συγκεκριμένου ζευγνύοντος δέντρου. Έτσι, για την ακμή $(R,\ \mathrm{M})$ εισάγουμε τον πρώτο της τεχνητό κόμβο #0 και για την $(\mathrm{K}_2,\ \mathrm{P})$ εισάγουμε τον #1. Επίσης, αντικαθιστούμε τις ακμές γράφου $(R,\ \mathrm{M})$ και $(\mathrm{K}_2,\ \mathrm{P})$ στο σύνολο GE με τις νέες ακμές γράφου $(\#0,\ M)$ και $(\#1,\ P)$.

Ακμή	*Χαρακτηριστικά Μονοπάτια*	*Κλωστές*	*Κατασκευή Ομάδων*
(Δ_1, I)	(R, Δ_1, I) (R, K_4, A, I)	(Δ_1, A)	Κατασκευάζει την πρώτη ομάδα O_1 του R. Κάνει τον Δ_1 Δευτερεύοντα κόμβο και τον K_4 Κύριο κόμβο στην O_1.
(Δ_2, K)	(R, Δ_2, K) (R, K_4, A, K)	(Δ_2, A)	Κάνει τον Δ_2 Δευτερεύοντα κόμβο στην O_1.
(#0, M)	(R, #0, M) (R, K_4, Γ, M)	(#0, Γ)	Κάνει τον #0 Δευτερεύοντα κόμβο στην O_1.
(Δ, X)	(R, K_1, Δ, X) (R, K_2, Z, O, X)	(Δ, O) (Δ, Z)	Κατασκευάζει τη δεύτερη ομάδα O_2 του R. Κάνει τον K_1 και τον K_2 Κύριους κόμβους στην O_2.
(Δ, Ψ)	(R, K_1, Δ, Ψ) (R, K_2, Z, O, Ψ)	(Δ, O) (Δ, Z)	
(E, N)	(R, K_2, E, N) (R, K_1, Δ, N)	(E, Δ)	
(K_1, E)	(R, K_1, E) (R, K_2, E)		
(#1, P)	(K_2, #1, P) (K_2, Z, P)		Κατασκευάζει την πρώτη ομάδα O_3 του K_2. Κάνει τον Z Κύριο κόμβο στην O_3 και τον #1 Δευτερεύοντα κόμβο.
(H, P)	(R, K_3, H, P) (R, K_2, Z, P)	(H, Z)	Κάνει τον K_3 Κύριο κόμβο στην O_2.

Πίνακας 3.1: Πληροφορία που συγκεντρώνεται στη πρώτη φάση

3.7.2 Πρώτη Φάση.

Η πληροφορία που υπολογίζεται κατά τη πρώτη φάση του αλγορίθμου, δηλαδή τα χαρακτηριστικά μονοπάτια κάθε ακμής γράφου, οι κλωστές και οι ομάδες κόμβων δίνονται στον πίνακα 3.1. Στον πίνακα 3.2 δίνονται τα στοιχεία των ομάδων κόμβων που δημιουργούνται κατά τη πρώτη φάση.

Ομάδα	*Κύριοι κόμβοι*	*Δευτερεύοντες κόμβοι*	*Ζεύγη κόμβων*
O_1	K_4	Δ_1 Δ_2 #0	(K_4, Δ_1, 2, {I}) (K_4, Δ_2, 2, {K}) (K_4, #0, 2, {M})
O_2	K_1 K_2 K_3		(K_1, K_2, 6, {X, Ψ, N, E}) (K_2, K_3, 1, {P})
O_3	Z	#1	(Z, #1, 1, {P})

Πίνακας 3.2: Οι ομάδες κόμβων που δημιουργούνται κατά τη πρώτη φάση

3.7.3 Δεύτερη Φάση.

Στη συνέχεια περιγράφουμε τα βήματα που εκτελεί ο αλγόριθμος κατά τη διάρκεια της δεύτερης φάσης. Αρχικά υπολογίζεται το πλήθος των ακμών γράφου και δέντρου που υπάρχουν σε κάθε υπογράφο μεταξύ των επιπέδων του. Η πληροφορία αυτή αποθηκεύεται στη δομή SubgraphWidth. Για παράδειγμα ο υπογράφος με ρίζα τον K_1 έχει 2 ακμές μεταξύ των επιπέδων 1 και 2, 5 ακμές μεταξύ των επιπέδων 2 και 3 και 2 ακμές μεταξύ των επιπέδων 3 και 4. Για κάθε τρέχοντα κόμβο περιγράφουμε τα βήματα που εκτελεί ο αλγόριθμος. Μετά το τέλος της εξέτασης των κόμβων ενός επιπέδου, αλλά και με την τοποθέτηση κάθε κόμβου ο οποίος περίμενε την επίλυση κάποιων γεγονότων αναμονής δείχνουμε τα περιεχόμενα της δομής Tree. Αρχικά η δομή Tree είναι η εξής:

Tree[0] = / R /

και Tree[1], Tree[2], Tree[3], Tree[4] είναι κενά.

Οι υπολογισμοί των Κατευθύνσεων κόμβων γίνονται παίρνοντας υπόψη το τελευταίο στιγμιότυπο της δομής Tree. Η υλοποίηση της διάσχισης κατά πλάτος του ζευγνύοντος δέντρου γίνεται μέσω μιας ουράς Q. Αρχικά η ουρά περιέχει μόνο τον κόμβο R. Κάθε φορά που διατάσσουμε τα παιδιά ενός κόμβου τα τοποθετούμε στη Q. Στην Q δεν τοποθετούμε προφανώς κόμβους οι οποίοι δεν έχουν παιδιά. Ο τρέχων κόμβος στη παρακάτω περιγραφή είναι κάθε φορά ο επόμενος κόμβος της Q.

Τρέχων κόμβος = R:

Ομάδα O_1:

Διάταξη των Κύριων κόμβων: τετριμμένη.

Διάταξη των Δευτερεύοντων κόμβων:

Δημιουργία γεγονότων αναμονής: $\Delta_1 \overleftarrow{\longrightarrow} I$, $\Delta_2 \overleftarrow{\longrightarrow} K$, #0 $\overleftarrow{\longrightarrow} M$

Ομάδα O_2:

Διάταξη των Κύριων κόμβων:

Αρχικά διατάσσονται οι K_1 και K_2. Η τοποθέτηση του K_3 πριν τον K_1 δημιουργεί 5 τομές ακμών (τομή της (H, P) με τον υπογράφο $G(K_1)$ μεταξύ των επιπέδων 2 και 3). Η τοποθέτηση του K_3 μεταξύ των K_1 και K_2 δημιουργεί 4 τομές ακμών (τομή της (K_1, E) με τον υπογράφο $G(K_3)$ μεταξύ των επιπέδων 1 και 2 και τομή της (E, N) με τον υπογράφο $\Gamma(K_3)$ μεταξύ των επιπέδων 2 και 3). Η τοποθέτηση του K_3 μετά τον K_2 δεν δημιουργεί τομές ακμών. Επομένως η διάταξη των Κύριων κόμβων είναι: K_1, K_2, K_3.

Διάταξη των Δευτερεύοντων κόμβων: δεν υπάρχουν.

Κατευθύνσεις κόμβων: δεν υπολογίζονται.

Κατευθύνσεις ομάδων: δεν υπολογίζονται.

Ταξινόμηση κόμβων: K_4, K_1, K_2, K_3.

Τέλος κόμβων επιπέδου 0:

Tree[0] = / R /

Tree[1] = / K_4 / K_1 K_2 K_3 /

Tree[2] =

Tree[3] =

Tree[4] =

Τρέχων κόμβος = K_4:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Β) = 0.

Κατεύθυνση(Α) = 0, Δ_Κατεύθυνση(Α) = $2 * C$.

Κατεύθυνση(Γ) = 0, Δ_Κατεύθυνση(Γ) = C.

Ταξινόμηση κόμβων:

Είναι ΚΘΚ=1 και ΚΑΚ=0. Στη Δ_Κατεύθυνση(Α) δίνεται αρνητικό πρόσημο, επομένως ΚΑΚ = 1, Κατεύθυνση(Α) = -2 $* C$ και ΥποθετικήΣχετικηΘεση(Δ_1) = ΥποθετικήΣχετικηΘεση(Δ_2) = ΠΡΙΝ. Στη Δ_Κατεύθυνση(Γ) δίνεται θετικό πρόσημο και επομένως ΚΘΚ = 2 και Κατεύθυνση(Γ) = C. ΥποθετικήΣχετικηΘεση(#0) = ΜΕΤΑ. Τελική ταξινόμηση: Α, Β, Γ.

Τρέχων κόμβος = K_1:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Δ) = $2 * C$.

Ταξινόμηση κόμβων: Δ.

Τρέχων κόμβος = K_2:

Ομάδα O_3:

Διάταξη των Κύριων κόμβων: τετριμμένη.

Διάταξη των Δευτερεύοντων κόμβων:

Δημιουργία γεγονότων αναμονής: #1 $\rightleftarrows$ P

Κατευθύνσεις κόμβων:

Κατεύθυνση(Ε) = $-I - C$.

Κατεύθυνση(Ζ) = $-2 * C + C = -C$.

Κατευθύνσεις ομάδων:

Κατεύθυνση(O_3) = $-C$.

Ταξινόμηση κόμβων: Ε, Ζ.

Τρέχων κόμβος = K_3:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Η) = $-C$.

Κατεύθυνση(Θ) = 0.

Ταξινόμηση κόμβων: Η, Θ.

Τέλος κόμβων επιπέδου 1:

Tree[0] = / R /

Tree[1] = / K_4 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ / Η Θ /

Tree[3] =

Tree[4] =

Τρέχων κόμβος = Α:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Ι) = $-I$ (αφού ΥποθετικήΣχετικηΘεση(Δ_1) = ΠΡΙΝ).

Κατεύθυνση(Κ) = $-$Ι (αφού ΥποθετικήΣχετικηΘεση(Δ_2) = ΠΡΙΝ).

Κατεύθυνση(Λ) = 0.

Ταξινόμηση κόμβων: Ι, Κ, Λ.

Τρέχων κόμβος = Γ:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Μ) = I (αφού ΥποθετικήΣχετικηΘεση(#0) = ΜΕΤΑ).

Ταξινόμηση κόμβων: Μ.

Τρέχων κόμβος = Δ:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Ν) = I

Ταξινόμηση κόμβων: Ν.

Τρέχων κόμβος = Ε:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Ξ) = 0

Ταξινόμηση κόμβων: Ξ.

Τρέχων κόμβος = Z:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Ο) = $2 * C$

Κατεύθυνση(Π) = 0.

Κατεύθυνση(Ρ) = I, Δ_Κατεύθυνση(Ρ) = I.

Ταξινόμηση κόμβων:

Στη Δ_Κατεύθυνση(Ρ) δίνεται θετικό πρόσημο, επομένως Κατεύθυνση(Ρ) = $2 * I$ και ΥποθετικήΣχετικηΘεση(#1) = ΜΕΤΑ. Τελική ταξινόμηση: Ο, Π, Ρ.

Τρέχων κόμβος = Η:

Κατευθύνσεις κόμβων:

Κατεύθυνση(Σ) = 0

Ταξινόμηση κόμβων: Σ.

Τέλος κόμβων επιπέδου 2:

Tree[0] = / R /

Tree[1] = / K_4 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ / Η Θ /

Tree[3] = / Ι Κ Λ / Μ / Ν / Ξ / Ο Π Ρ / Σ /

Tree[4] =

Επίλυση γεγονότων αναμονής που εμπλέκουν κόμβους του επιπέδου 3

Επίλυση του γεγονότος $\Delta_1 \overleftrightarrow{\longleftrightarrow} I$:

Ο Δ_1 μπορεί πλέον να τοποθετηθεί. Στη θέση πρίν τον K_4 αντιστοιχούν 0 τομές ακμών. Στη θέση μετά τον K_4 αντιστοιχούν 3 τομές ακμών (τομές της (Δ_1, Ι) με τις (Γ, Μ), (Α, Λ) και (Α, Κ). Επομένως ο Δ_1 τοποθετείται πρίν τον K_4 στη δομή Tree η οποία γίνεται:

Tree[0] = / R /

Tree[1] = / Δ_1 K_4 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ / Η Θ /

Tree[3] = / Ι Κ Λ / Μ / Ν / Ξ / Ο Π Ρ / Σ /

Tree[4] =

Επίλυση του γεγονότος $\Delta_2 \overleftrightarrow{\longleftrightarrow} K$:

Ο Δ_2 μπορεί πλέον να τοποθετηθεί. Στη θέση πριν τον K_4 και μετά τον Δ_1 (φωλιασμός ακμών γράφου Δευτερεύοντων κόμβων) αντιστοιχεί 1 τομή ακμών. Στη θέση μετά τον K_4 αντιστοιχούν 2 τομές ακμών. Επομένως ο Δ_1 τοποθετείται πρίν τον K_4 στη δομή Tree η οποία γίνεται:

Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 / K_1 K_2 K_3 /

Tree[2] = / A B Γ / Δ / E Z / H Θ /

Tree[3] = / I K Λ / M / N / Ξ / O Π P / Σ /

Tree[4] =

Επίλυση του γεγονότος #0 $\overrightarrow{\longleftarrow}$ M:

Ο #0 μπορεί πλέον να τοποθετηθεί. Στη θέση πριν τον K_4 και μετά τον Δ_2 αντιστοιχούν 3 τομές ακμών. Στη θέση μετά τον K_4 αντιστοιχούν 0 τομές ακμών. Επομένως ο #0 τοποθετείται μετά τον K_4 στη δομή Tree η οποία γίνεται:

Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 **#0** / K_1 K_2 K_3 /

Tree[2] = / A B Γ / Δ / E Z / H Θ /

Tree[3] = / I K Λ / M / N / Ξ / O Π P / Σ /

Tree[4] =

Επίλυση του γεγονότος #1 $\overrightarrow{\longleftarrow}$ P:

Ο #1 μπορεί πλέον να τοποθετηθεί. Στη θέση πριν τον Z αντιστοιχούν 2 τομές ακμών. Στη θέση μετά τον Z αντιστοιχούν 0 τομές ακμών. Επομένως ο #1 τοποθετείται μετά τον Z στη δομή Tree η οποία γίνεται:

Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 #0 / K_1 K_2 K_3 /

Tree[2] = / A B Γ / Δ / E Z **#1** / H Θ /

Tree[3] = / I K Λ / M / N / Ξ / O Π P / Σ /

Tree[4] =

Τρέχων κόμβος = K:

Κατευθύνσεις κόμβων:

Κατεύθυνση(T) = 0

Κατεύθυνση(Y) = 0

Κατεύθυνση(Φ) = 0

Ταξινόμηση κόμβων: T, Y, Φ.

Τρέχων κόμβος = O:

Κατευθύνσεις κόμβων:

Κατεύθυνση(X) = I

Κατεύθυνση(Ψ) = I

Κατεύθυνση(Ω) = 0

Ταξινόμηση κόμβων: Χ, Ψ, Ω.

Τέλος κόμβων επιπέδου 3:

Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 #0 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ #1 / Η Θ /

Tree[3] = / Ι Κ Λ / Μ / Ν / Ξ / Ο Π Ρ / Σ /

Tree[4] = / Τ Υ Φ / Χ Ψ Ω /

3.7.4 Εισαγωγή του πρώτου τεχνητού κόμβου σε μακριές ακμές γράφου.

Εδώ εξετάζουμε μακριές ακμές γράφου οι οποίες δεν είναι πρόσθιες ακμές. Επομένως στο παράδειγμά μας θα εισάγουμε το πρώτο τεχνητό κόμβο #2 της ακμής (Δ, Χ) και τον πρώτο τεχνητό κόμβο #3 της ακμής (Δ, Ψ). Ισχύει η σχέση Κατεύθυνση(#2) = Κατεύθυνση(#3) = C. Η λίστα των αδελφιών αυτών των κόμβων αποτελείται από τον κόμβο Ν ο οποίος έχει Κατεύθυνση Ι. Επομένως εξετάζουμε τις θέσεις πρίν και μετά τον Ν.

