ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

Εξόρυξη δεδομένων από γράφους κοινωνικών δικτύων

Σωτήριος Ραμπούσης

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Σεπτέμβριος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Σχολή Χρηματοοικονομικής και Στατιστικής

Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΣΤΗΝ **ΕΦΑΡΜΟΣΜΕΝΗ ΣΤΑΤΙΣΤΙΚΗ**

Εξόρυξη δεδομένων από γράφους κοινωνικών δικτύων

Σωτήριος Ραμπούσης

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στην *Εφαρμοσμένη Στατιστική*

Πειραιάς
Σεπτέμβριος 2024

Η παρούσα Διπλωματική Εργασία εγκρίθηκε ομόφωνα από την Τριμελή Εξεταστική Επιτροπή που ορίσθηκε από τη ΓΣΕΣ του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς στην υπ' αριθμ. …….. συνεδρίασή του σύμφωνα με τον Εσωτερικό Κανονισμό Λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Στατιστική

Τα μέλη της Επιτροπής ήταν:
- Καθηγητής Μάρκος Κούτρας (Επιβλέπων)
- Καθηγητής Ιωάννης Θεοδωρίδης
- Αναπληρωτής Καθηγητής Πελέκης Νικόλαος

Η έγκριση της Διπλωματική Εργασίας από το Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης του Πανεπιστημίου Πειραιώς δεν υποδηλώνει αποδοχή των γνωμών του συγγραφέα.

UNIVERSITY OF PIRAEUS
School of Finance and Statistics

Department of Statistics and Insurance Science

POSTGRADUATE PROGRAM IN
APPLIED STATISTICS

Data Mining from Social-Network Graphs

By
Sotirios Rampousis

MSc Dissertation

submitted to the Department of Statistics and Insurance Science of the University of Piraeus in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

Piraeus, Greece
September 2024

Στην οικογένεια μου

Ευχαριστίες

Η συγκεκριμένη διπλωματική εργασία εκπονήθηκε στα πλαίσια του μεταπτυχιακού προγράμματος σπουδών «Εφαρμοσμένης Στατιστικής» του Τμήματος Στατιστικής και Ασφαλιστικής Επιστήμης, του Πανεπιστημίου Πειραιώς. Αρχικά Θα ήθελα να εκφράσω τις θερμές ευχαριστίες μου στον επιβλέποντα καθηγητή της παρούσας διπλωματικής εργασίας κ. Κούτρα Μάρκο, Καθηγητή του Τμήματος Στατιστικής και Ασφαλιστικής επιστήμης του Πανεπιστημίου Πειραιώς, όπου με επιστημονική και συμβουλευτική καθοδήγηση, με βοήθησε στην κατανόηση και ανάπτυξη του θέματος για την εκπόνηση της διπλωματικής μου εργασίας, όπως επίσης για την εμπιστοσύνη που μου έδειξε ώστε να ασχοληθώ με το συγκεκριμένο θέμα. Επιπροσθέτως θέλω να ευχαριστήσω την οικογένεια μου αλλά και τους φίλους μου για τη στήριξη που μου προσέφεραν καθ' όλη τη διάρκεια των σπουδών μου.

Για την διπλωματική εργασία χρησιμοποιήθηκε (και) εξοπλισμός του εργαστηρίου Στατιστικής που αποκτήθηκε με χρηματοδότηση της πράξης «Προμήθεια και Εγκατάσταση Εξειδικευμένου Ερευνητικού Εξοπλισμού στο Πανεπιστήμιο Πειραιώς» (MIS 5066760) του Περιφερειακού Επιχειρησιακού Προγράμματος «Αττική 2014 – 2020»

Περίληψη

Τα τελευταία χρόνια η ραγδαία ανάπτυξη των κοινωνικών δικτύων έχει οδηγήσει τους χρήστες, σε καθημερινή και πολύωρη χρήση τους. Το φαινόμενο αυτό, έχει στρέψει το ενδιαφέρον πολλών ερευνητών προς τη μελέτη της δομής διαφόρων κοινωνικών δικτύων, αφού τα αποτελέσματα μπορούν να χρησιμοποιηθούν σε τομείς όπως οι πωλήσεις ειδών και υπηρεσιών μέσα από κοινωνικά δίκτυα, στη τουριστική βιομηχανία, στη μελέτη διασποράς ειδήσεων, στη κατανόηση των σχέσεων μεταξύ κρατών κλπ. Έτσι έχουν δημιουργηθεί διάφοροι αλγόριθμοι, οι οποίοι μπορούν να ερμηνεύσουν την κοινωνική σύσταση ενός δικτύου και μπορούν να μας βοηθήσουν στην κατανόηση της δομής ποικίλων δικτύων, όπως επίσης και τις ανάγκες των ατόμων από τα οποία αποτελείται. Τα σύνολα κοινωνικών δικτύων είναι γιγαντιαία με αποτέλεσμα, τα συστήματα να χρειάζονται πολύ χρόνο για να τα επεξεργαστούν. Στα πλαίσια της διπλωματικής εργασίας θα παρουσιάσουμε, τη σημαντικότητα εξόρυξης δεδομένων από κοινωνικά δίκτυα και θα δούμε πως μπορούμε να τα αναπαραστήσουμε σε φυσική μορφή. Έπειτα, θα μελετήσουμε διάφορα χρήσιμα μέτρα και αλγόριθμους, με τα οποία μπορούμε να μελετήσουμε τη δομή των δικτύων και να κατανοήσουμε το πως μεταφέρεται η πληροφορία μέσα σε αυτά. Τέλος θα εφαρμόσουμε τους αλγορίθμους σε συνθετικά αλλά και πραγματικά σύνολα δεδομένων όπου θα αξιολογήσουμε την ποιότητα των αποτελεσμάτων τους, αλλά και το πόσο εύχρηστοι είναι οι αλγόριθμοι σε μεγάλα σύνολα δεδομένων.