Τοποθέτηση του #2. Η τοποθέτηση του #2 πριν τον Ν αντιστοιχεί σε 0 τομές ακμών. Η τοποθέτηση μετά τον Ν αντιστοιχεί σε μία τομή ακμών (τομή της (Δ, Χ) με την (Ε, Ν)). Επομένως ο #2 τοποθετείται πριν τον Ν και η δομή Tree γίνεται:

Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 #0 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ #1 / Η Θ /

Tree[3] = / Ι Κ Λ / Μ / **#2** Ν / Ξ / Ο Π Ρ / Σ /

Tree[4] = / Τ Υ Φ / Χ Ψ Ω /

Τοποθέτηση του #3. Η τοποθέτηση του #3 πριν τον #2 παραβιάζει τις συνθήκες φωλιασμού των ακμών γράφου. Η τοποθέτηση μετά τον #2 και πρίν τον Ν αντιστοιχεί σε 1 τομή ακμής (τομή της (Δ, Ψ) με την (Ο, Χ)). Η τοποθέτηση μετά τον Ν αντιστοιχεί σε 2 τομές ακμών (τομή της (Δ, Ψ) με την (Ο, Χ) και την (Ε, Ν)). Επομένως ο #3 τοποθετείται μεταξύ του #2 και του Ν και η δομή Tree παίρνει τη τελική της μορφή, η οποία είναι:

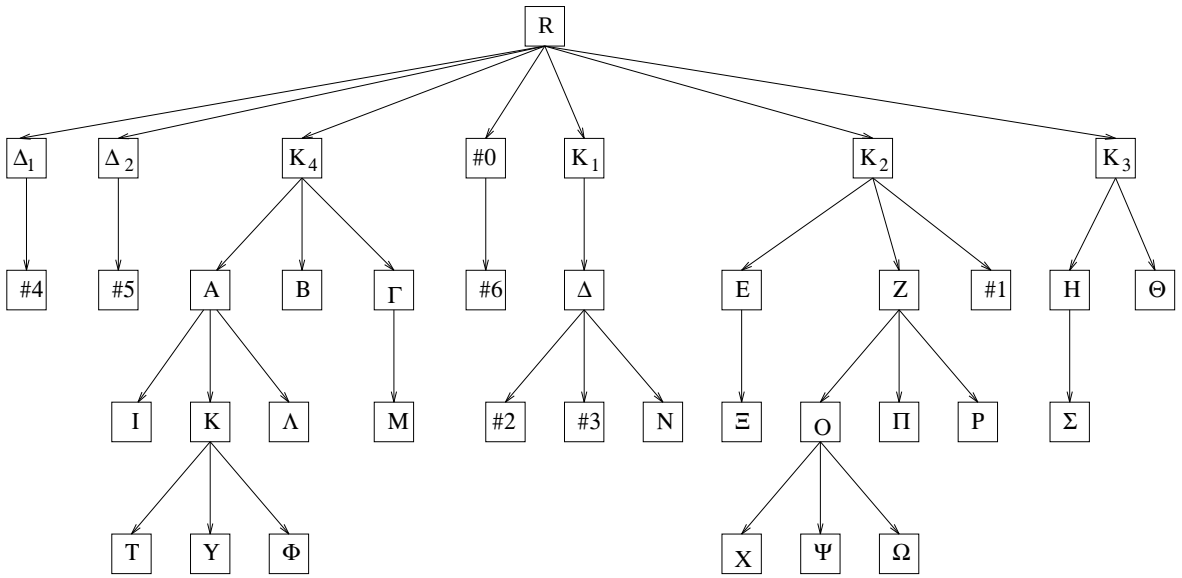
Tree[0] = / R /

Tree[1] = / Δ_1 Δ_2 K_4 #0 / K_1 K_2 K_3 /

Tree[2] = / Α Β Γ / Δ / Ε Ζ #1 / Η Θ /

Tree[3] = / Ι Κ Λ / Μ / #2 **#3** Ν / Ξ / Ο Π Ρ / Σ /

Tree[4] = / Τ Υ Φ / Χ Ψ Ω /



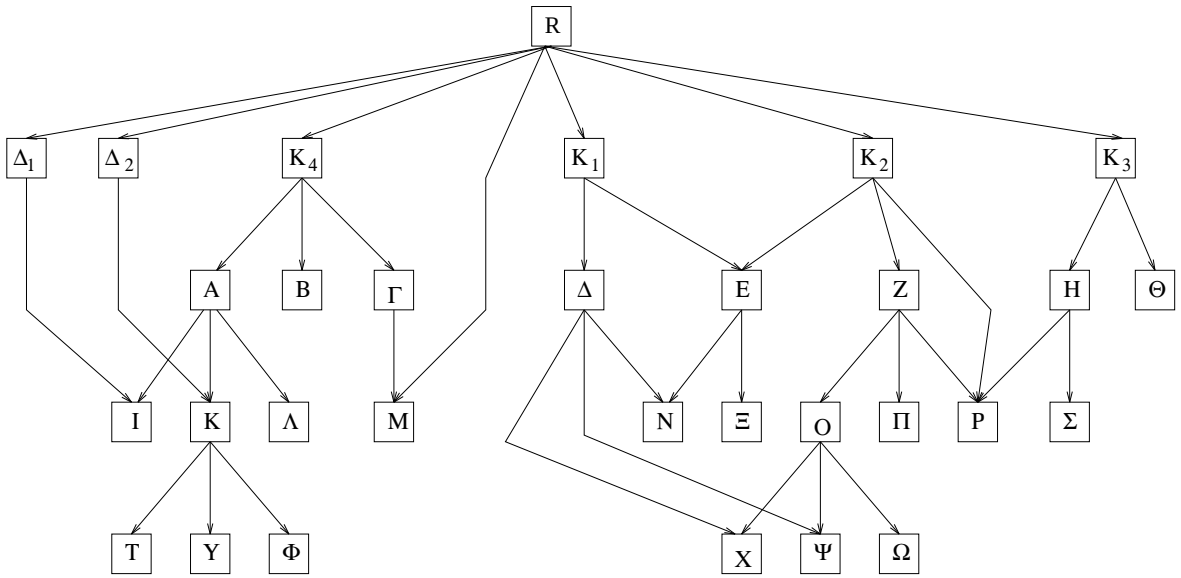
Σχήμα 3.14: Ενα ενδιάμεσο στάδιο στην κατασκευή του σχεδίου του G

3.7.5 Αναδιάταξη του δέντρου και σχεδιασμός

Χρησιμοποιώντας την πληροφορία της δομής Tree, αλλάζουμε τις τιμές των δεικτών που δείχνουν στο διπλανό αδέλφι και το πρώτο παιδί κάθε κόμβου. Παραδείγματος χάριν για το κόμβο R κάνουμε το δείκτη που δείχνει στο πρώτο παιδί του να δείχνει στον κόμβο Δ_1 και αλλάζουμε τους δείκτες των παιδιών του που δείχνουν στο διπλανό αδέλφι έτσι ώστε να είναι: $\Delta_1 \rightarrow \Delta_2 \rightarrow K_4 \rightarrow \#0 \rightarrow K_1 \rightarrow K_2 \rightarrow K_3$.

Εισάγουμε στο δέντρο τους υπόλοιπους τεχνητούς κόμβους που αντιστοιχούν σε μακριές ακμές γράφου. Έτσι, εισάγουμε τον κόμβο #4 στην ακμή $(\Delta_1,\ \text{I})$, τον #5 στην $(\Delta_2,\ \text{K})$, και τον #6 στην (#0, M). Αφού εισάγουμε τους τεχνητούς κόμβους, σχεδιάζουμε το δέντρο χρησιμοποιώντας τον αλγόριθμο του S. Moen. Το σχέδιο του δέντρου φαίνεται στο σχήμα 3.14 όπου φαίνονται και οι τεχνητοί κόμβοι. Παρατηρούμε ότι οι τεχνητοί κόμβοι καταλαμβάνουν τον χώρο στον οποίο θα δρομολογήσουμε τις ακμές γράφου που τους αντιστοιχούν.

Το τελευταίο βήμα του αλγορίθμου είναι ο σχεδιασμός των τελευταίων τμημάτων των ακμών γράφου. Στο σχήμα 3.15 φαίνεται το τελικό σχέδιο του γράφου G.



Σχήμα 3.15: Το τελικό σχέδιο του G

Κεφάλαιο 4

Αξιολόγηση του αλγορίθμου

Το κεφάλαιο αυτό περιγράφει τις μετρήσεις που κάναμε για την αξιολόγηση το αλγορίθμου που σχεδιάσαμε. Στο εξής αναφερόμαστε σ'αυτόν τον αλγόριθμο με το όνομα *ReArrange*, περισσότερο για λόγους οικονομίας κειμένου και όχι επειδή αυτό είναι το καλύτερο όνομα για τον αλγόριθμο που σχεδιάσαμε.

Για να αξιολογήσουμε την απόδοση του αλγορίθμου *ReArrange* χρησιμοποιούμε πέντε αντικειμενικά κριτήρια τα οποία περιγράφουμε στη συνέχεια. Επίσης εξετάζουμε συγκριτικά με την απόδοση του αλγορίθμου *ReArrange* και την απόδοση του αλγορίθμου *STT* ως προς τα ίδια κριτήρια χρησιμοποιώντας ένα σύνολο 12 παραδειγμάτων που ποικίλουν ως προς το μέγεθος και τα χαρακτηριστικά.

Η έκδοση του αλγορίθμου *STT* που χρησιμοποιούμε για τη σύγκριση είναι αυτή που περιγράφεται στο [29], δηλαδή η αρχική έκδοση του αλγορίθμου *STT*. Στη φάση της ελαχιστοποίησης τομών ακμών του *STT* επαναλαμβάνουμε τη Φάση 2 μία φορά και τη Φάση 1 δύο φορές. Προσθέσαμε στην έκδοση αυτή, τη δυνατότητα να χειρίζεται κύκλους και να τοποθετεί κάθε κόμβο στο μέγιστο δυνατό επίπεδο έτσι ώστε να παράγει σχέδια τα οποία να μπορούν να συγκριθούν με αυτά του αλγορίθμου *ReArrange*, αφού μ'αυτό το τρόπο, και για τους δύο αλγορίθμους, θεωρώντας ένα συγκεκριμένο γράφο, το σύνολο των κόμβων κάθε επιπέδου θα είναι το ίδιο. Ο αλγόριθμος *STT* έχει υλοποιηθεί στη γλώσσα C ενώ ο *ReArrange* στη γλώσσα C++.

Όπως είδαμε στο κεφάλαιο 2, κατά τη διάρκεια της δεκαετίας του '80 παρουσιάστηκαν στην βιβλιογραφία διάφορες βελτιώσεις της αρχικής έκδοσης του αλγορίθμου *STT*. Αποφασίσαμε να συγκρίνουμε την αρχική έκδοση του *STT* με τον *ReArrange* κυρίως διότι δεν είχαμε πρόσβαση σε υλοποιήσεις εργασιών που δημοσιεύτηκαν και βελτίωναν τον *STT*. Ήδη η υλοποίηση του *STT* κατανάλωσε αρκετό από το συνολικό χρόνο αυτής της εργασίας, οπότε η υλοποίηση των βελτιώσεων ήταν χρονικά αδύνατη. Η σύγκριση με την αρχική έκδοση

Όνομα Γράφου	*Κόμβοι*	*Ακμές*	*Πυκνότητα*	*Τύπος Γράφου*
mod_call	35	75	6.1	Γράφος κλήσης υποπρογραμμάτων
shortlist	37	63	4.6	Γράφος ιεραρχίας τύπων
world1	43	60	3.2	Χάρτης
world2	43	69	3.7	Χάρτης
SelfPortrait	81	107	1.6	Γράφος κλήσης ρουτινών
pert	99	145	4.5	Διάγραμμα PERT
p104	104	176	1.6	Τυχαίος επίπεδος γράφος
t_hierarchy	166	189	0.7	Γράφος ιεραρχίας τύπων
main1	105	186	1.7	Γράφος κλήσης ρουτινών
aircraft	124	432	2.8	Μηχανή πεπερασμένων καταστάσεων
call_graph1	144	289	1.4	Γράφος κλήσης ρουτινών
main2	178	441	1.4	Γράφος κλήσης ρουτινών
bitalt	188	461	1.3	Μηχανή πεπερασμένων καταστάσεων
call_graph2	259	811	3.2	Γράφος κλήσης ρουτινών
conveyer300	300	711	0.8	Μηχανή πεπερασμένων καταστάσεων
conveyer400	400	1014	0.6	Μηχανή πεπερασμένων καταστάσεων
	Μέση Πυκνότητα		2.5	

Πίνακας 4.1: Το σύνολο των γράφων που χρησιμοποιήθηκαν στις συγκριτικές μετρήσεις

του *STT* δεν κάνει λιγότερο σημαντικές τις μετρήσεις που δίνονται εδώ, αφού η σύγκριση του *ReArrange* με μία από τις διάφορες βελτιωμένες εκδόσεις μπορεί να προκύψει έμμεσα χρησιμοποιώντας τις μετρήσεις που δίνονται για τη βελτιωμένη έκδοση.

Όλες οι μετρήσεις έγιναν σε έναν σταθμό εργασίας SPARCclassic 4.1.3 με 16 Mbytes μνήμη και τοπικό δίσκο με διαθέσιμα 30 Mbytes που χρησιμοποιούνται ως swap space. Το λειτουργικό σύστημα που χρησιμοποιείται είναι το SunOS-4.1.3C.

Το υπόλοιπο αυτού του κεφαλαίου οργανώνεται ως εξής. Στην παράγραφο 4.1 περιγράφουμε το σύνολο των παραδειγμάτων που χρησιμοποιούμε για την αξιολόγηση του αλγορίθμου. Στην παράγραφο 4.2 δίνουμε τα πέντε αντικειμενικά κριτήρια μέσω των οποίων θα αξιολογήσουμε τον αλγόριθμο *ReArrange* και θα τον συγκρίνουμε με τον *STT*. Τέλος η παράγραφος 4.3 συγκρίνει την απόδοση των δύο αλγορίθμων ως προς τα πέντε κριτήρια.

4.1 Το σύνολο των παραδειγμάτων

Τα παραδείγματα γράφων που επιλέξαμε για να συγκρίνουμε τους δύο αλγορίθμους περιγράφονται στον πίνακα 4.1. Για καθε παράδειγμα δίνουμε το όνομα με το οποίο θα

αναφερόμαστε σ'αυτό, το μέγεθός του (αριθμός κόμβων και ακμών), την πυκνότητά[1] του και το είδος του. Επιλέχθηκαν γράφοι των οποίων το μέγεθος ποικίλει από αρκετά μικρό (75 ακμές) έως πολύ μεγάλο (1014 ακμές), ενώ η μέση πυκνότητά τους είναι 2.5 %. Οι γράφοι αυτοί παρουσιάζονται σε διάφορες εφαρμογές του πραγματικού κόσμου, για παράδειγμα γράφοι κλήσεων υπορουτινών, χάρτες, διαγράμματα PERT κ.λ.π. Ο γράφος p104 είναι ένας επίπεδος γράφος που δημιουργήθηκε με τυχαίο τρόπο. Οι γράφοι ποικίλλουν και ως προς διάφορα στοιχεία της δομής τους. Έτσι, υπάρχουν άκυκλοι και κυκλικοί γράφοι, γράφοι με μία ή περισσότερες ρίζες, επίπεδοι και μή επίπεδοι γράφοι.

Ένα μεγάλο ποσοστό από τους γράφους που επιλέξαμε παρουσιάζονται σε προηγούμενες εργασίες. Ο γράφος shortlist είναι από το [14]. Οι γράφοι world1 και world2 είναι παραλλαγές του γράφου World Dynamics που δίνεται στο [15]. Ο γράφος SelfPortrait παρουσιάζεται στο [16]. Ο γράφος pert είναι ένα διάγραμμα PERT το οποίο σχεδιάστηκε με το σύστημα *MacProject* [1]. Τέλος, ο γράφος p104 είναι ένας τυχαίος επίπεδος γράφος που χρησιμοποιείται στο [21]. Από τους υπόλοιπους γράφους του συνόλου, οι call_graph1 και call_graph2 αντιστοιχούν σε γράφους κλήσης υπορουτινών των υλοποιήσεων μας των αλγορίθμων *STT* και *ReArrange*.

Στους πίνακες που δίνονται στη συνέχεια οι δεκαέξι γράφοι του συνόλου έχουν χωριστεί σε δύο ομάδες. Στην πρώτη ομάδα ανήκουν οι μικρότεροι γράφοι ενώ στη δεύτερη οι μεγαλύτεροι. Οι γράφοι της πρώτης ομάδας έχουν το κοινό χαρακτηριστικό ότι ο χρόνος εκτέλεσης του αλγορίθμου *STT* γι'αυτούς είναι μικρότερος από 1 δευτερόλεπτο. Τα σχέδια 11 γράφων του συνόλου και από τους δύο αλγορίθμους δίνονται στο παράρτημα Β.

4.2 Τα κριτήρια σύγκρισης

Προκειμένου να μπορέσουμε να συγκρίνουμε τους δύο αλγορίθμους χρειαζόμαστε ένα σύνολο αντικειμενικών κριτηρίων που να δείχνουν ποσοτικά την ποιότητα των παραγόμενων σχεδίων από κάθε αλγόριθμο. Τα κριτήρια που επιλέξαμε είναι τα εξής:

1. *Χρόνος Εκτέλεσης*. Ο χρόνος για την εκτέλεση του αλγορίθμου στην προαναφερθείσα μηχανή. Αυτός είναι ο χρόνος που χρειάζεται ο αλγόριθμος για να υπολογίσει τις συντεταγμένες χ και ψ των κόμβων, και δεν περιλαμβάνει το χρόνο που χρειάζεται το εργαλείο σχεδιασμού για να σχεδιάσει το γράφο, ούτε και το χρόνο που απαιτείται για την επικοινωνία με το εργαλείο αυτό.

2. *Αριθμός Τομών Ακμών*. Ο αριθμός των δημιουργούμενων τομών ακμών.

[1] Η πυκνότητα ενός γράφου G = (E, V) είναι η ποσότητα $100 \times (E/V^2)$, δηλαδή το ποσοστό των ακμών που εμφανίζονται στο γράφο έναντι του μέγιστου αριθμού ακμών

3. *Συνολικό Μήκος Ακμών.* Το άθροισμα των μηκών όλων των ακμών του γράφου. Η κλίμακα της μέτρησης είναι αυτή που χρησιμοποιεί και το γραφικό εργαλείο με το οποίο σχεδιάζεται ο γράφος. Σ'αυτή τη κλίμακα το ύψος κάθε κόμβου είναι περίπου 0.5cm.

4. *Συνολικός Αριθμός Κάμψεων Ακμών.* Ο αριθμός των σημείων όπου τα τμήματα ακμών αλλάζουν κλίση, οπότε και δημιουργούνται κάμψεις στις ακμές γράφου που τα περιλαμβάνουν.