Abstract

In recent years, the rapid growth of social networks has led users to use social networks on a daily and extended time basis. The phenomenon has drawn the interest of many researchers, to study the structure of various social networks, since the results can be used in various fields, such as the sales of products or services through social platforms, the tourism industry, news dispersion, understanding relations between countries etc. Thus, various algorithms have been created, which can interpret the social composition of a network and help us understand the structure of various networks, as well as the needs of their members. The volume of social networks is huge and, as a result, systems take a long time to analyze them. In this thesis we will present, the importance of data mining from networks and review how we can represent them in a physical form. We will then study several useful measures and algorithms, with which we can study the structure of networks and understand how information is transferred within them. Finally, we will apply the algorithms to synthetic and real datasets where we will evaluate the quality of the results and how flexible the algorithms are when applied on large data sets.

Περιεχόμενα

Κατάλογος Πινάκων ..... *xviii*

Κατάλογος Σχημάτων ..... *xx*

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή *1*

ΚΕΦΑΛΑΙΟ 2 Γράφοι και κοινωνικά δίκτυα *3*

2.1 Κοινωνικά δίκτυα **3**

2.2 Διάφορες κατηγορίες κοινωνικών δικτύων **4**

2.3 Γράφοι **6**

2.3.1 Βασικά χαρακτηριστικά ενός δικτύου 6

2.3.2 Πίνακες αναπαράστασης 8

2.4 Κατηγορίες γράφων **8**

2.5 Τα κοινωνικά δίκτυα ως γράφοι **13**

ΚΕΦΑΛΑΙΟ 3 Χρήσιμα μέτρα *15*

3.1 Εισαγωγή **15**

3.2 Ασυμπτωτική ανάλυση χρόνου **15**

3.3 Μέτρα απόστασης **18**

3.4 Μέτρα ομοιότητας **19**

3.5 Μέτρα κεντρικότητας **23**

ΚΕΦΑΛΑΙΟ 4 Απαρίθμηση τριγώνων *35*

4.1 Εισαγωγή **35**

4.2 Αλγόριθμοι καταμέτρησης και καταχώρησης τριγώνων **35**

4.3 Χρησιμότητα εύρεσης τριγώνων σε δίκτυα **43**

ΚΕΦΑΛΑΙΟ 5 Κοινότητες *47*

5.1 Εισαγωγή **47**

5.2 Ιεραρχικές μέθοδοι 47

5.3 Μέθοδοι μεγιστοποίησης ποιοτικών συναρτήσεων 52

5.3.1 Modularity 52

5.3.2 Βελτιστοποίηση υπολογισμού modularity 53

5.3.3 Αλγόριθμος Louvain 55

5.3.4 Αλγόριθμος Leiden 59

5.4 Τυχαίοι περίπατοι 65

5.4.1 Αλγόριθμος Walktrap 66

5.5 Αλληλοκαλυπτόμενες κοινότητες 71

ΚΕΦΑΛΑΙΟ 6 Εφαρμογή αλγορίθμων 75

6.1 Εισαγωγή 75

6.2 Συνθετικά δίκτυα γράφων 75

6.3 Αποτελέσματα της μελέτης συνθετικών δικτύων 82

6.3.1 Αξιολόγηση των ομαδοποιήσεων 83

6.3.2 Αποτελέσματα εύρεσης κοινοτήτων σε συνθετικά δίκτυα 84

6.3.3 Αποτελέσματα σχετικά με την απαρίθμηση τριγώνων σε συνθετικούς γράφους 89

6.4 Αποτελέσματα πραγματικών δικτύων 90

6.4.1 Αξιολόγηση ομαδοποιήσεων σε σύνολα πραγματικών δεδομένων 92

6.4.2 Αξιολόγηση αλγορίθμων καταμέτρησης τριγώνων σε σύνολα πραγματικών δεδομένων 97

Βιβλιογραφία..... 103

Παράρτημα..... 107

Κατάλογος Πινάκων

		Σελίδα
Πίνακας 3.1.	Αποτελέσματα κεντρικότητας Katz	30
Πίνακας 6.1.	Αποτελέσματα κοινοτήτων ενός Gaussian μοντέλου	79
Πίνακας 6.2.	Παράμετροι για τη δημιουργία συνθετικών δικτύων για τις διάφορες τιμές παραμέτρου μίξης	84
Πίνακας 6.3.	Χαρακτηριστικά πραγματικών συνόλων δεδομένων	92
Πίνακας 6.4.	Ποσοστά αποσυνδεμένων κοινοτήτων αλγορίθμου Louvain για δέκα επαναλήψεις	95
Πίνακας 6.5.	Στατιστικά χρόνων εκτέλεσης των αλγορίθμων καταμέτρησης τριγώνων για τα πραγματικά σύνολα δεδομένων	101