5. *Μέγιστο Εμβαδόν.* Το εμβαδό αυτό είναι η ποσότητα *μήκος σχεδίου* $\times$ *πλάτος σχεδίου*, επομένως είναι το εμβαδόν του παραλληλογράμμου που περικλείει τους κόμβους και τις ακμές του γράφου. Η πραγματική επιφάνεια που καλύπτει ο γράφος είναι αρκετά μικρότερη.

Αν και τα αντικειμενικά κριτήρια είναι χρήσιμα για την εκτίμηση της ποιότητας ενός σχεδίου γράφου, δεν μπορούν να συλλάβουν με ακρίβεια και να περιγράψουν τις προτιμήσεις των χρηστών. Η σύγκριση που επιχειρούμε εδώ θα ήταν πιο ολοκληρωμένη αν είχαμε εκτελέσει ένα πείραμα όπου θα ζητούσαμε από χρήστες να εκτιμήσουν τα σχέδια που παράγουν οι δύο αλγόριθμοι. Λόγω έλλειψης χρόνου δεν εκτελέσαμε τέτοιο πείραμα αλλά ρωτήσαμε 7 χρήστες να σχολιάσουν σύντομα τα σχέδια που παράγουν οι δύο αλγόριθμοι. Όλοι οι χρήστες εκτίμησαν ότι τα σχέδια του *ReArrange* είναι σε όλους τους γράφους καλύτερα έως πολύ καλύτερα από του *STT*, κυρίως λόγω του ότι δεν περιέχουν πολλές κάμψεις ακμών ενώ αντίθετα σχεδιάζουν τις μακριές ακμές με όσο το δυνατόν μεγαλύτερα ευθύγραμμα τμήματα τα οποία μπορεί κανείς να τα ακολουθήσει με το μάτι, όταν ο γράφος δεν είναι πολύ μεγάλος.

4.3 Συγκριτική παρουσίαση των αλγορίθμων *ReArrange* και *STT*

Για κάθε ένα από τα παραπάνω κριτήρια μας ενδιαφέρει να συγκρίνουμε τις τιμές $X_{ReArrange}$ και X_{STT} που δίνουν οι δύο αλγοριθμοι ως προς το συγκεκριμένο κριτήριο, για κάθε έναν από τους δεκαέξι γράφους του συνόλου που επιλέξαμε. Η σύγκριση για το χρόνο εκτέλεσης των δύο αλγορίθμων γίνεται χρησιμοποιώντας το πηλίκο των δύο τιμών $\frac{X_{STT}}{X_{ReArrange}}$. Το πηλίκο αυτό δείχνει πόσες φορές γρηγορότερος είναι ο *ReArrange* από τον *STT*. Η σύγκριση για τα υπόλοιπα κριτήρια γίνεται χρησιμοποιώντας τη ποσοστιαία μεταβολή της τιμής $X_{ReArrange}$ ως προς την τιμή X_{STT}. Η ποσοστιαία αυτή μεταβολή δίνεται από το πηλίκο $\frac{X_{ReArrange}-X_{STT}}{X_{STT}}$. Μια αρνητική τιμή ποσοστιαίας μεταβολής δείχνει ότι ο *ReArrange* μειώνει τη τιμή του συγκεκριμένου κριτηρίου, ενώ μια θετική τιμή δείχνει ότι την αυξάνει.

Όνομα Γράφου	*STT*	*ReArrange*	Φορές Γρηγορότερα
mod_call	0.14	0.23	0.61
shortlist	0.12	0.27	0.44
world1	0.08	0.18	0.44
world2	0.24	0.33	0.73
SelfPortrait	0.30	0.26	1.15
pert	0.70	4.03	0.17
p104	0.24	0.77	0.31
t_hierarchy	0.34	0.45	0.76
		Μέσος όρος	0.58
main1	1.46	0.74	1.97
call_graph1	11.4	2.07	5.51
main2	42.6	3.33	12.79
aircraft	55.9	6.50	8.60
bitalt	146.9	8.39	17.51
conveyer300	106.2	9.1	11.67
call_graph2	212.7	16.9	12.58
conveyer400	489.4	16.6	29.48
		Μέσος όρος	12.51

Πίνακας 4.2: Χρόνοι Εκτέλεσης (σε δευτερόλεπτα).

Ο λόγος που κάνουμε με διαφορετικό τρόπο τη σύγκριση των χρόνων εκτέλεσης είναι το ότι μία ποσοστιαία μεταβολή στο χρόνο εκτέλεσης δεν δίνει άμεση πληροφορία για το πόσες φορές γρηγορότερος είναι ο ένας αλγόριθμος σε σχέση με τον άλλο, το οποίο είναι αυτό που μας ενδιαφέρει να δείξουμε.

Χρόνος Εκτέλεσης. Οι χρόνοι εκτέλεσης των δύο αλγορίθμων φαίνονται στον πίνακα 4.2. Οι χρόνοι αυτοί είναι ο μέσος όρος από 20 συνολικά εκτελέσεις του κάθε αλγορίθμου. Οι χρόνοι εκτέλεσης των δύο αλγορίθμων μεταβάλλονται διαφορετικά καθώς το μέγεθος των γράφων αυξάνει. Αποφασίσαμε να μελετήσουμε τους χρόνους εκτέλεσης χωρίζοντας τους γράφους σε δύο ομάδες.

Στην πρώτη ομάδα όλοι οι χρόνοι (με εξαίρεση το χρόνο του *ReArrange* για το γράφο pert) είναι μικρότεροι από 1 δευτερόλεπτο. Οι χρόνοι αυτοί είναι τόσο μικροί ώστε το γεγονός ότι ένας αλγόριθμος θέλει το μισό χρόνο εκτέλεσης σε σχέση με τον άλλο δεν είναι ιδιαίτερα σημαντικό. Στο σύνολο των 8 παραδειγμάτων βλέπουμε ότι ο *ReArrange* είναι περίπου δύο φορές αργότερος από τον *STT*. Κύρια αιτία γι'αυτό το αποτέλεσμα είναι το γεγονός ότι ο *ReArrange* χρησιμοποιεί πιο πολύπλοκες δομές οι οποίες χρειάζονται και περισσότερο χρόνο αρχικοποίησης. Επίσης αρκετές από αυτές τις δομές αρχικοποιούνται πριν την εκτέλεση του αλγορίθμου ενώ κατά τη διάρκεια της εκτέλεσης χρησιμοποιείται τμήμα μόνο των

Όνομα Γράφου	*STT*	*ReArrange*	Μεταβολή (%)
mod_call	76	46	-39
shortlist	38	45	18
world1	25	32	28
world2	39	54	38
SelfPortrait	27	26	-3.7
pert	33	59	79
p104	23	14	-39
t_hierarchy	13	12	-7.7
main1	545	527	-3.3
aircraft	2072	2231	7.7
call_graph1	1760	2310	31
main2	4559	4649	2
bitalt	2175	1420	-35
call_graph2	27522	30916	12
conveyer300	6788	10468	54
conveyer400	16198	26394	63
		Μέση Μεταβολή	13%

Πίνακας 4.3: Αριθμός Τομών Ακμών

περιεχομένων τους. Στην περίπτωση ενός μικρού γράφου το κόστος αυτής της μη αναγκαίας αρχικοποίησης είναι αρκετά μεγάλο, αφού ο συνολικός χρόνος εκτέλεσης είναι μικρός. Με μια πιο προσεκτική σχεδίαση των δομών που χρησιμοποιεί ο *ReArrange*, περιμένουμε ότι η διαφορά μεταξύ του *STT* και του *ReArrange* στους χρόνους εκτέλεσης θα εξομαλυνθεί.

Στο σύνολο των 8 μεγαλύτερων γράφων ο *ReArrange* έχει φανερά καλύτερη απόδοση από τον *STT*. Βλέπουμε ότι ο χρόνος εκτέλεσης του *STT* αυξάνει δραματικά καθώς μεγαλώνει το μέγεθος του γράφου, ενώ αντίθετα οι χρόνοι του *ReArrange* αυξάνονται πιο ομαλά. Στο σύνολο αυτών των παραδειγμάτων ο *ReArrange* είναι κατά μέσο όρο 13 φορές περίπου γρηγορότερος. Οι χρόνοι εκτέλεσης του *ReArrange* δείχνουν ότι πετύχαμε τον αρχικό μας στόχο, δηλαδή την επίτευξη μικρών χρόνων σχεδίασης ακόμα και για μεγάλους γράφους. Κανένας από τους μεγάλους γράφους δεν χρειάζεται παραπάνω από 17 δευτερόλεπτα για να σχεδιαστεί.

Αριθμός Τομών Ακμών. Στο πίνακα 4.3 συγκρίνουμε τους δύο αλγορίθμους ως πρός τον αριθμό των δημιουργούμενων τομών ακμών. Βλέπουμε ότι στο σύνολο των δεκαέξι παραδειγμάτων ο *ReArrange* δημιουργεί κατά μέσο όρο 13 % περισσότερες τομές ακμών έναντι του *STT*. Η αύξηση των τομών ακμών κατά 13 % περίπου είναι αρκετά τυπικό φαινόμενο της συμπεριφοράς του *ReArrange* ο οποίος δεν καταφέρνει να έχει τα ίδια καλά αποτελέσματα

Όνομα Γράφου	*STT*	*ReArrange*	Μεταβολή (%)
mod_call	723	760	5.1
shortlist	577	597	3.5
world1	372	388	4.3
world2	643	920	43
SelfPortrait	689	520	-25
pert	789	798	1.1
p104	1229	1225	-0.3
t_hierarchy	2595	1706	-34
main1	3233	3905	21
aircraft	87997	24698	-72
call_graph1	10367	12926	25
main2	18951	16380	-14
bitalt	36417	21183	-42
call_graph2	68976	65299	-5.3
conveyer300	98186	40202	-59
conveyer400	230649	77825	-66
		Μέση Μεταβολή	-13%

Πίνακας 4.4: Συνολικό Μήκος Ακμών (σε cm).

με τον *STT* ως προς την μείωση των τομών ακμών. Αν και στο σύνολο των παραδειγμάτων ο *ReArrange* δημιουργεί περισσότερες τομές ακμών, υπάρχουν και περιπτώσεις όπως στους γράφους mod_call, p104 και bitalt όπου παρουσιάζει σημαντική βελτίωση. Είναι σημαντικό πάντως το γεγονός ότι η αύξηση των τομών ακμών από τον *ReArrange* δεν είναι αρκετά μεγάλη.

Συνολικό Μήκος Ακμών. Το συνολικό μήκος ακμών ενός γράφου είναι ενδεικτικό του πόσο χώρο καταλαμβάνει το σχέδιο του γράφου (περισσότερος χώρος σχεδίασης δημιουργεί εν γένει ακμές με μεγαλύτερο μήκος), και ακόμη του αριθμού των κάμψεων ακμών (όσο περισσότερο τεθλασμένη σχεδιάζεται μια γραμμή τόσο μεγαλύτερο μήκος έχει). Επομένως βλέπουμε ότι τα κριτήρια 2, 3 και 5 είναι αλληλένδετα. Γενικά μικρότερες τιμές σ'αυτά τα κριτήρια αντιστοιχούν και σε καλύτερα σχέδια, αν και αυτό δεν είναι κανόνας αφού η υπερβολική οικονομία χώρου δεν είναι το ζητούμενο για τους αλγορίθμους που εξετάζουμε εδώ.

Στον πίνακα 4.4 συγκρίνουμε τους δύο αλγορίθμους ως προς το συνολικό μήκος ακμών των δεκαέξι γράφων. Ο *ReArrange* παρουσιάζει μια ποσοστιαία βελτίωση της τάξης του 13 %, η οποία οφείλεται κυρίως στο γεγονός ότι δημιουργεί μικρότερο αριθμό κάμψεων ακμών από τον *STT*.

Όνομα Γράφου	*STT*	*ReArrange*	Μεταβολή (%)
mod_call	43	43	0
shortlist	52	40	-23
world1	24	15	-38
world2	78	36	-54
SelfPortrait	31	31	0
pert	27	19	-30
p104	120	102	-15
t_hierarchy	62	23	-63
main1	190	124	-35
aircraft	2244	365	-84
call_graph1	536	241	-55
main2	652	372	-43
bitalt	2428	416	-83
call_graph2	978	703	-28
conveyer300	1942	538	-72
conveyer400	3082	814	-74
		Μέση Μεταβολή	-44%

Πίνακας 4.5: Αριθμός Κάμψεων Ακμών

Αριθμός Κάμψεων Ακμών. Στον πίνακα 4.5 δείχνουμε τον αριθμό των κάμψεων ακμών που δημιουργούν οι δύο αλγόριθμοι για κάθε γράφο του συνόλου. Όπως είπαμε και στο τρίτο κεφάλαιο ο *ReArrange* δημιουργεί το πολύ δύο κάμψεις ανά μακριά ακμή γράφου και σ'αυτό οφείλεται η πολύ καλή του συμπεριφορά ως προς αυτό το κριτήριο. Συνολικά παρουσιάζει βελτίωση 44 % έναντι του *STT*.

Μέγιστο Εμβαδόν. Στο πίνακα 4.6 φαίνεται το εμβαδόν του περικλείει κάθε έναν από τους δεκαέξι γράφους του συνόλου, στα σχέδια που παράγουν οι δύο αλγόριθμοι. Εδώ, αναμένουμε ο *ReArrange* να έχει χειρότερη απόδοση από τον *STT*, εφόσον ο *ReArrange* καταναλώνει περισσότερο χώρο προκειμένου να σχεδιάζει τις ακμές του γράφου με λίγες κάμψεις. Συνολικά ο *ReArrange* καταναλώνει περίπου 9 % περισσότερο χώρο από τον *STT* για να σχεδιάσει έναν γράφο του συνόλου που δίνεται εδώ. Αν και όπως δείχνουν οι γράφοι aircraft, bitalt, conveyer300 και conveyer400, είναι δυνατόν ο *ReArrange* να χρησιμοποιήσει λιγότερο χώρο από τον *STT* για τη σχεδίαση του ίδιου γράφου, στη γενική περίπτωση ο *ReArrange* καταναλώνει περισσότερο χώρο για το σχεδιασμό ενός γράφου.

Όνομα Γράφου	*STT*	*ReArrange*	Μεταβολή (%)
mod_call	830	724	-13
shortlist	676	871	29
world1	372	473	27
world2	740	1076	45
SelfPortrait	689	642	-6.8
pert	705	943	34
p104	1362	2626	93
t_hierarchy	9520	6377	33
main1	3171	4521	43
aircraft	238800	25416	-89
call_graph1	5965	10570	77
main2	7883	11820	50
bitalt	80710	28389	-65
call_graph2	24163	33553	39
conveyer300	90226	30401	-66
conveyer400	696783	46559	-93
		Μέση Μεταβολή	9%

Πίνακας 4.6: Μέγιστο Εμβαδό (σε cm^2).

Κεφάλαιο 5

Συμπεράσματα και Μελλοντική Εργασία

5.1 Συμπεράσματα

Οι γράφοι είναι ένα χρήσιμο μέσο αναπαράστασης πολλών τύπων πληροφορίας. Η ανάγκη να σχεδιάσουμε σχετικά μεγάλους γράφους ή γράφους που παράγονται αυτόματα μέσω κάποιας εφαρμογής επιβάλλει τη χρησιμοποίηση αλγορίθμων που να σχεδιάζουν γράφους αυτόματα. Κατά τη διάρκεια των τριών τελευταίων δεκαετιών έχουν παρουσιαστεί στη βιβλιογραφία διάφοροι αλγόριθμοι σχεδιασμού γράφων, οι περισσότεροι από τους οποίους όμως ασχολούνται με το σχεδιασμό γράφων ειδικής μορφής και ανάλογα με την εφαρμογή που τους χρησιμοποιεί.

Ένας από τους γνωστότερους αλγορίθμους της προηγούμενης δεκαετίας είναι ο αλγόριθμος των Sugiyama, Tagawa και Toda, ή αλγόριθμος STT. Αρχικά υλοποιήσαμε τον STT, και διαπιστώσαμε τα ελαττώματά του για την εφαρμογή μας: μεγάλος χρόνος εκτέλεσης, πολλές κάμψεις ακμών. Ματά απ'αυτό αναπτύξαμε ένα νέο αλγόριθμο προκειμένου να ξεπεράσουμε τα προβλήματα του STT τον οποίο ονομάσαμε ReArrange. Ο ReArrange, όπως και ο STT, ανήκουν στην κατηγορία αλγορίθμων σχεδιασμού γράφων οι οποίοι προσπαθούν να σχεδιάσουν γράφους που δεν υπάγονται σε συγκεκριμένη κατηγορία, αλλά είναι γενικοί προσανατολισμένοι γράφοι. Προκειμένου να είναι εύκολο να κατανοηθεί το περιεχόμενο των σχεδίων γράφων, αυτοί οι αλγόριθμοι δημιουργούν σχέδια που υπακούουν σε ένα σύνολο περιορισμών (ιεραρχική σχεδίαση, ελάχιστος αριθμός τομών ακμών, ελάχιστο συνολικό μήκος ακμών κ.α.). Οι περιορισμοί αυτοί απηχούν την αντίληψη των χρηστών σχετικά με το τι είναι ένα καλό σχέδιο γράφου.