Κατάλογος Σχημάτων

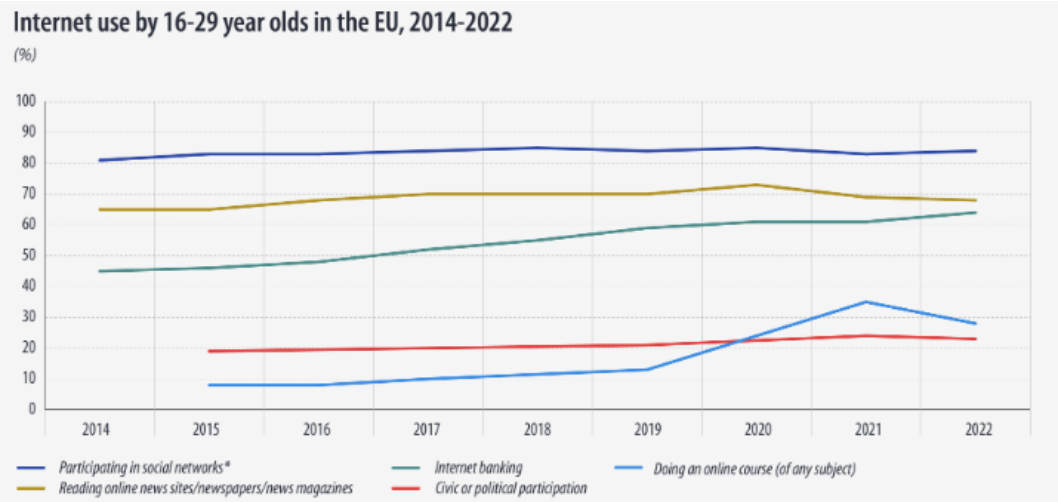
		Σελίδα
Σχήμα 1.1.	Χρήση διαφόρων λειτουργειών του διαδικτύου για ηλικίες 16-29 ετών	1
Σχήμα 2.1.	Κατευθυνόμενος γράφος	9
Σχήμα 2.2.	Μη-Κατευθυνόμενος γράφος	10
Σχήμα 2.3.	Γράφος με βάρη	10
Σχήμα 2.4.	Διμερής γράφος	11
Σχήμα 2.5.	Προσημικός γράφος	12
Σχήμα 3.1.	Γραφική αναπαράσταση σύμβολου Ο	16
Σχήμα 3.2.	Γραφική αναπαράσταση σύμβολου Ω	16
Σχήμα 3.3.	Γραφική αναπαράσταση σύμβολου Θ	17
Σχήμα 3.4.	Ταξινόμηση προβλημάτων σε κλάσεις	18
Σχήμα 3.5.	Κόμβοι συνδεδεμένοι με όλο το δίκτυο	22
Σχήμα 3.6.	Δίκτυο με μια ομάδα ατόμων απομακρυσμένη από το υπόλοιπο δίκτυο	25
Σχήμα 3.7.	Μη αποσυνδεμένο δίκτυο με κόμβους με μηδενική κεντρικότητα	28
Σχήμα 3.8.	Σχηματική αναπαράσταση ενός υποσυνόλου 10 ιστοσελίδων.	32
Σχήμα 3.9.	Αναπαράσταση ενός loop	33
Σχήμα 4.1.	Παράδειγμα αλγόριθμου forward	42
Σχήμα 5.1.	Ενδιάμεσος σε δύο κοινότητες κόμβος	49
Σχήμα 5.2.	Παράδειγμα 2 πλήρων γράφων που ενώνονται με ένα κόμβο	50
Σχήμα 5.3.	Παράδειγμα δενδροδιαγράμματος	50
Σχήμα 5.4.	Σχηματική αναπαράσταση βημάτων του αλγόριθμου Louvain	57
Σχήμα 5.5.	Αποσυνδεδεμένες κοινότητες εξόδου και μια κρυφή στιβάδα	59
Σχήμα 5.6.	Σχηματική αναπαράσταση βημάτων του αλγόριθμου Leiden	62
Σχήμα 5.7.	Γραφική αναπαράσταση τυχαίου περιπάτου	66
Σχήμα 5.8.	Γραφική αναπαράσταση τυχαίου περιπάτου	70
Σχήμα 5.9.	Παράδειγμα εύρεσης k-κλικών	72
Σχήμα 6.1.	Παράδειγμα δικτύων για διάφορους εσωτερικούς βαθμούς κόμβων.	78
Σχήμα 6.2.	Παράδειγμα ενός Gaussian μοντέλου με τέσσερις συστάδες	79
Σχήμα 6.3.	Παράδειγμα ενός LFR δικτύου αναφοράς	81
Σχήμα 6.4.	Τιμές NMI των αλγορίθμων για διάφορες τιμές παραμέτρου μίξης	85
Σχήμα 6.5.	Αποτελέσματα modularity για τον κάθε αλγόριθμο	86
Σχήμα 6.6.	Σύγκριση πλήθους κοινοτήτων των αλγορίθμων	87
Σχήμα 6.7.	Χρόνοι για την εύρεση κοινοτήτων που χρειάστηκε κάθε αλγόριθμος	88
Σχήμα 6.8.	Χρόνοι για την εύρεση τριγώνων για δίκτυα αναφοράς LFR	89
Σχήμα 6.9.	Χρόνοι για την εύρεση τριγώνων για δίκτυα Erdos Renyi	90

Σχήμα 6.10.	Αποτελέσματα NMI και Modularity των συνόλων Email και DBLP	93
Σχήμα 6.11.	Αποτελέσματα Modularity για τα σύνολα Facebook και GitHub	94
Σχήμα 6.12.	Απαιτούμενος χρόνος για τον υπολογισμό των κοινοτήτων των πραγματικών δεδομένων	96
Σχήμα 6.13.	Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του email	98
Σχήμα 6.14.	Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του DBLP	99
Σχήμα 6.15.	Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του GitHub	99
Σχήμα 6.16.	Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του Facebook	100

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Τα κοινωνικά δίκτυα είναι πλατφόρμες οι οποίες επιτρέπουν στους χρήστες να επικοινωνούν εύκολα, γρήγορα και από οπουδήποτε βρίσκονται. Είναι γεγονός ότι όλοι χρησιμοποιούν πλέον τα κοινωνικά δίκτυα σχεδόν καθημερινά και αρκετές φορές εντός της μέρας. Μέσα από τις πλατφόρμες κοινωνικών δικτύων, μπορεί κάποιος πέρα από το να επικοινωνήσει με φίλους, συγγενείς κλπ. να βρει χρήσιμες πληροφορίες για πράγματα που τον ενδιαφέρουν (π.χ. άρθρα) ή και για πράγματα που χρειάζεται για τη δουλειά ή τις σπουδές του (π.χ. μέσω GitHub) μέσα από τις αναρτήσεις των υπολοίπων χρηστών. Σε έρευνα της Eurostat (Eurostat,2023), παρατηρείται πως τα άτομα ηλικίας 16-29 ετών χρησιμοποιούν το διαδίκτυο σχεδόν σε καθημερινή βάση με κύριο λόγο τις πλατφόρμες κοινωνικών δικτύων. Στην έρευνα παρατηρήθηκε πως πάνω από το 96% των ατόμων ηλικιακής ομάδας 16-29 που χρησιμοποιεί καθημερινά το διαδίκτυο. Ενώ όπως παρατηρείτε στο Σχήμα 1.1 πάνω από το 80% χρησιμοποιούν τα κοινωνικά δίκτυα.