Σ'αυτή την εργασία προσεγγίζουμε το πρόβλημα του σχεδιασμού γράφων από μια εντελώς

διαφορετική σκοπιά από αυτή του αλγορίθμου STT και των αλγορίθμων που βασίζονται στον STT και βελτιώνουν την απόδοσή του. Δείξαμε ότι είναι δυνατόν να σχεδιάσουμε ένα γράφο αν επιλέξουμε ένα κατάλληλο ζευγνύον δέντρο του γράφου και σε κάθε κόμβο αυτού του δέντρου αναδιατάξουμε κατάλληλα τα παιδιά του. Η δομή του αλγορίθμου επιτρέπει την επίτευξη μικρών χρόνων σχεδιασμού, οι οποίοι μπορεί να είναι έως και δέκα φορές μικρότεροι από τους αντίστοιχους του STT στην περίπτωση μεγάλων γράφων.

Η εργασία πετυχαίνει τον αρχικό στόχο της που ήταν η πολύ γρήγορη σχεδίαση ακόμα και για μεγάλους γράφους. Ταυτόχρονα, ο αλγόριθμος έχει πολύ καλές επιδόσεις ως προς τη δημιουργία κάμψεων ακμών ενώ ως προς άλλα κριτήρια όπως ο αριθμός δημιουργούμενων τομών ακμών, οι επιδόσεις διατηρούνται χοντρικά στα ίδια επίπεδα με τις επιδόσεις του STT. Ο αλγόριθμος που σχεδιάσαμε τείνει να δημιουργεί σχέδια με λίγο μεγαλύτερη επιφάνεια από αυτήν των αντίστοιχων σχεδίων του STT αλλά με μικρότερο συνολικό μήκος ακμών, αφού φροντίζει να ελαχιστοποιεί τον αριθμό των κάμψεων ακμών και να σχεδιάζει με ευθείες όσο το δυνατόν μεγαλύτερα τμήματα ακμών. Μια μικρή αύξηση στην επιφάνεια του σχεδίου είναι ένα μικρό τίμημα όταν αποφεύγονται κατ'αυτό το τρόπο αρκετές αντιαισθητικές κάμψεις ακμών.

Επίσης ένα πλεονέκτημα τους αλγορίθμου ReArrange έναντι του STT είναι ότι έχει καλύτερα αποτελέσματα όταν ο γράφος είναι δέντρο.

5.2 Βελτιώσεις και Επεκτάσεις

Ο αλγόριθμος που υλοποιήσαμε σ'αυτή την εργασία δεν είναι παρά μια πρώτη υλοποίηση, που δείχνει ότι η αναδιάταξη των κόμβων του γράφου με το τρόπο που προτείνει ο ReArrange έχει καλά αποτελέσματα. Στον αλγόριθμό μας, στην τωρινή του μορφή, υπάρχουν αρκετά σημεία που θα χρειάζονταν διόρθωση προκειμένου να βελτιωθεί ακόμη περισσότερο η απόδοσή του.

Είδαμε στο κεφάλαιο 4 ότι ο αλγόριθμος ReArrange έχει μεγαλύτερο χρόνο εκτέλεσης από τον STT στην περίπτωση των μικρών γράφων του συνόλου των παραδειγμάτων. Η κύρια αιτία γι'αυτή τη διαφορά των χρόνων είναι η διαφορά στις δομές που χρησιμοποιούνται στους δύο αλγορίθμους. Ο ReArrange χρησιμοποιεί πιο πολύπλοκες δομές, οι οποίες χρειάζονται περισσότερο χρόνο για αρχικοποίηση αφ'ενός και ενημέρωση αφ'ετέρου κατά τη διάρκεια της εκτέλεσης του αλγορίθμου. Επιπλέον, ο ReArrange χρησιμοποιεί δομές όπως την δομή Subtree οι οποίες αρχικοποιούνται πριν την εκτέλεση του αλγορίθμου, ενώ κατά την εκτέλεση του αλγορίθμου μπορεί να μην χρησιμοποιηθούν καθόλου ή να χρησιμοποιηθεί τμήμα μόνο των περιεχομένων τους. Επομένως μια επόμενη έκδοση του αλγορίθμου θα πρέπει να φροντίζει

να μην υπολογίζει περισσότερη πληροφορία από αυτή που πραγματικά θα χρειαστεί ο αλγόριθμος.

Κατά τη περιγραφή του αλγορίθμου μας, είδαμε ότι παρουσιάζεται συχνά η ανάγκη για μέτρηση τομών ακμών, είτε όταν πρόκειται για διάταξη των μελών ομάδων είτε όταν πρόκειται για την τοποθέτηση τεχνητών κόμβων. Ο υπολογισμός των δημιουργούμενων τομών ακμών από την τοποθέτηση ενός κόμβου σε μια συγκεκριμένη θέση είναι αρκετά χρονοβόρα διαδικασία. Η επόμενη έκδοση του αλγορίθμου ReArrange θα πρέπει να βρεί ένα καλύτερο τρόπο για τον υπολογισμό των τομών ακμών, ο οποίος δεν θα ακολουθεί την χρονοβόρα διαδικασία της σύγκρισης σχετικών θέσεων κόμβων.

Η εύρεση ενός εναλλακτικού τρόπου σύνδεσης των χαρακτηριστικών μονοπατιών μιάς ακμής γράφου μέσω κλωστών, ο οποίος θα ελλάτωνε το χρόνο υπολογισμού των Κατευθύνσεων κόμβων, θα βελτίωνε σημαντικά το χρόνο εκτέλεσης του αλγορίθμου, αφού ο υπολογισμός Κατευθύνσεων κόμβων είναι μια διαδικασία που εκτελείται συχνά.

Η σχέση μεταξύ του βάρους που δίνεται σε ακμές γράφου και του βάρους που δίνεται σε μια κλωστή είναι σημαντική, και μπορεί να επηρεάσει την σχετική τοποθέτηση κόμβων. Πρέπει να μελετηθεί περισσότερο το πώς επιδρά η σχέση μεταξύ αυτών των βαρών στη δημιουργία τομών ακμών, και πιθανότατα να χρειαστεί η δημιουργία ενός πιο πολύπλοκου μηχανισμού ανάθεσης βαρών σε κλωστές.

Κατά τη τοποθέτηση του πρώτου τεχνητού κόμβου σε μία ακμή γράφου, εξετάζουμε τη τοποθέτηση του τεχνητού κόμβου σε όλες τις δυνατές θέσεις μεταξύ αδελφιών του με ίδιο πρόσημο στη Κατεύθυνση. Όπως είναι αναμενόμενο, η διαδικασία αυτή μπορεί να είναι χρονοβόρα. Πρέπει να βρεθούν ευρηματικοί κανόνες αποκλεισμού κάποιων απο τις δυνατές θέσεις για την τοποθέτηση ενός κόμβου. Η σχετική τοποθέτηση δύο πρώτων τεχνητών κόμβων (για δύο ακμές γράφου) οι οποίοι είναι αδέρφια, δεν μπορεί να αποφασιστεί άμεσα αφού αυτοί μπορεί να έχουν ίδιες Κατευθύνσεις. Το ίδιο πρόβλημα όμως έχουμε και για οποιουσδήποτε άλλους κόμβους οι οποίοι είναι αδέρφια και έχουν ίδιες Κατεύθύνσεις. Στην τρέχουσα υλοποίηση η σχετική τοποθέτηση για δύο τέτοιους κόμβους είναι τυχαία. Παρόλ'αυτά θα πρέπει να βρεθούν ευρηματικοί τρόποι εξέτασης όλων των σχετικών τοποθετήσεων και επιλογή αυτής που είναι η καλύτερη, χρησιμοποιώντας ως κριτήριο τις δημιουργούμενες τομές ακμών.

Όπως είπαμε και στο κεφάλαιο 3 η επιλογή του ζευγνύοντος δέντρου το οποίο χρησιμοποιούμε για την αναδιάταξη των κόμβων επηρεάζει την απόδοση του αλγορίθμου. Ο τρόπος επιλογής των παιδιών ενός κόμβου πρέπει να γίνει με κριτήρια που θα εξασφαλίζουν αργότερα μειωμένο αριθμό τομών ακμών.

Τέλος, πρέπει να μελετηθούν θέματα όπως: η υποστήριξη από τον αλγόριθμο περιορισμών

που δίνονται από τον χρήστη σχετικά με τοποθέτηση κόμβων ή φορά ακμών, ο σχεδιασμός δυναμικά μεταβαλλόμενων γράφων (χειρισμός εισαγωγών και διαγραφών κόμβων ή ακμών) κ.α.

Παράρτημα Α

Ο αλγόριθμος για σχεδιασμό δέντρων του S. Moen

Στο παράρτημα αυτό δίνεται μια εκτεταμένη περιγραφή του αλγορίθμου του S. Moen καθώς και ορισμένες από τις ρουτίνες που τον υλοποιούν. Ο αλγόριθμος αυτός χρησιμοποιήθηκε στην εργασία αυτή για το σχεδιασμό του ζευγνύοντος δέντρου του γράφου (παράγραφος 3.1).

Ο αλγόριθμος αυτός έχει τα εξής χαρακτηριστικά :

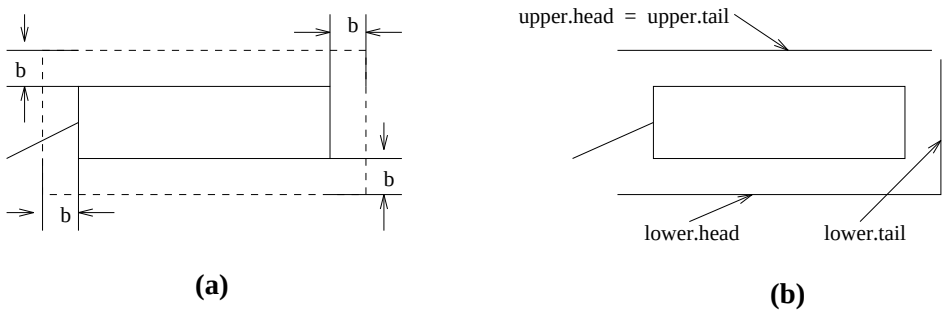
- γραμμικό χρόνο εκτέλεσης,
- επιτρέπει κομβους με μεταβλητό μήκος και τύπο,
- μπορεί να χρησιμοποιηθεί σε εφαρμογές που μεταβάλλουν δυναμικά τα δέντρα που επεξεργάζονται,
- δημιουργεί σχέδια με μικρό εμβαδόν, χρησιμοποιώντας γεωμετρικές τεχνικές,
- τα σχέδια που παράγει έχουν προσανατολισμό από αριστερά προς τα δεξιά (η ρίζα του δέντρου είναι ο αριστερότερος κόμβος) και
- εύκολη υλοποίηση.

Παρακάτω δίνονται οι βασικότερες δομές και ρουτίνες που χρησιμοποιεί ο αλγόριθμος καθώς και οι επεξηγήσεις τους. Επειδή σ'αυτή την εργασία δεν ασχοληθήκαμε με το θέμα των λειτουργιών ενημέρωσης σε ένα γράφο (εισαγωγή/διαγραφή κόμβου κ.α.) δεν δίνονται παρακάτω οι αντίστοιχες ρουτίνες του αλγορίθμου οι οποίες υλοποιούν τις λειτουργίες ενημέρωσης σε ένα δέντρο. Παρ'όλα αυτά ο αλγόριθμος του S. Moen υποστηρίζει πλήρως αυτές τις λειτουργίες με απλό και αποδοτικό τρόπο.

```
/* Δηλώσεις τύπων για γραμμές, πολύγονα και κόμβους */
typedef struct line {
        short dx, dy;
        struct line *link;
} *polyline;
typedef struct polygon {
        struct {
                polyline head, tail;
        } upper, lower;
};
typedef struct tnode {
        char *label;
        struct tnode *parent, *child, *sibling;
        int width, height, border;
        struct { short x, y; } pos, offset;
        struct polygon contour;
} *tree;
/* layout -- η βασική ρουτίνα */
void layout(tree t) {
        tree c;
        for(c = t->child; c; c = c->sibling)
                layout(c);
        if (t->child) attach_parent(t, Join(t));
        else layout_leaf(t);
}
/* layout_leaf -- υπολογίζει τη περιφέρεια ενός φύλλου */
void layout_leaf(tree t) {
        t->contour.upper.tail =
            line(t->width + 2*t->border, 0, 0);
        t->contour.upper.head = t->contour.upper.tail;
        t->contour.lower.tail =
            line(0, -t->height - 2*t->border, 0);
        t->contour.lower.head =
            line(t->width + 2*t->border, 0, t->contour.lower.tail);
}
/* attach_parent -- επεκτείνει το περίγραμμα ενός κλαδιού έτσι
ώστε να περικλείει το περίγραμμα του πατέρα */
void attach_parent(tree t, int h) {
        int x, y1, y2;
        x = t->border + parent_distance;
        y2 = (h - t->height)/2 - t->border;
        y1 = y2 + t->height + 2*t->border - h;
        t->child->offset.x = t->width + x;
        t->child->offset.y = y1;
        t->contour->upper.head =
            line(t->width, 0, line(x, y1, t->contour->upper.head));
        t->contour->lower.head =
            line(t->width, 0, line(x, y2, t->contour->lower.head));
}
/* join -- συνενώνει τα περιγράμματα των παιδιών */
int join(tree t) {
        tree c;
        int d, h, sum;
        c = t->child;
        t->contour = c->contour;
        sum = h = c->height + 2*c->border;
        c = c->sibling;
        while ( c ) {
                d = merge(&t->contour, &c->contour);
                c->offset.y = d + h;
                c->offset.x = 0;
                h = c->height + 2*c->border;
                sum += d + h;
                c = c->sibling;
        }
        return sum;
}
```

Αρχικά καθορίζονται τα πεδία *ύψος* (height), *πλάτος* (width) και *περιθώριο* (border) για κάθε κόμβο. Στη συνέχεια, η βασική ρουτίνα του σχεδιασμού η `layout()`, δημιουργεί τα πολύγωνα που περικλείουν κάθε υποδέντρο, και τοποθετεί κάθε κόμβο του δέντρου. Στη πραγματικότητα ο αλγόριθμος δεν υπολογίζει απόλυτες συντεταγμένες για κάθε κόμβο αλλά υπολογίζει τη θέση κάθε κόμβου σε *σχετικά* με τη θέση του πατέρα του. Η σχετική αυτή θέση αποθηκεύεται στο πεδίο `offset`. Ετσι οι απόλυτες θέσεις όλων των κόμβων μπορούν να υπολογιστούν αν τοποθετήσουμε τη ρίζα του δέντρου σε συγκεκριμένες συντεταγμένες χψ. Ο αλγόριθμος αποτελείται από τέσσερεις κύριες ρουτίνες οι οποίες δίνονται εδώ σε κώδικα C. Η βασική ρουτίνα είναι η `layout()` η οποία καλεί δύο άλλες, τις `attach_parent()` `layout_leaf()`. Η `layout_leaf()` είναι ειδική περίπτωση της `layout()` η οποία χρησιμοποιείται όταν ο τρέχων κόμβος στην `layout()` είναι φύλλο. Παρακάτω επεξηγούνται οι βασικές ρουτίνες.

Κατασκεύη πολυγώνου φύλλου . Εάν ο τρέχων κόμβος είναι φύλλο τότε η κατασκευή του πολυγώνου που το περικλείει είναι τετριμμένη και η `layout()` καλεί την `layout_leaf()` γι'αυτή τη κατασκευή. Στο σχήμα Α.1 φαίνεται πως κατασκευάζει η `layout_leaf()` το



Σχήμα Α.1: (a) Ενα φύλλο και το περιθώριό του, (b) το πολύγωνο ενός φύλλου

πολύγωνο ενός φύλλου.

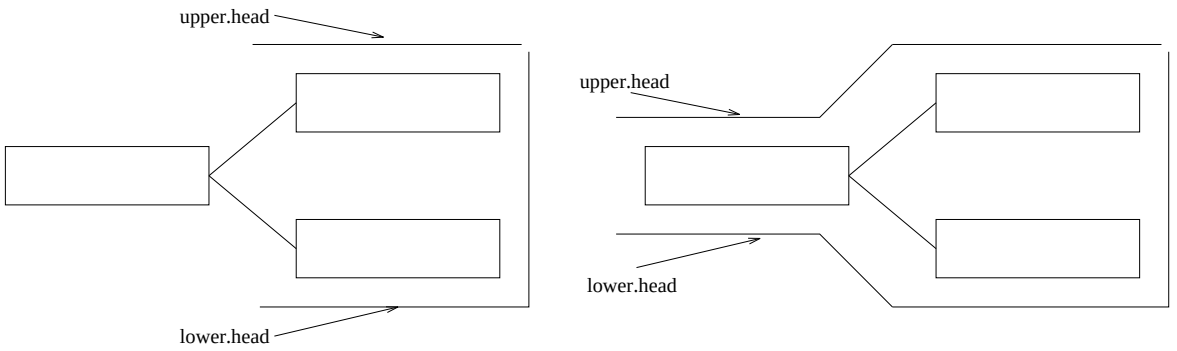
Κατασκευή πολυγώνου υποδέντρου . Εάν ο τρέχων κόμβος είναι ρίζα ενός υποδέντρου τότε η κατασκευή του πολυγώνου που το περικλείει γίνεται σε τρία στάδια :

1. Δημιουργούνται τα πολύγωνα των υποδέντρων με ρίζες τα παιδιά του τρέχοντος κόμβου (χρησιμοποιώντας τις `layout()` και `layout_leaf()`).

2. Τα πολύγωνα των παιδιών του τρέχοντος κόμβου τοποθετούνται όσο το δυνατόν πιο κοντά το ένα στο άλλο, υπολογίζονται οι σχετικές θέσεις των παιδιών και δημιουργείται η ένωση των πολυγώνων των παιδιών η οποία αποθηκεύεται στο πολύγωνο του τρέχοντος κόμβου

(χρησιμοποιώντας τη `join`).