Σχήμα 1.1. Χρήση διαφόρων λειτουργειών του διαδικτύου για ηλικίες 16-29 ετών (Eurostat, 2023).

Όπως αντιλαμβανόμαστε, η εξόρυξη γνώσης σε δεδομένα κοινωνικών δικτύων είναι ένα απαραίτητο εργαλείο αφού μπορεί να μας δώσει ποικίλες πληροφορίες. Για παράδειγμα, ένα τμήμα έρευνας αγοράς μπορεί να κατανοήσει αν μια διαφήμιση που θέλουν να ετοιμάσουν θα είναι χρήσιμη για την εταιρεία. Ένα άλλο παράδειγμα είναι οι ίδιες οι πλατφόρμες κοινωνικών δικτύων, οι οποίες μπορούν να χρησιμοποιήσουν τα δεδομένα τους σωστά για να βελτιώσουν τις εφαρμογές τους. Βέβαια όπως μπορούμε να καταλάβουμε ο όγκος των δεδομένων των δικτύων είναι γιγαντιαίος, με αποτέλεσμα τα συστήματα και οι αλγόριθμοι που βρίσκονται πίσω από την εξόρυξη δεδομένων να μελετιούνται και να εξελίσσονται συνεχώς. Έτσι στη παρούσα εργασία θα μελετήσουμε διάφορους αλγορίθμους εξόρυξης γνώσης από κοινωνικά δίκτυα και θα τους συγκρίνουμε ως προς την αποτελεσματικότητα αλλά και τη λειτουργικότητα τους.

Αρχικά, θα μελετήσουμε και θα ορίσουμε τι είναι τα κοινωνικά δίκτυα αλλά και πως μπορούμε να τα αναπαραστήσουμε σε φυσική μορφή ως γράφους. Έπειτα, θα αναλύσουμε διάφορα χρήσιμα μέτρα και αλγορίθμους τα οποία χρησιμοποιούνται για τη κατανόηση της δομής ενός δικτύου και θα αναλύσουμε πως μπορούν να εφαρμοστούν σε πραγματικά προβλήματα. Τέλος, θα παρουσιάσουμε αποτελέσματα από εφαρμογές των τεχνικών σε συνθετικά αλλά και πραγματικά δεδομένα.

ΚΕΦΑΛΑΙΟ 2

Γράφοι και κοινωνικά δίκτυα

2.1 Κοινωνικά δίκτυα

Όταν αναφερόμαστε σε κοινωνικά δίκτυα, συνήθως σκεφτόμαστε τα δημοφιλή ονόματα, όπως το Facebook, το Instagram, το πρώην Twitter (γνωστό πλέον ως X) και άλλες παρόμοιες ιστοσελίδες ή εφαρμογές. Αυτές οι πλατφόρμες αντιπροσωπεύουν πραγματικά τα κοινωνικά δίκτυα, καθώς πληρούν τις απαραίτητες προϋποθέσεις για να χαρακτηριστούν ως τέτοια. Οι βασικές προϋποθέσεις που πρέπει να πληροί ένα δίκτυο για να θεωρηθεί κοινωνικό (Rajaraman, 2011) είναι οι εξής

1) Υπάρχει μια συλλογή οντοτήτων που συμμετέχουν στο δίκτυο. Συνήθως, αυτές οι οντότητες είναι άνθρωποι, αλλά θα μπορούσαν να είναι και κάτι εντελώς διαφορετικό. Ως μια ξεχωριστή οντότητα, θα μπορούσε να θεωρηθεί και μια ομάδα ατόμων.

2) Υπάρχει τουλάχιστον μία σχέση μεταξύ των οντοτήτων του δικτύου. Στο Facebook ή σε άλλα παρόμοια δίκτυα, αυτή η σχέση ονομάζεται φίλοι. Μερικές φορές οι σχέσεις που σχηματίζονται ανάμεσα σε οντότητες είναι απόλυτη. Για παράδειγμα, στο Facebook δύο άτομα είναι φίλοι ή όχι και δεν μπορεί να υπάρξει κάποια ενδιάμεση κατάσταση. Αντίθετα, υπάρχουν κοινωνικά δίκτυα που η σχέση μεταξύ δύο οντοτήτων έχει κάποιο βαθμό. Για παράδειγμα στο LinkedIn, μπορεί να είσαι απευθείας συνδεδεμένος με κάποιο άτομο (παρόμοιο με το να είστε φίλοι στο Facebook) ή μπορεί απλά να ακολουθείς ένα άτομο(όπως στο Instagram) η οποία σύνδεση είναι πιο αδύναμη από το να είστε απευθείας συνδεμένοι μεταξύ σας. Οι σχέσεις των οντοτήτων σε αυτή τη περίπτωση θα μπορούσαν να αναπαρασταθούν από μια διακριτή τιμή ή θα μπορούσε να είναι ένας πραγματικός

αριθμός. Ως παράδειγμα θα μπορούσαμε να έχουμε τη μέση συχνότητα επικοινωνίας μεταξύ δύο ατόμων κατά τη διάρκεια της ημέρας.

3) Υπάρχει η υπόθεση της μη τυχαιότητας ή της τοπικότητας. Αυτή η συνθήκη είναι η πιο δύσκολη να διατυπωθεί, αλλά διαισθητικά θα λέγαμε πως είναι ότι οι σχέσεις τείνουν να συγκεντρώνονται. Δηλαδή, αν η οντότητα Α σχετίζεται τόσο με τη Β όσο και με τη Γ, τότε αναμένεται πως η Β σχετίζεται με τη Γ.

2.2 Διάφορες κατηγορίες κοινωνικών δικτύων

Υπάρχουν διάφορα παραδείγματα κοινωνικών δικτύων εκτός από τα δίκτυα "φίλων". Παρακάτω παρουσιάζονται μερικά παραδείγματα τα οποία παρουσιάζουν τοπικότητα των σχέσεων, πέρα των υπολοίπων προϋποθέσεων.

a. Τηλεφωνικά δίκτυα

Εδώ οι υπό μελέτη οντότητες, αντιπροσωπεύουν τηλεφωνικούς αριθμούς, οι οποίοι στην πραγματικότητα είναι άτομα (ή ομάδες ατόμων, όπως οικογένειες). Υπάρχει σχέση μεταξύ δύο κόμβων εάν έχει πραγματοποιηθεί μια κλήση μεταξύ αυτών των τηλεφώνων σε κάποια καθορισμένη χρονική περίοδο, όπως τον τελευταίο μήνα ή "ποτέ". Η δύναμη των σχέσεων μεταξύ οντοτήτων, θα μπορούσαν να σταθμιστούν με βάση τον αριθμό των κλήσεων που πραγματοποιήθηκαν μεταξύ αυτών των τηλεφώνων κατά τη διάρκεια μιας περιόδου. Οι κοινότητες σε ένα τηλεφωνικό δίκτυο σχηματίζονται από ομάδες ανθρώπων που επικοινωνούν συχνά, όπως ομάδες φίλων, μέλη μιας αθλητικής ομάδας ή άνθρωποι που εργάζονται στην ίδια εταιρεία.

b. Δίκτυα ηλεκτρονικού ταχυδρομείου

Σε τέτοιου είδους δίκτυα μελετάμε διευθύνσεις ηλεκτρονικού ταχυδρομείου, οι οποίες είναι και πάλι άτομα (μπορεί να είναι και ομάδα ατόμων). Οι σχέσεις μεταξύ των διευθύνσεων, αντιπροσωπεύει το γεγονός ότι υπήρξε τουλάχιστον ένα μήνυμα ηλεκτρονικού ταχυδρομείου προς μία τουλάχιστον κατεύθυνση μεταξύ των δύο διευθύνσεων. Εναλλακτικά, θα μπορούσαμε να θεωρήσουμε ότι σχετίζονται, μόνο εάν υπήρχαν μηνύματα ηλεκτρονικού ταχυδρομείου και προς τις δύο κατευθύνσεις. Με αυτόν τον τρόπο, αποφεύγουμε να θεωρούμε τους spammers ως "φίλους" με όλα τα θύματά τους. Μια άλλη προσέγγιση είναι να χαρακτηρίσουμε τις σχέσεις ως αδύναμες

ή ισχυρές. Οι ισχυρές σχέσεις αντιπροσωπεύουν επικοινωνία και προς τις δύο κατευθύνσεις, ενώ οι αδύναμες υποδηλώνουν ότι η επικοινωνία ήταν μόνο προς μία κατεύθυνση. Οι κοινότητες που παρατηρούνται στα δίκτυα ηλεκτρονικού ταχυδρομείου προέρχονται από τα ίδια είδη ομαδοποιήσεων που αναφέραμε σε σχέση με τα τηλεφωνικά δίκτυα. Ένα παρόμοιο είδος δικτύου περιλαμβάνει ανθρώπους που στέλνουν μηνύματα σε άλλους ανθρώπους μέσω των κινητών τους τηλεφώνων.

c. Δίκτυα συνεργατών

Στα δίκτυα συνεργατών (Collaboration networks) οι οντότητες που μελετάμε αποτελούνται από άτομα (ή ομάδες) που έχουν δημοσιεύσει ερευνητικές εργασίες. Εάν υπάρχουν σχέσεις μεταξύ δύο οντοτήτων τότε μπορούμε να θεωρήσουμε πως έχουν συνεργαστεί και δημοσιεύσει από κοινού τουλάχιστον μια εργασία. Όσο δυνατότερη η σχέση που θα εμφανίσουν τόσο περισσότερες πρέπει οι συνεργασίες των δύο αυτών ατόμων. Οι κοινότητες σε αυτό το δίκτυο είναι συγγραφείς που εργάζονται πάνω σε ένα συγκεκριμένο θέμα. Μια εναλλακτική θεώρηση των ίδιων δεδομένων είναι ως δίκτυο στο οποίο οι οντότητες είναι εργασίες. Δύο εργασίες μπορεί να παρουσιάσουν σχέσεις, εάν έχουν τουλάχιστον έναν κοινό συγγραφέα. Τώρα, σχηματίζουμε κοινότητες οι οποίες είναι συλλογές εργασιών πάνω στο ίδιο θέμα.

Υπάρχουν πολλά άλλα είδη δεδομένων που σχηματίζουν δίκτυα με παρόμοιο τρόπο. Για παράδειγμα, μπορούμε να εξετάσουμε τα άτομα που επεξεργάζονται άρθρα της Wikipedia, όπως επίσης το είδος των άρθρων που επεξεργάζονται. Δύο συντάκτες συνδέονται αν έχουν επεξεργαστεί ένα κοινό άρθρο. Οι κοινότητες είναι ομάδες συντακτών που ενδιαφέρονται για το ίδιο θέμα. Αντίστροφα, μπορούμε να δημιουργήσουμε ένα δίκτυο άρθρων και να συνδέσουμε τα άρθρα αν έχουν επεξεργαστεί από το ίδιο άτομο. Εδώ, έχουμε κοινότητες άρθρων με παρόμοια ή συναφή θέματα. Δεδομένα τα οποία μας υποδεικνύουν τις σχέσεις μεταξύ πελάτη και προϊόντος, μπορούν να θεωρηθούν ότι σχηματίζουν ζεύγη δικτύων, ένα για τους πελάτες και ένα για τα προϊόντα. Οι πελάτες που αγοράζουν τα ίδια είδη προϊόντων, όπως βιβλία επιστημονικής φαντασίας, θα σχηματίσουν κοινότητες και αντίστροφα, τα προϊόντα που αγοράζονται από τους ίδιους πελάτες θα σχηματίσουν κοινότητες, για παράδειγμα όλα τα βιβλία επιστημονικής φαντασίας.