3. Υπολογίζεται η απόσταση μεταξύ του πατέρα και των παιδιών και ολοκληρώνεται το πολύγωνο του πατέρα (χρησιμοποιώντας τη `attach_parent()`). Στο σχήμα Α.2 φαίνεται πως ολοκληρώνεται η κατασκευή του πολυγώνου του τρέχοντος κόμβου αφού έχει υπολογιστεί



Σχήμα Α.2: Επέκταση του πολυγώνου ενός πατέρα ώστε να συμπεριλάβει τα πολύγωνα των παιδιών του

η ένωση των πολυγώνων των παιδιών του.

Περισσότερες λεπτομέρειες σχετικά με τη κατασκευή και τη συνένωση των πολυγώνων που περικλείουν υποδέντρα καθώς και σχήματα που επεξηγούν την όλη διαδικασία της κατασκευής υπάρχουν στο [23].

Παράρτημα Β

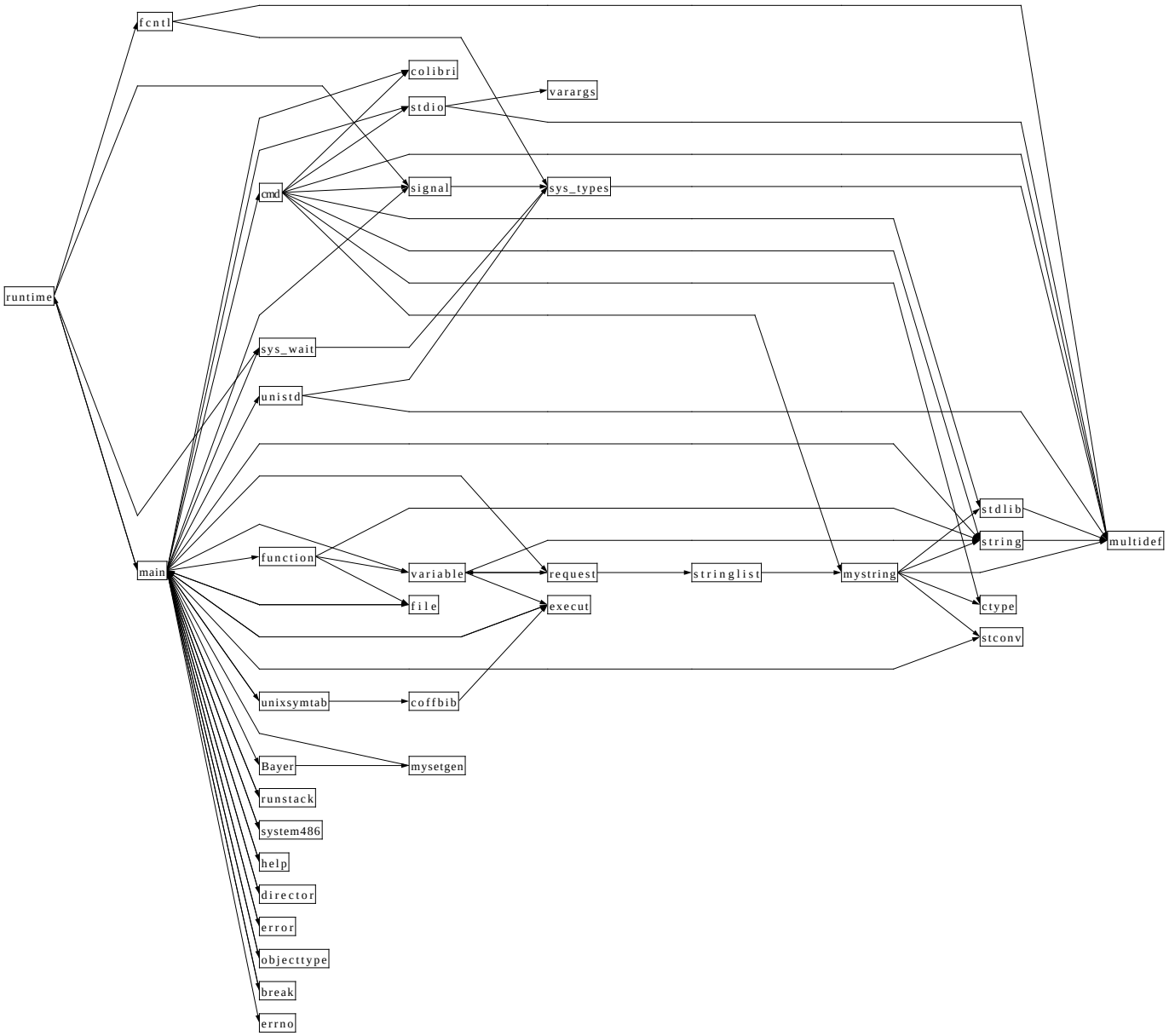
Οι γράφοι του συνόλου μετρήσεων

Στο παράρτημα αυτό δίνονται τα σχέδια που δημιουργούν οι δύο αλγόριθμοι για 11 από τους γράφους του συνόλου που χρησιμοποιήσαμε στις μετρήσεις του κεφαλαίου 4.

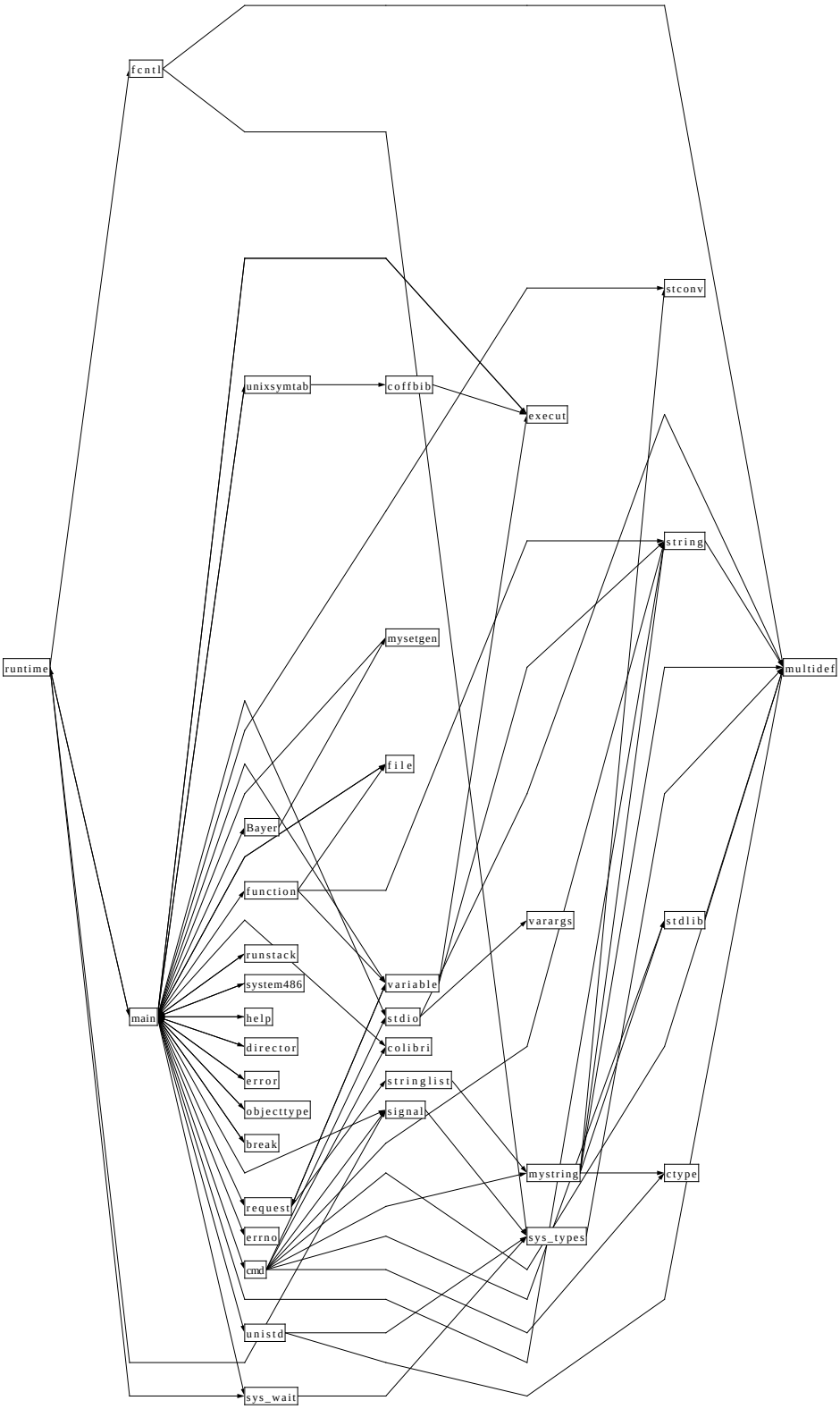
Μπορεί να παρατηρήσει κανείς τη τάση του αλγορίθμου ReArrange να σχεδιάζει όσο το δυνατόν μεγαλύτερα ευθύγραμμα τμήματα. Στα σχέδια που παράγει ο STT φαίνεται ο μεγάλος αριθμός ανεπιθύμητων κάμψεων ακμών. Αν και τελικά ο STT δημιουργεί σχέδια με λιγότερες τομές ακμών τα σχεδιά του τείνουν να είναι περισσότερο μπερδεμένα από ότι του ReArrange λόγω των πολλών κάμψεων ακμών.

Στη περίπτωση των γράφων με μικρή πυκνότητα, όπως ο γράφος SelfPortrait, ο ReArrange δημιουργεί καλύτερα σχέδια από τον STT. Αυτοί οι γράφοι μετατρέπονται δε δέντρα με αφαίρεση λίγων μόνον από τις ακμές τους. Ουσιαστικά έχουν μια δενδροειδή μορφή η οποία αναδεικνύεται από τον ReArrange ενώ δεν είναι φανερή στα σχέδια του STT.

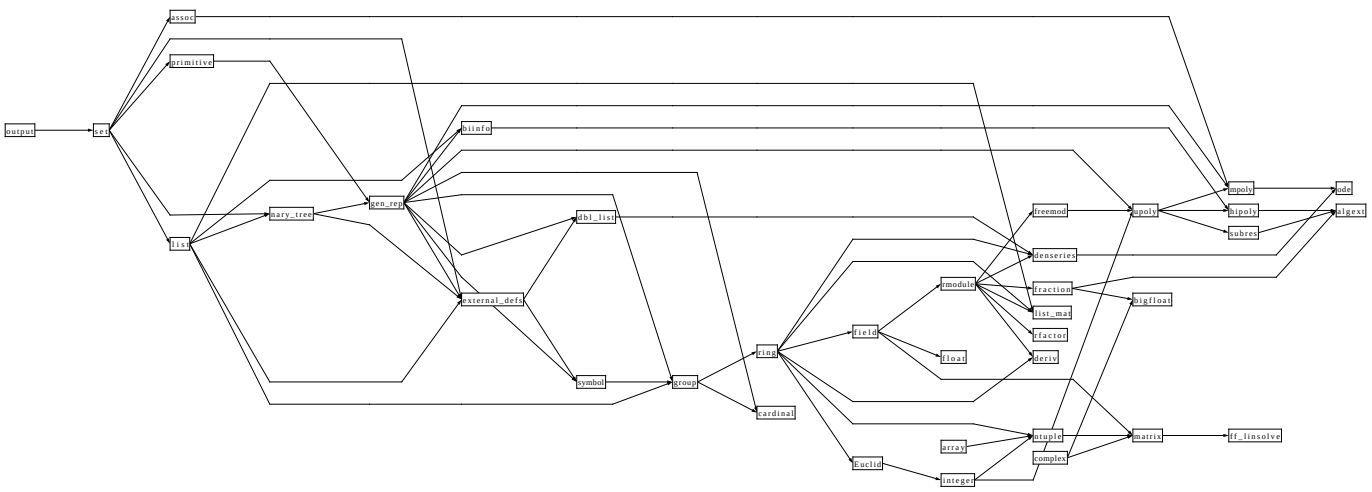
Οι γράφοι που δίνονται στη συνέχεια έχουν σμικρυνθεί κατάλληλα έτσι ώστε να χωρούν σε μια σελίδα. Κάποιοι από τους γράφους που καταλαμβάνουν μεγάλη απόσταση στον άξονα χ (έχουν μεγάλο μήκος) παρουσιάζονται εδώ περιστραμμένοι κατά 90^o.



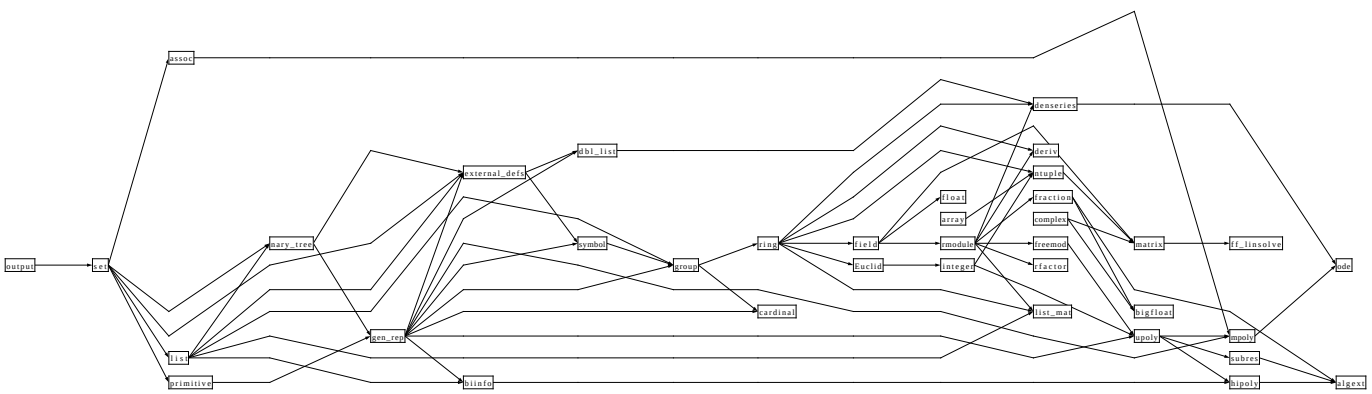
Σχήμα B.1: Σχέδιο του ReArrange για το γράφο mod_call



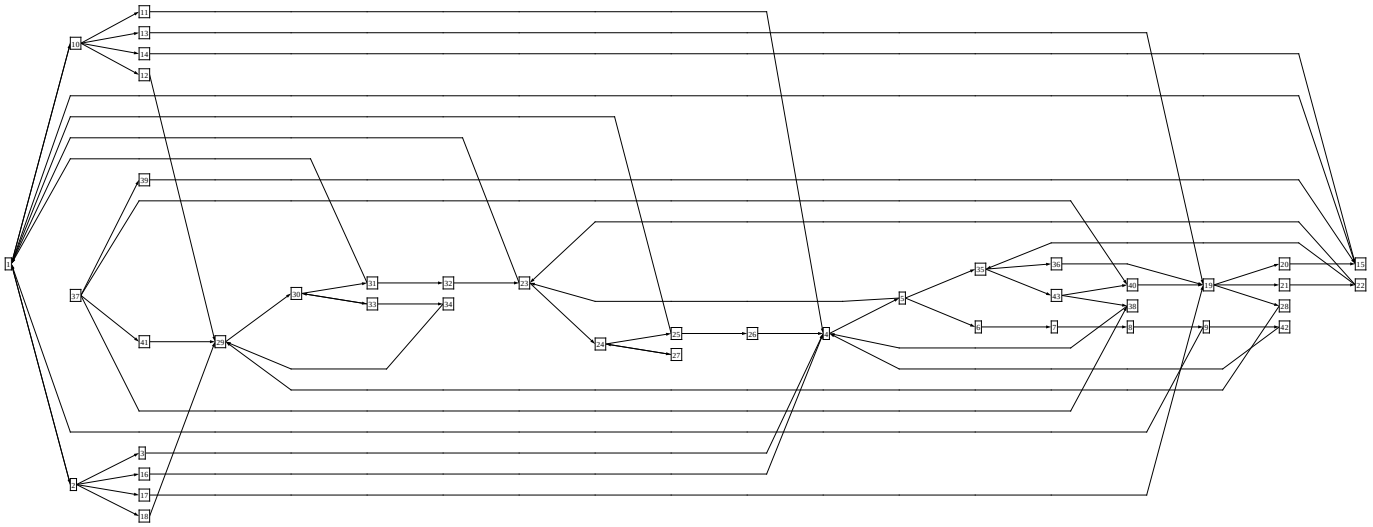
Σχήμα B.2: Σχέδιο του STT για το γράφο mod_call