2.3 Γράφοι

Ως γράφο (δίκτυο) περιγράφουμε ένα διατεταγμένο ζεύγος $G = (V, E)$, το οποίο αποτελείται από ένα σύνολο κόμβων (ή κορφών) V και ένα σύνολο ακμών E. Μια ακμή ενώνει δύο κόμβους μεταξύ τους. Συμβολίζουμε το σύνολο (πληθικό αριθμό) του V (τάξη) με $|V| = n$ και το πλήθος ακμών του E με $|E| = m$. Σε έναν μη κατευθυνόμενο γράφο $G = (V, E)$ το σύνολο των ακμών θεωρείται ως ένα σύνολο $\{u, v\}$ δύο στοιχείων του V . Δύο κόμβοι u, v θεωρούνται γειτονικοί εάν ενώνονται από κάποια ακμή $\{u, v\} \in E$. Ένας γράφος που δεν έχει self-loops δηλαδή $\forall u \in V | (u, u) \notin E$ ονομάζεται απλός. Στα κοινωνικά δίκτυα είναι πολύ σπάνιο το φαινόμενο να υπάρχουν self-loops. Γενικά, θεωρούμε ότι ένας μη-κατευθυνόμενος γράφος G είναι απλός όταν ισχύει

$$m \leq \binom{n}{2} = \frac{n(n-1)}{2}. \tag{2.1}$$

Ένας γράφος θεωρείται πλήρης όταν η σχέση (2.1) ισχύει ως ισότητα. Τέτοιου είδους γράφοι, με $m = \binom{n}{2}$ πλήθος ακμών ονομάζονται n-cliques. Οι κλίκες (cliques) σε έναν γράφο αντιπροσωπεύουν υποομάδες στις οποίες όλοι οι κόμβοι ενώνονται μεταξύ τους. Αυτές οι ομάδες ατόμων, που είναι στενά συνδεδεμένες μεταξύ τους, αποτελούν βασικό εργαλείο για την κατανόηση της δομής αλλά και της σύνδεσης των ομάδων μεταξύ τους. Ουσιαστικά μια κλίκα, σε ένα γράφο $G = (V, E)$, είναι ένα υποσύνολο κόμβων $C \subseteq V$ τέτοιο ώστε ο γράφος $H = (C, E_1)$ να είναι πλήρης. Ο λόγος

$$\rho(G) = \frac{m}{\binom{n}{2}} \tag{2.2}$$

ονομάζεται πυκνότητα του γράφου.

2.3.1 Βασικά χαρακτηριστικά ενός δικτύου

Οι γείτονες σε έναν γράφο αποτελούν ένα σημαντικό κομμάτι της δομής του, προσφέροντας πλούσιες πληροφορίες για τη σχέση μεταξύ των κόμβων. Οι γείτονες ενός κόμβου αναφέρονται στους άμεσα συνδεδεμένους κόμβους με αυτόν. Σε έναν κοινωνικό γράφο, οι γείτονες ενός ατόμου αντιστοιχούν στους φίλους και τις επαφές του. Το σύνολο των γειτόνων $N(v)$ ενός κόμβου v είναι το σύνολο των κόμβων οι οποίοι έχουν μια ακμή που τους συνδέει με τον v

$$N(v) = \{u \epsilon V : (u, v) \epsilon E\}.$$

Οι γείτονες ενός υποσυνόλου S του V ορίζεται ως

$$N(S) = \{\forall s \in S, \exists v \in V \supset S | (s, v) \in E\}.$$

Ο βαθμός (degree) $d(v)$ ενός κόμβου v ορίζεται ως το πλήθος των ακμών ή ισοδύναμα ο αριθμός των γειτόνων του

$$d(v) = |N(v)|.$$

Ο μεγαλύτερος βαθμός ενός γράφου G ορίζεται ως

$$d_{max} = \max \{d(v): v \in V\}.$$

Σε κατευθυνόμενους γράφους οι γείτονες χωρίζονται σε δύο κατηγορίες, τους εσωτερικούς (In neighbors) και τους εξωτερικούς (Out neighbors) γείτονες. Εσωτερικούς γείτονες λέμε τους κόμβους που ενώνονται με τον υπό μελέτη κόμβο και η κατεύθυνση της ακμής ¨δείχνει¨ τον κόμβο αυτό. Τους εσωτερικούς γείτονες ενός κατευθυνόμενου γράφου $D = (V, A)$, τους συμβολίζουμε με I και ορίζονται ως εξής,

$$I(u) = \{v \in V: (v, u) \in A)\}.$$

Αντίθετα εξωτερικούς γείτονες ονομάζουμε τους κόμβους που ενώνονται με τον υπό μελέτη κόμβο αλλά η ακμή έχει κατεύθυνση προς τους κόμβους που είναι γείτονες. Τους εξωτερικούς γείτονες ενός κατευθυνόμενου γράφου $D = (V, A)$, τους συμβολίζουμε με O και ορίζονται ως εξής,

$$O(u) = \{v \in V: (u, v) \in A)\}.$$

Για παράδειγμα στο Σχήμα 2.1 οι εσωτερικοί γείτονες του κόμβου 2 είναι οι 1 και 3 ενώ ως εξωτερικό γείτονα θα θεωρούσαμε μόνο τον 3.

Ένα ακόμα βασικό χαρακτηριστικό σε ένα δίκτυο, είναι ο τρόπος μετάδοσης της πληροφορίας μέσα σε αυτό. Για παράδειγμα, για να φτάσει μια πληροφορία από έναν κόμβο σε έναν άλλον, μπορεί να χρειαστεί να ακολουθήσει μια διαδρομή όπου θα ‘περάσει’ πρώτα από διάφορους άλλους κόμβους. Τα μονοπάτια σε έναν γράφο αντιπροσωπεύουν τη μικρότερη διαδρομή από έναν κόμβο σε έναν άλλον. Όσο μικρότερο είναι το μονοπάτι που πρέπει να ακολουθήσουμε για να φτάσουμε σε έναν κόμβο τόσο πιο κοντά σημαίνει πως βρίσκεται ο κόμβος αυτός.

2.3.2 Πίνακες αναπαράστασης

Στη θεωρία γράφων, οι πίνακες αναπαράστασης χρησιμοποιούνται για να αποθηκεύσουμε διάφορες δομικές πληροφορίες ενός γράφου σε μαθηματική μορφή. Οι πίνακες μας προσφέρουν ένα ουσιαστικό εργαλείο για τη κατανόηση της δομής των δικτύων και αξιοποιούνται από διάφορους αλγόριθμους κατά την μελέτη των δικτύων.

Ο πίνακας γειτνίασης (adjacency matrix) $A = \left(\alpha_{ij}\right)_{n \times n}$ του γράφου G όπου

$$\alpha_{ij} = \begin{cases} w_{ij}, & \alpha\nu\ (v_i, v_j)\epsilon E \\ 0, & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Ο πίνακας αναπαριστά τη σχέση μεταξύ των στοιχείων όπου η κάθε στήλη (ή αντίστοιχη γραμμή) αναπαριστά ένα μοναδικό κόμβο (άτομο). Κάθε στοιχείο (α_{ij}) του πίνακα αναπαριστά τη σχέση μεταξύ του ατόμου i με το άτομο j. Το μέγεθος του πίνακα είναι $n \times n$, όπου με n συμβολίζεται το πλήθος των κόμβων που μελετάμε. Στη περίπτωση όπου μελετάμε γράφους χωρίς βάρη, το $w_{ij} = 1$. Ο πίνακας είναι πολύ χρήσιμο εργαλείο και συμβάλει άμεσα στη στατιστική ανάλυση αλλά και στην περιγραφή της δομής των δικτύων. Σε μη-κατευθυνόμενους γράφους ο πίνακας γειτνίασης είναι συμμετρικός, δηλαδή $a_{ij} = a_{ji}$.

Ο πίνακας βαθμών D_{ij} χρησιμοποιείται κυρίως σε μη κατευθυνόμενους γράφους.

$$D_{ij} = \begin{cases} \deg(v_i), & \alpha\nu\ i = j \\ 0, & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Ο πίνακας βαθμών είναι διαγώνιος όπου στη κύρια διαγώνιο του υπάρχει ο βαθμός του αντίστοιχου κόμβου. Ο βαθμός του κόμβου σε ένα μη κατευθυνόμενο γράφο συμβολίζει το πλήθος των ακμών (πλήθος γειτόνων = $\deg(v_i)$) που τον ενώνουν με τους γειτονικούς του κόμβους. Σε περίπτωση που έχουμε ένα κατευθυνόμενο γράφο, με το βαθμό μπορούμε να συμβολίσουμε και το indegree και το outdegree, δηλαδή το πλήθος των εισερχόμενων και εξερχόμενων ακμών αντίστοιχα.

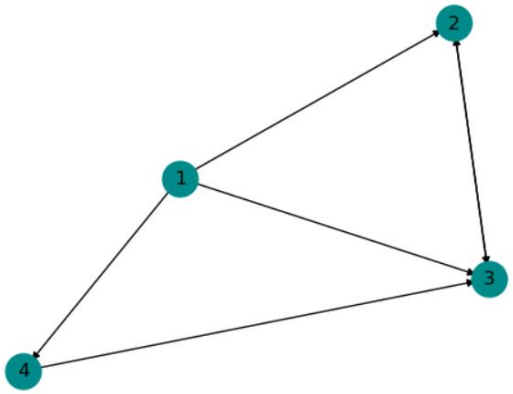
2.4 Κατηγορίες γράφων

Τα κοινωνικά δίκτυα μπορούν να μελετηθούν με διάφορα είδη γράφων. Ανάλογα με το σύνολο δεδομένων που αναλύουμε θα πρέπει να επιλέξουμε ένα γράφο, με τον οποίο μπορούμε να

αναπαραστήσουμε το σύνολο αυτό. Στη παρούσα διπλωματική θα αναλυθούν και θα χρησιμοποιηθούν μη-κατευθυνόμενοι γράφοι. Σε συνέχεια παρουσιάζονται γράφοι που θα μπορούσαν να χρησιμοποιηθούν σε διαφορετικά σύνολα δεδομένων κοινωνικών γράφων.

a. Κατευθυνόμενοι γράφοι (directed graphs)

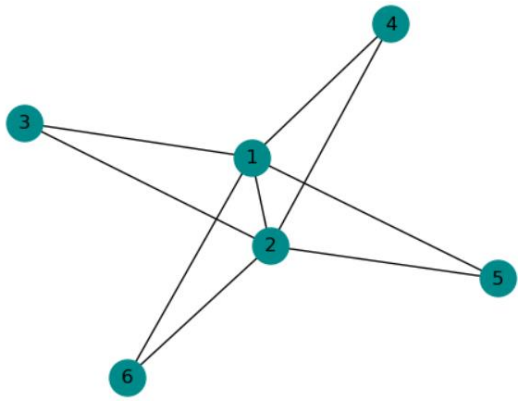
Η πιο απλή εκδοχή γράφων είναι οι κατευθυνόμενοι. Κατευθυνόμενος (ή προσανατολισμένος) ονομάζεται ένας γράφος $G = (V, A)$ που αποτελείτε από ένα σύνολο κορυφών V και ένα σύνολο A ακμών που ονομάζονται τόξα. Υπάρχουν περιπτώσεις δικτύων που παρατηρείται μια μονόπλευρη ένωση μεταξύ ακμών, δηλαδή όταν είμαστε σε μια ακμή Α να μπορούμε να μετακινηθούμε στη Β αλλά όχι το αντίθετο. Αυτές είναι περιπτώσεις δικτύων που αναπαρίστανται από κατευθυνόμενους γράφους. Όταν για παράδειγμα ένα άτομο ακολουθεί κάποιο άλλο σε ένα δίκτυο όπως το Instagram, δεν σημαίνει ότι θα ισχύει και το αντίθετο. Σε έναν κατευθυνόμενο γράφο αν δεν υπάρχουν κορυφές που ενώνονται με περισσότερες από δύο ακμές (τόξα) τότε ο γράφος αυτός ονομάζεται αντισυμμετρικός. Δηλαδή, για κάθε $(u, v) \in A$ το αντίστοιχο $(v, u) \notin A$.