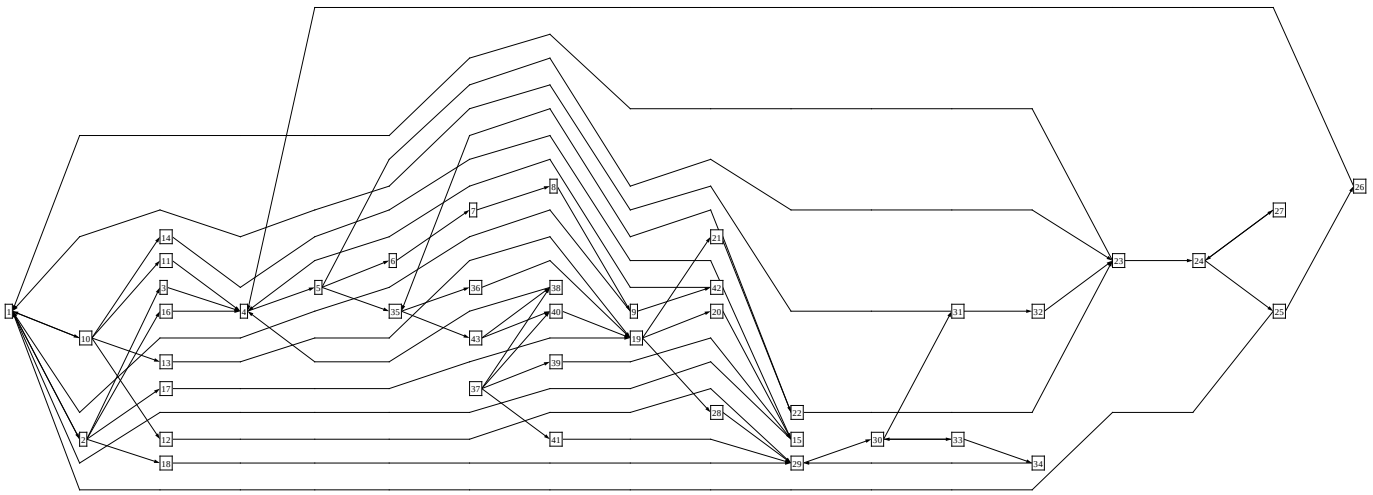
Σχήμα Β.3: Σχέδιο του ReArrange για το γράφο shortlist



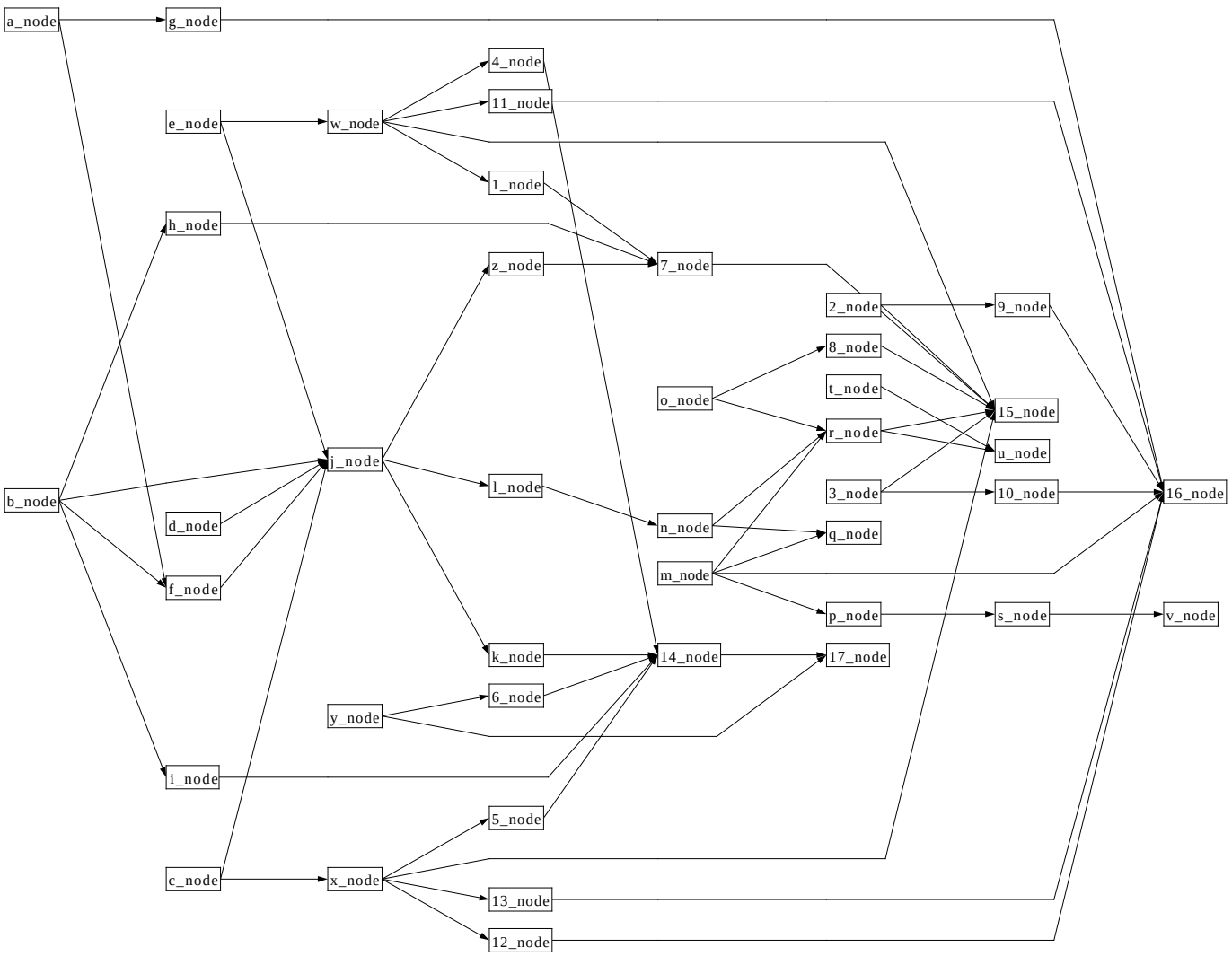
Σχήμα Β.4: Σχέδιο του STT για το γράφο shortlist



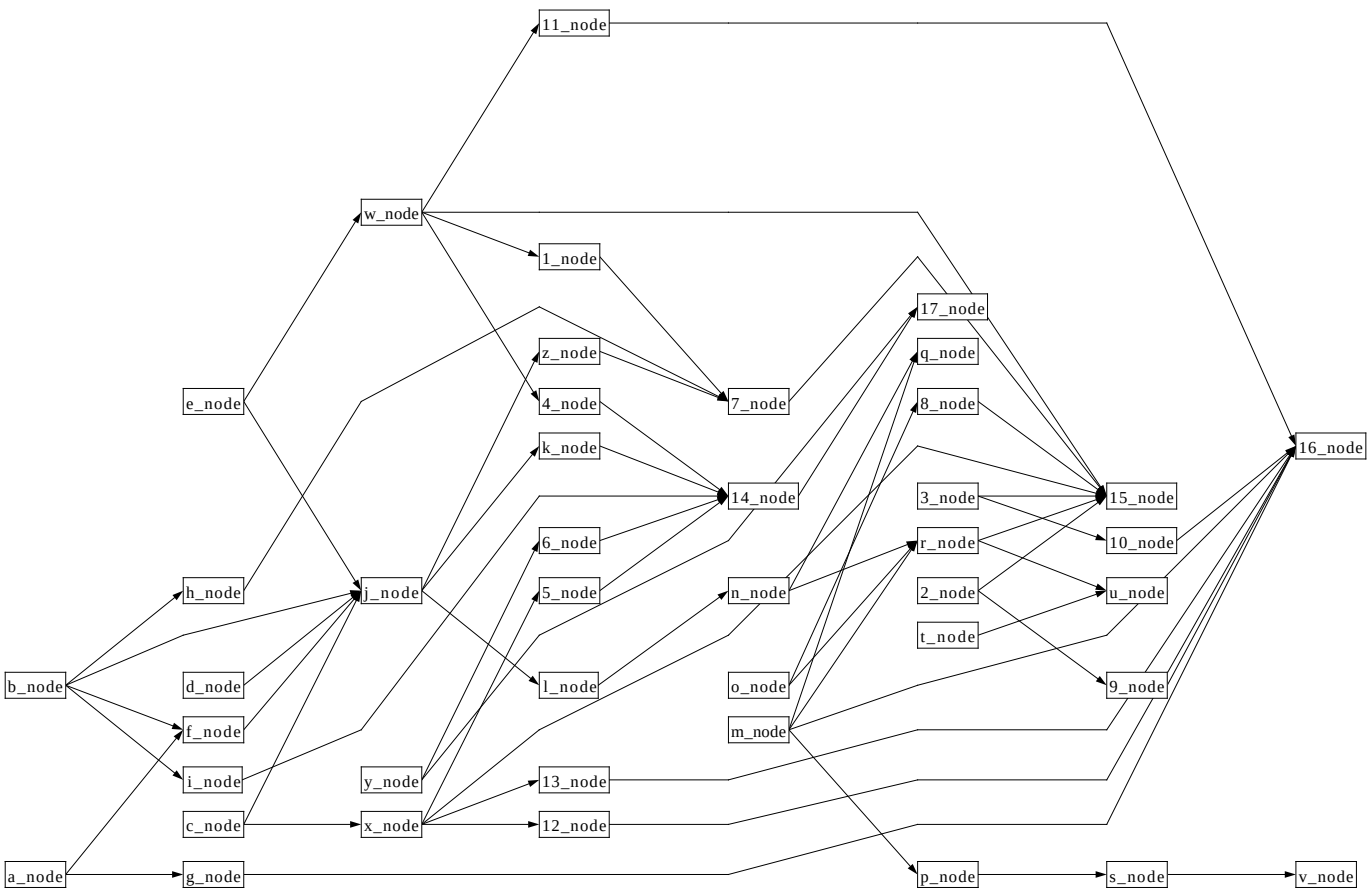
Σχήμα Β.5: Σχέδιο του ReArrange για το γράφο world2



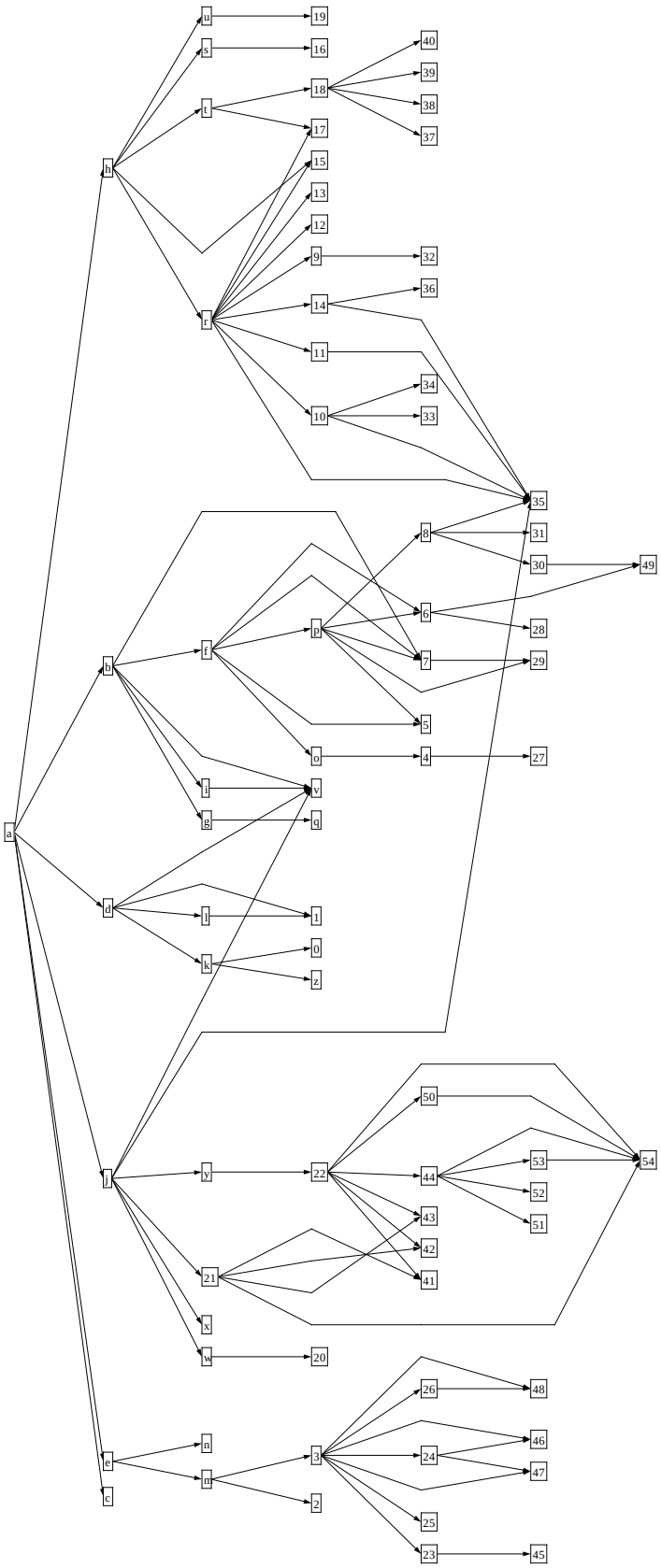
Σχήμα Β.6: Σχέδιο του STT για το γράφο world2



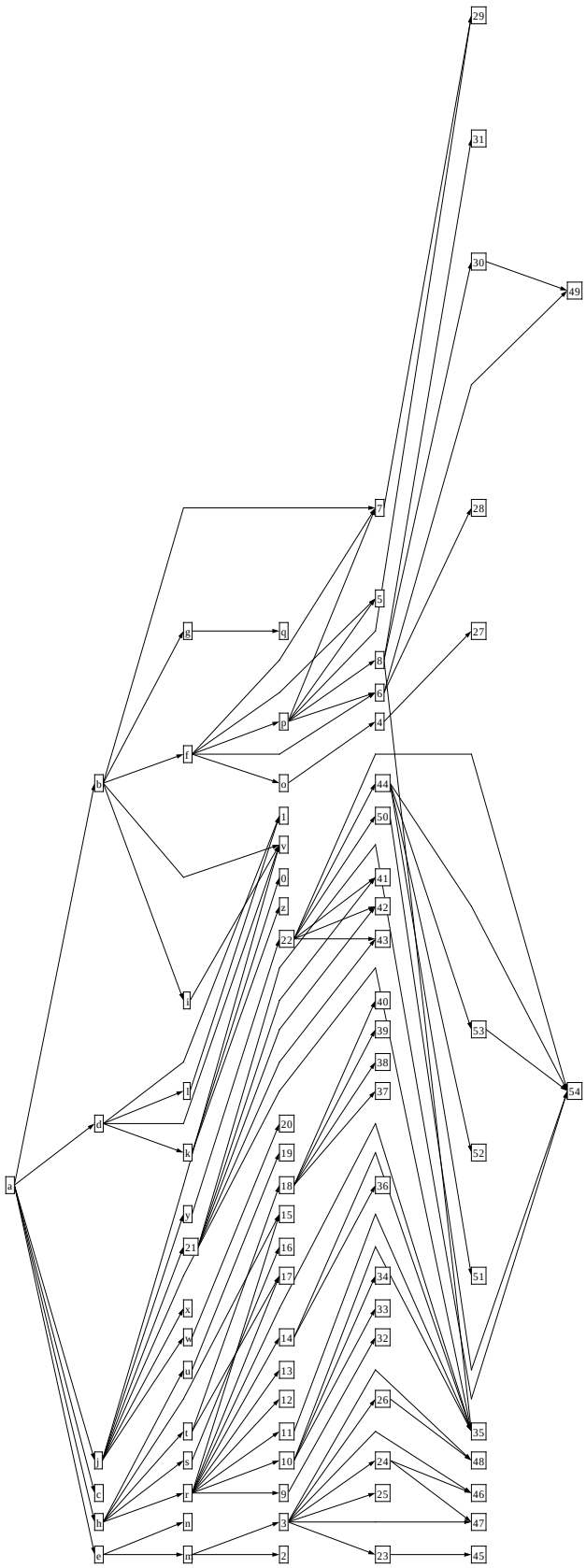
Σχήμα Β.7: Σχέδιο του ReArrange για το γράφο world1



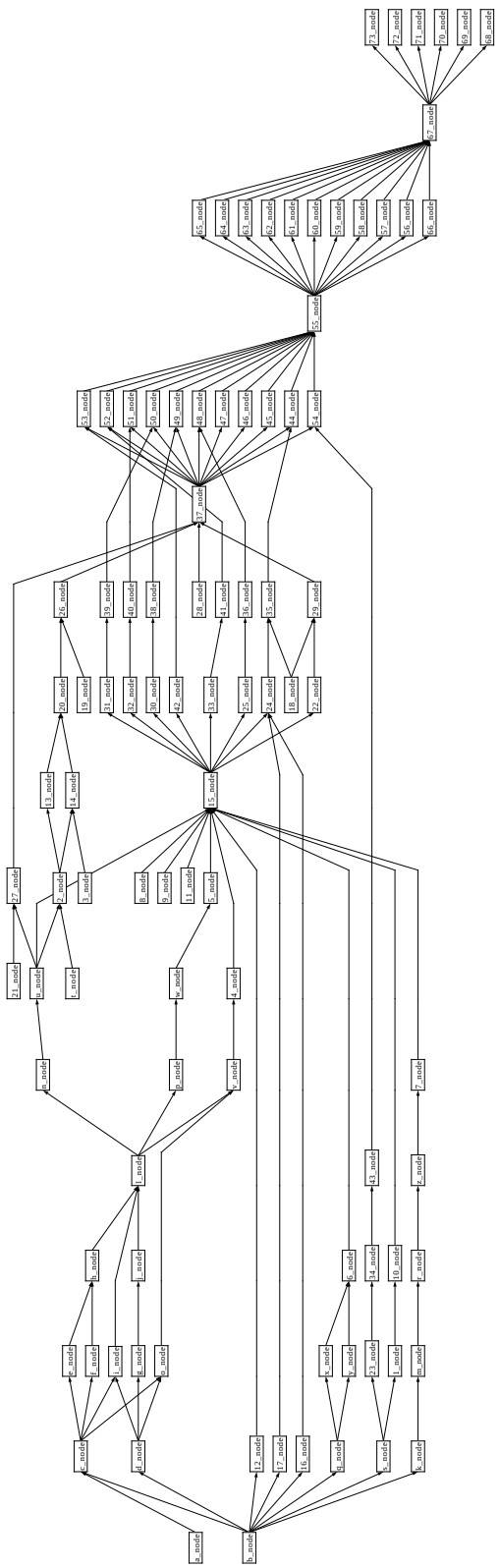
Σχήμα Β.8: Σχέδιο του STT για το γράφο world1



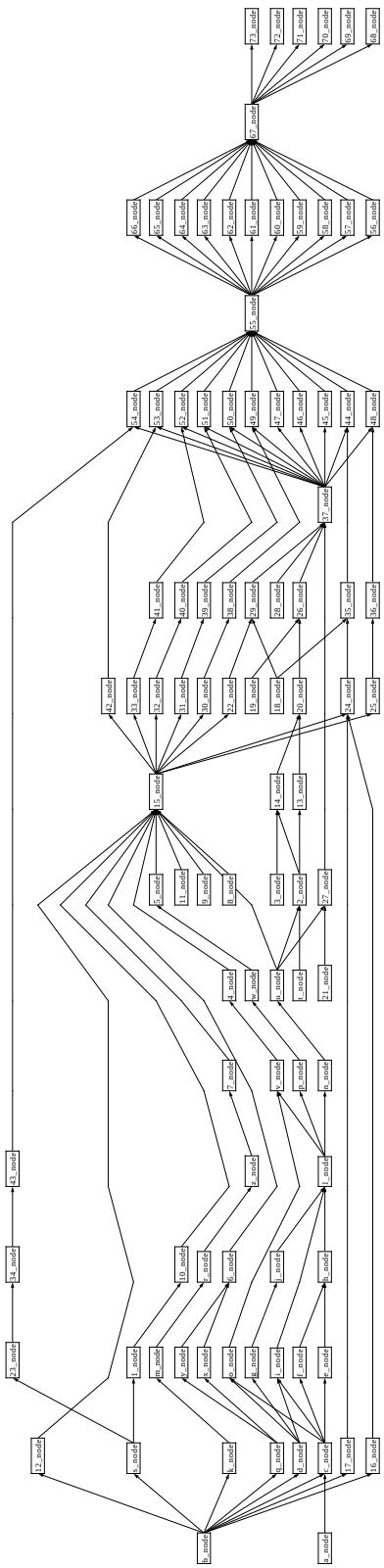
Σχήμα Β.9: Σχέδιο του ReArrange για το γράφο SelfPortrait



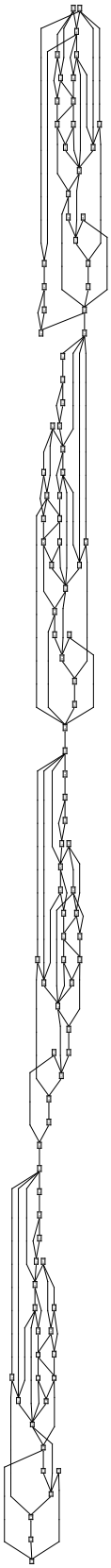
Σχήμα Β.10: Σχέδιο του STT για το γράφο SelfPortrait



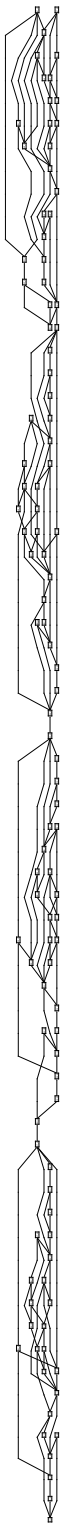
Σχήμα Β.11: Σχέδιο του ReArrange για το γράφο pert



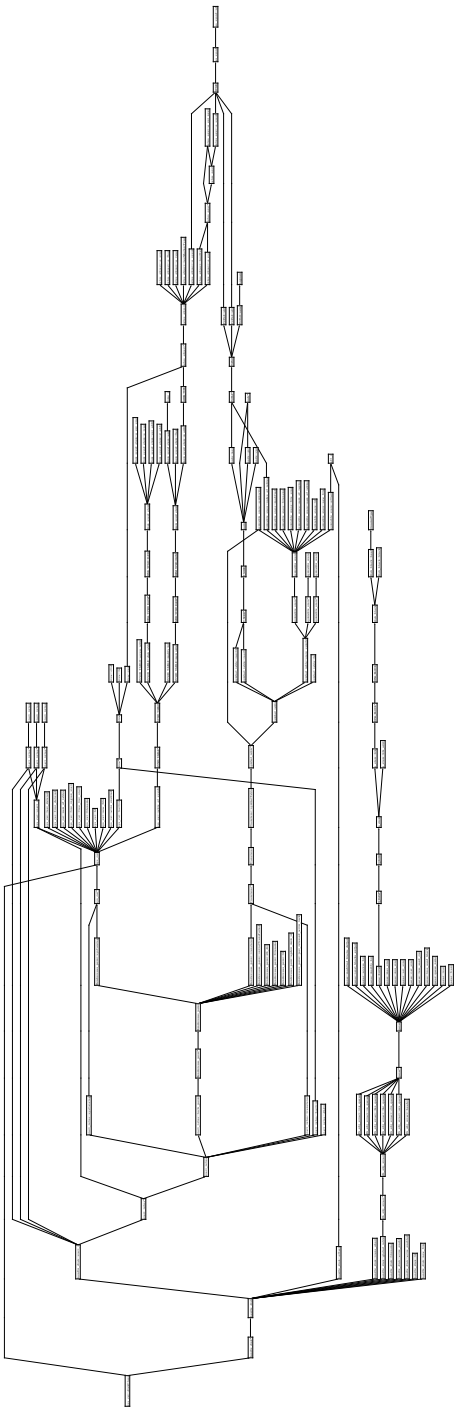
Σχήμα Β.12: Σχέδιο του STT για το γράφο pert



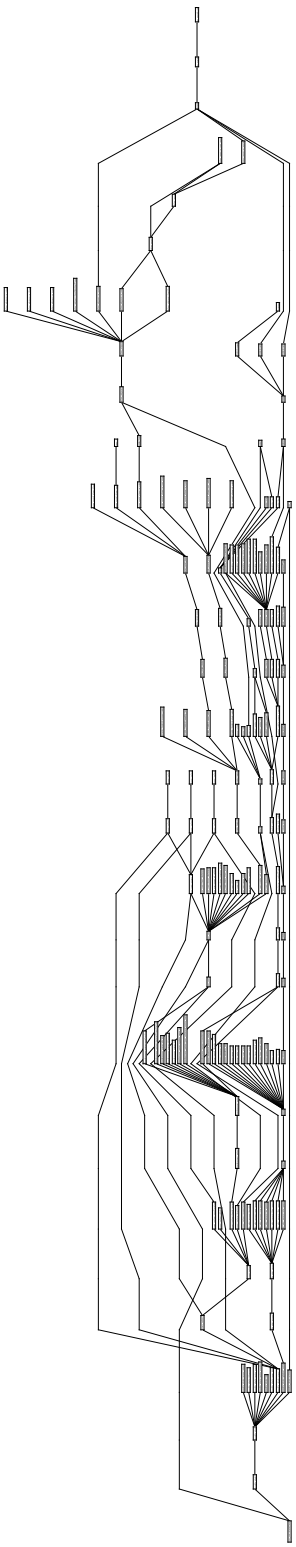
Σχήμα B.13: Σχέδιο του ReArrange για το γράφο p104



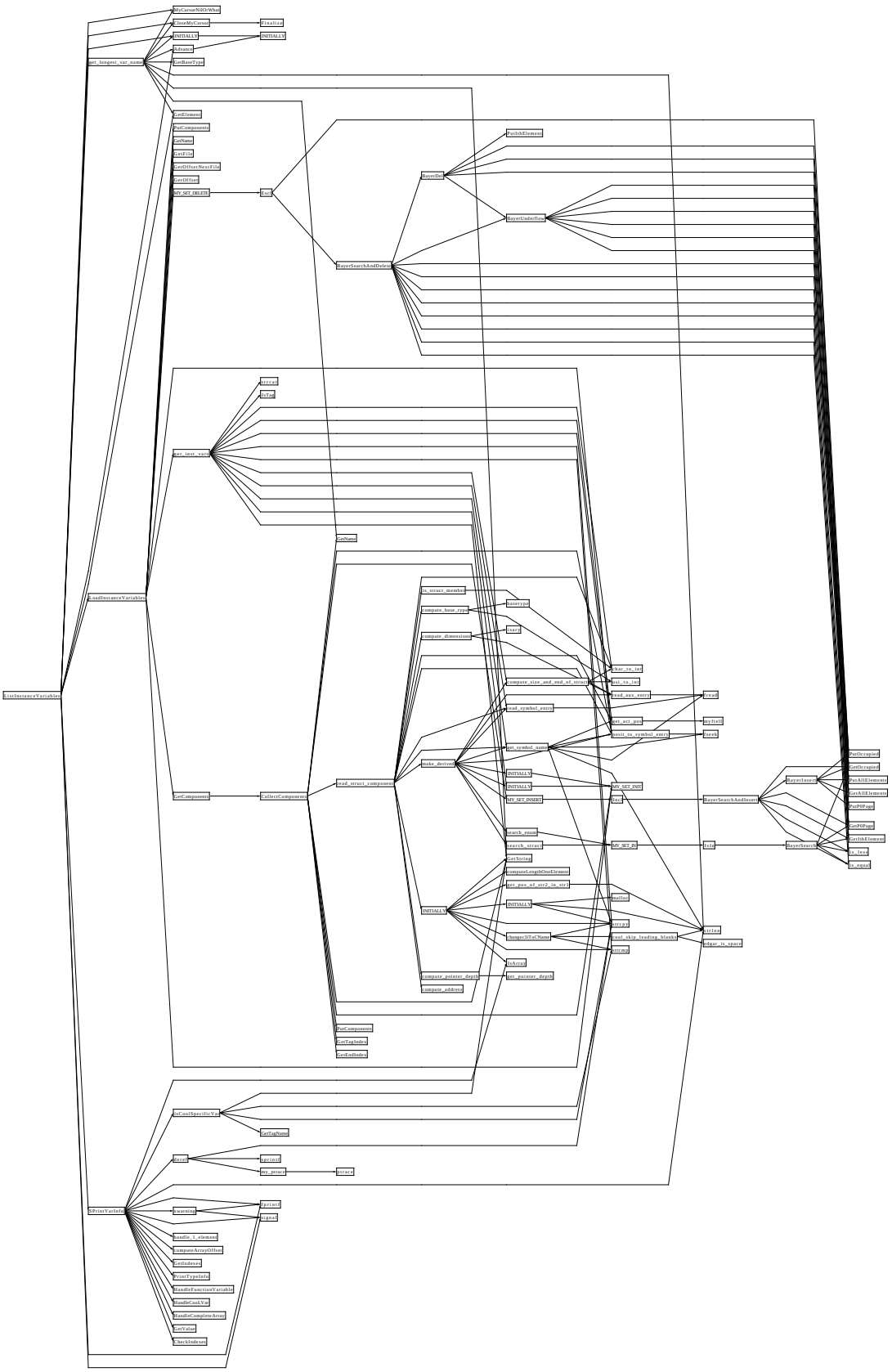
Σχήμα Β.14: Σχέδιο του STT για το γράφο p104



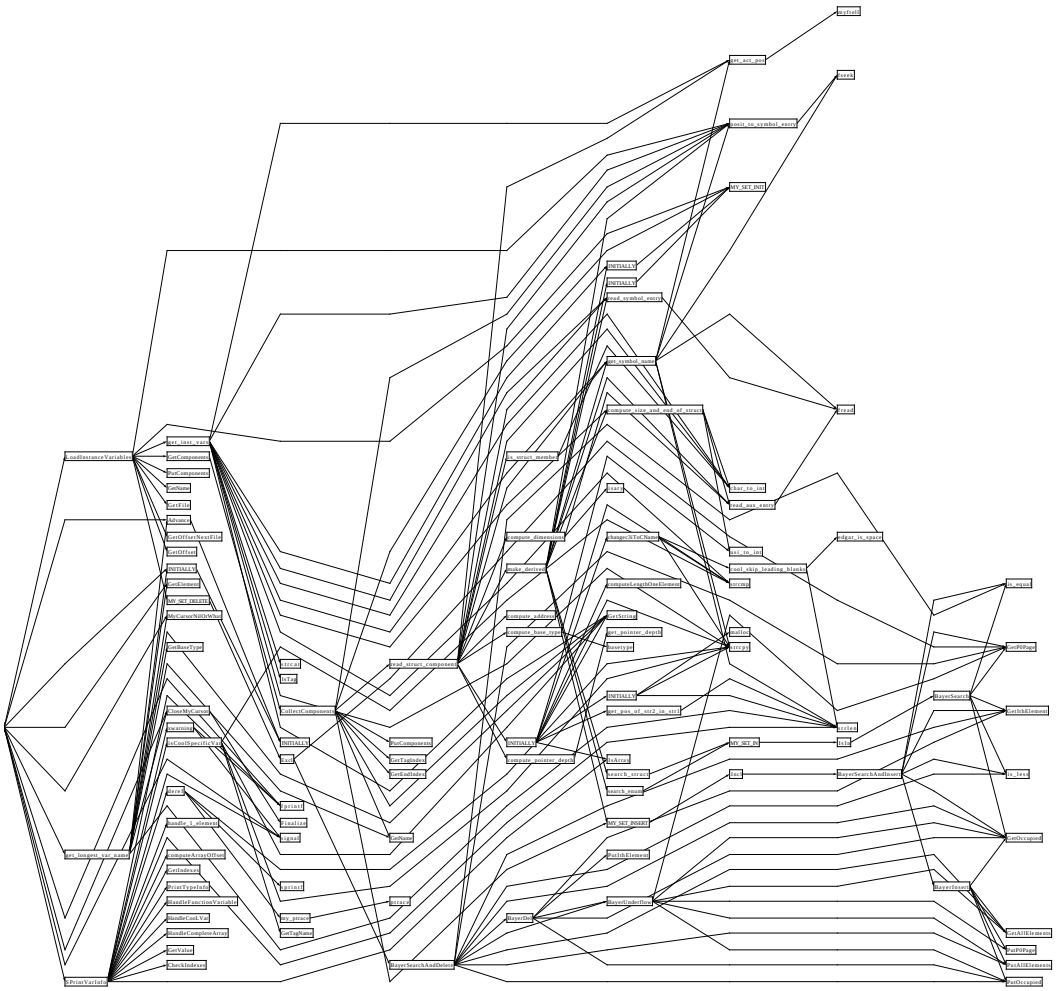
Σχήμα B.15: Σχέδιο του ReArrange για το γράφο t_hierarchy



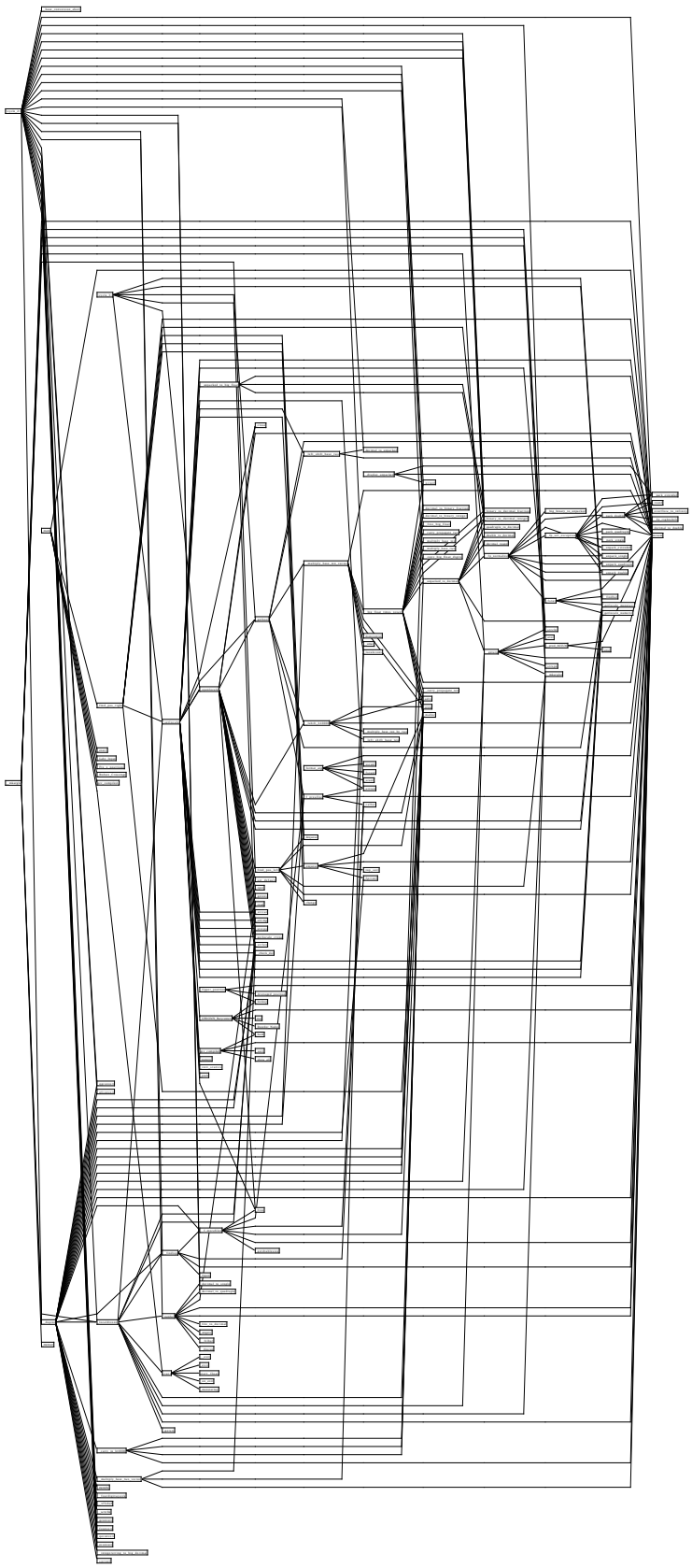
Σχήμα B.16: Σχέδιο του STT για το γράφο t_hierarchy



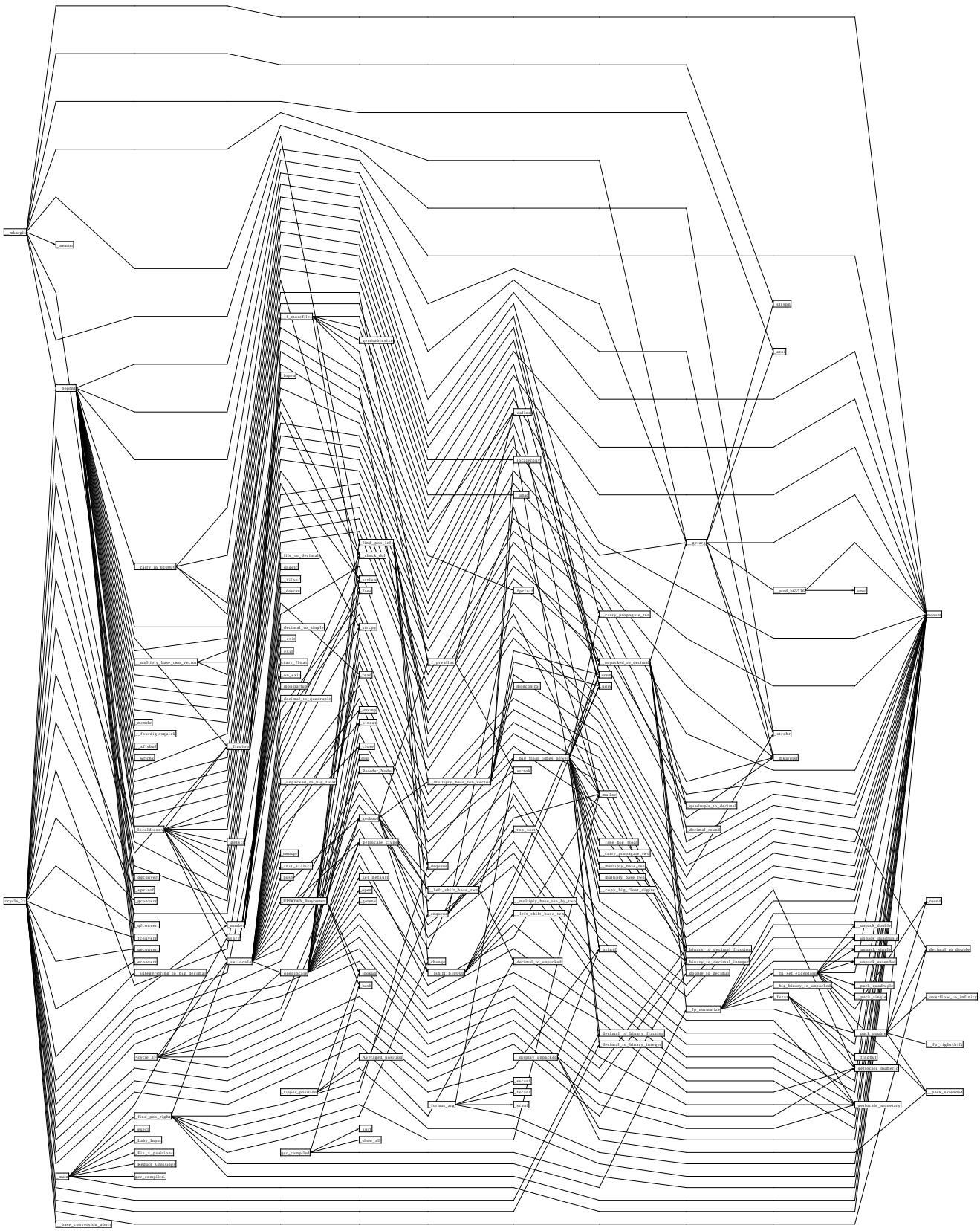
Σχήμα Β.17: Σχέδιο του ReArrange για το γράφο main1



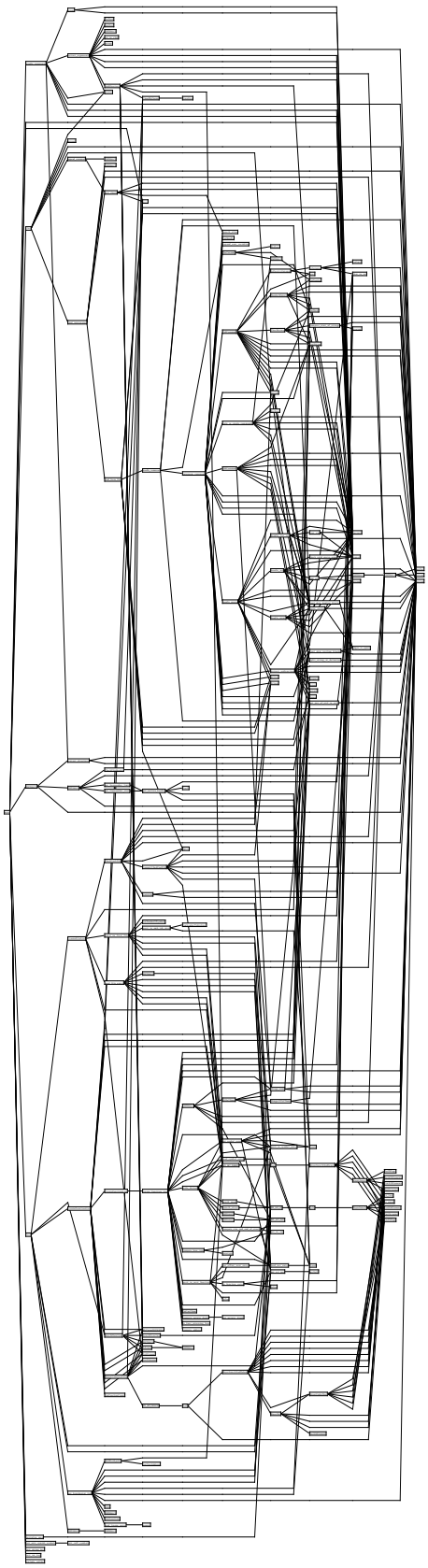
Σχήμα Β.18: Σχέδιο του STT για το γράφο main1



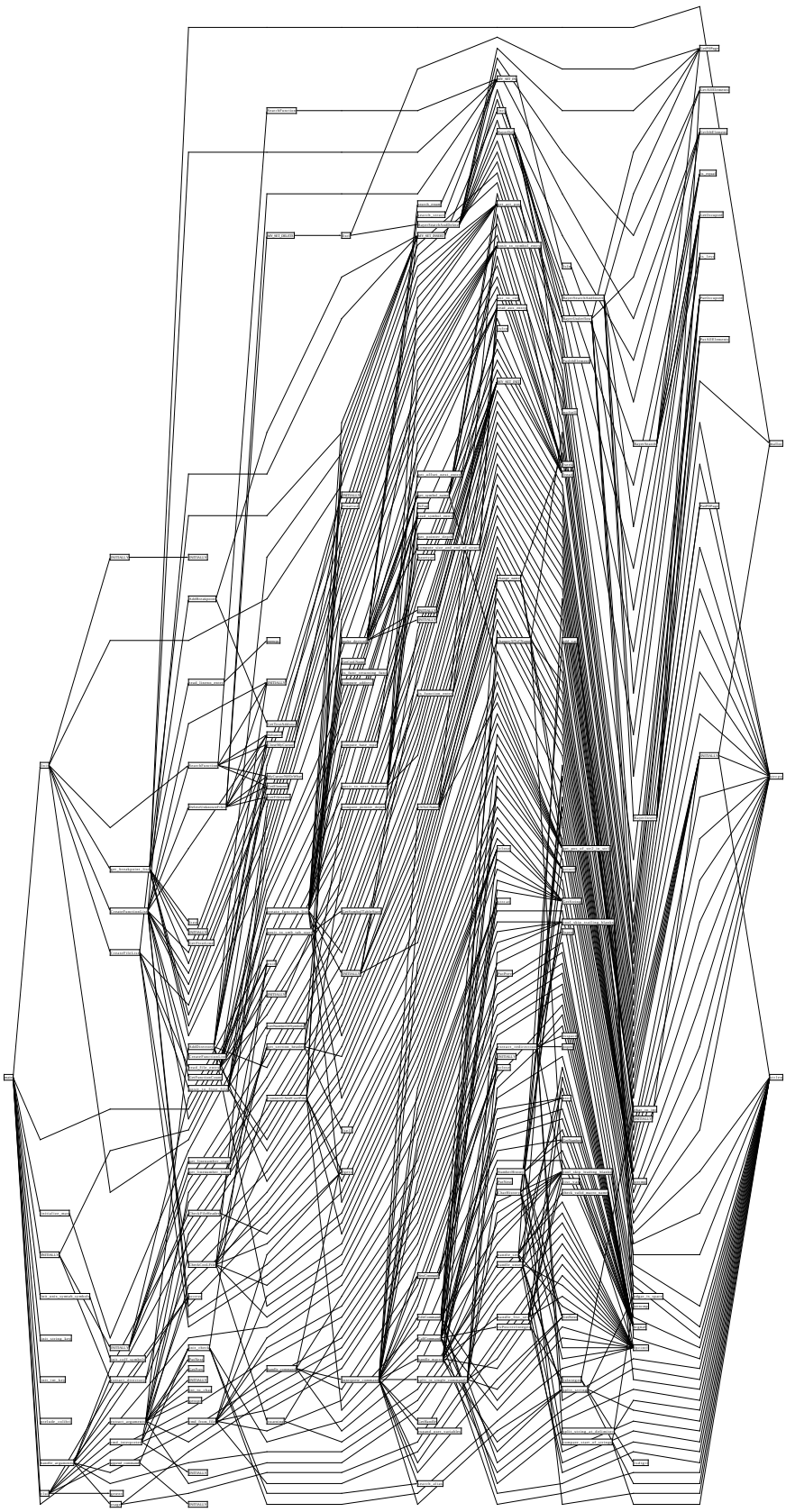
Σχήμα Β.19: Σχέδιο του ReArrange για το γράφο call_graph1



Σχήμα Β.20: Σχέδιο του STT για το γράφο call_graph1



Σχήμα B.21: Σχέδιο του ReArrange για το γράφο main2



Σχήμα Β.22: Σχέδιο του STT για το γράφο main2

Βιβλιογραφία

[1] *MacProject.* Apple Computer Inc., Cupertino, CA, 1984.

[2] G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis. ‘‘Algorithms for Drawing Graphs : an Annotated Bibliography’’. Technical Report UTDCS-7-93, The University of Texas Dallas, Richardson, Texas 75083-0688, Jun., 1993.

[3] G. di Battista and E. Nardelli. ‘‘An Algorithm for Testing Planarity of Hierarchical Graphs’’. *Graph-Theoretic Concepts in Computer Science*, (Proc. Int. Workshop WG ’86, Bernierd, June 1986), ed. G. Tinhofer and G. Schmidt., pages 277 -- 289, Lecture Notes in Computer Science, vol.246, Springer-Verlag, 1987.

[4] K. F. Böhringer and F. J. Newberry. ‘‘Using Constraints to Achieve Stability in Automatic Graph Layout Algorithms’’. In CHI’90 Proceedings, pages 43 -- 51, Apr. 1991.

[5] M. Carpano. ‘‘Automatic Display of Hierarchized Graphs’’. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-10(11):705 -- 715, 1980.

[6] P. Constantopoulos, M. Doerr, and Y. Vassiliou. ‘‘Repositories for Software Reuse: The Software Information Base’’. In Proc. IFIP WG 8.1 Conference on Information System Development Process, Como, pages 285 -- 307, September 1993.

[7] P. Constantopoulos and E. Pataki. ‘‘A Browser for Software Reuse’’. In Proc. Advanced Information Systems Engineering 4th International Conference, CAISE’92, Manchester, UK, pages 304 -- 319, May 1992.

[8] M. B. Davis. ‘‘A Layout Algorithm for a Graph Browser’’. Master’s Project Report, Computer Science Division, EECS, UCB, Berkeley, May 1985.

[9] M. Delarche. ‘‘Quelques Outils Infographiques Pour l’Analyse Structurale de Systemes’’. Dr. Ing. Thesis, Jun. 1979. University Grenoble.

[10] M. Dörr. Personal communication, August 1992.

[11] P. Eades and D. Kelly. ‘‘Heuristics for Reducing Crossings in 2-layered Networks’’. *Ars Combinatoria*, 21.A:89 -- 98, 1986.

[12] P. Eades, B. McKay, and N. Wormald. ‘‘An NP-hard Crossing Number Problem for Bipartite Graphs’’. Technical Report 60, Dept. of Computer Science, Univ. of Quinsland, 1985.

[13] P. Eades and N. Wormald. ‘‘The Median Heuristic for Drawing 2-layered Networks’’. Technical Report 69, Dept. of Computer Science, Univ. of Quinsland, 1986.

[14] J. K. Foderato. ‘‘The Design of a Language for Algebraic Computation Systems’’. Ph.D. Dissertation, Computer Science Division, EECS, UCB, Berkeley, CA, 1983.

[15] J. W. Forrester. *World Dynamics.* Wright-Allen Press, Cambridge, MA, 1971.

[16] E.R. Gansner, S.C. North, and K.P. Vo. ‘‘DAG -- A Program that Draws Directed Graphs’’. *Software - Practice & Experience*, 18(11):1047 -- 1062, Nov. 1988.

[17] F. Harary. *Graph Theory*. Addison-Wesley, Reading, PA, 1969.

[18] M. Katevenis, T. Sorilos, and P. Kalogerakis. Laby programmer’s manual. Technical Report ITHACA.FORTH.92.1.3.3.#1, CSI-FORTH, Jan 1992.

[19] D. E. Knuth. ‘‘Computer Drawn Flowcharts’’. *Communications of the ACM*, 6:555 -- 563, Sep. 1963.

[20] D. E. Knuth. ‘‘Optimum Binary Search Trees’’. *Acta Informatica*, 1:14 -- 25, 1971.

[21] E.B. Messinger. ‘‘Automatic Layout of Large Directed Graphs’’. Technical Report No.88-07-08, Department of Computer Science, University of Washington, Seattle, Washington 98195, Jul. 1988.

[22] C. Meyer. ‘‘A Browser for Directed Graphs’’. Master’s Project Report, Computer Science Division, EECS, UCB, Berkeley, CA, Dec. 1983.

[23] S. Moen. ‘‘Drawing Dynamic Trees’’. *IEEE Software*, pages 21 -- 28, Jul. 1990.

[24] S. Näher. *LEDA User Manual.* Max-Planck-Institut für Informatik, Im Stadtwald, D-6600 Saarbrücken, 1992.

[25] E. M. Reingold and J. Tilford. ‘‘Tidier Drawing of Trees’’. *IEEE Trans. on Software Engineering*, SE-7(2):223 -- 228, 1981.

[26] G. Robins. ‘‘The ISI Grapher: a Portable Tool for Displaying Graphs Pictorially’’. In *Symboliika ’87*, Helsinki, Finland, Aug. 17-18 1987.

[27] L.A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spyrakis, and A. Tuan. ‘‘A Browser for Directed Graphs’’. *Software - Practice & Experience*, 17(1):61 -- 76, Jan. 1987.

[28] Bjarne Stroustroup. *The C++ Programming Language.* Addison-Wesley, Reading, Massachusets, 1991.

[29] T. Sugiyama, S. Tagawa, and M.Toda. ‘‘Methods for Visual Understanding of Hierarchical System Structures’’. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC(11):109 -- 125, Feb. 1981.

[30] K. J. Supowit and E. M. Reingold. ‘‘The Complexity of Drawing Trees Nicely’’. *Acta Informatica*, 18:377 -- 392, 1983.

[31] R. Tamassia, G. di Battista, and C. Batini. ‘‘Automatic Graph Drawing and Readability of Diagrams’’. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61 -- 79, January/February 1988.

[32] W. T. Tutte. ‘‘How to Draw a Graph’’. In Proceedings of the London Mathematical Society 3rd Series, 13, *52*, pages 743 -- 748, 1963.

[33] J. Vaucher. ‘‘Pretty Printing of Trees’’. *Software - Practice & Experience*, 10(7):553 -- 561, 1980.

[34] J. Q. Walker II. ‘‘A Node Positioning Algorithm for General Trees’’. *Software - Practice & Experience*, 20(7):685 -- 705, 1990.

[35] J. N. Warfield. *‘‘Societal Systems’’*. John Wiley & Sons, New York, 1976.

[36] J. N. Warfield. ‘‘Crossing Theory and Hierarchy Mapping’’. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(7):505 -- 523, 1977.

[37] C. Whetherell and A. Shannon. ‘‘Tidy Drawing of Trees’’. *IEEE Trans. on Software Engineering*, SE-5(5):514 -- 520, 1979.