Σχήμα 2.1. Κατευθυνόμενος γράφος

b. Μη-κατευθυνόμενοι γράφοι (undirected graphs)

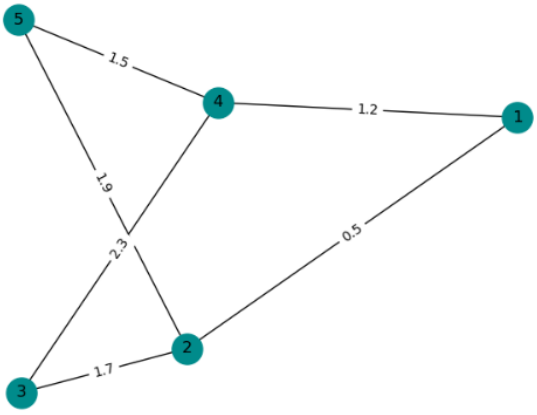
Όταν δύο ακμές ενώνονται αμφίδρομα, δηλαδή αφού έχουν ενωθεί μπορεί να μετακινηθούμε από την πρώτη στην δεύτερη και το αντίστροφο, τότε έχουμε ένα μη κατευθυνόμενο τύπο γράφου. Ένα απλό παράδειγμα είναι όταν έχουμε φίλους σε ένα κοινωνικό δίκτυο, όπως το Facebook, όπου αν ένα άτομο Α είναι φίλος με ένα άτομο Β τότε και το άτομο Β είναι φίλος με το άτομο Α.



Σχήμα 2.2. Μη-Κατευθυνόμενος γράφος

c. Γράφοι με βάρη (weighted graph)

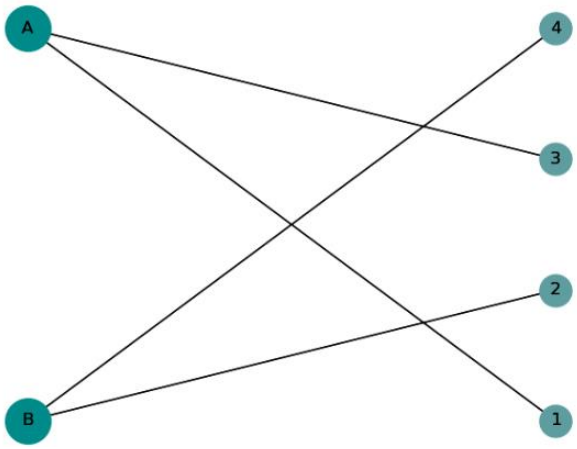
Στα περισσότερα δίκτυα που μελετάμε μας ενδιαφέρει αν έχει υπάρξει κάποια αλληλεπίδραση μεταξύ δύο ατόμων, όπου η σχέση μεταξύ των ατόμων παρουσιάζεται με μια δυαδική μορφή (υπάρχει σχέση ή όχι). Υπάρχουν όμως περιπτώσεις που θα μας ενδιέφερε και η συχνότητα των αλληλεπιδράσεων. Για παράδειγμα, μπορεί να μας ενδιαφέρει η συχνότητα επικοινωνίας μεταξύ ατόμων, όπου αν δύο άτομα ανταλλάσσουν καθημερινά e-mail, μπορούμε να θεωρήσουμε ότι έχουν μεγαλύτερη αλληλεπίδραση μεταξύ τους από δύο άτομα που επικοινωνούν μια φορά τον μήνα. Σε τέτοιες περιπτώσεις τη σχέση δεν θα τη συμβολίσουμε με 0 και 1, αλλά θα δίνεται μια αριθμητική τιμή που θα αντιπροσωπεύει το βάρος της συνένωσης (π.χ. συχνότητα επικοινωνίας) μεταξύ των στοιχείων.



Σχήμα 2.3. Γράφος με βάρη

d. Πολυμερείς γράφοι

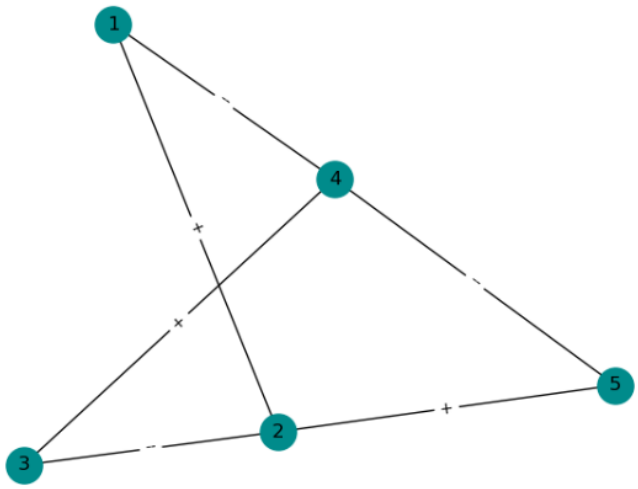
Μια ιδιαίτερη κατηγορία κοινωνικών δικτύων είναι τα πολυμερή δίκτυα (k-partite graph). Στα πολυμερή δίκτυα οι κόμβοι χωρίζονται σε διάφορες ομάδες. Κάθε κόμβος ενός συνόλου μπορεί να συνδεθεί μόνο με κόμβους ενός άλλου συνόλου. Οι κόμβοι της κάθε ομάδας μπορεί να αντιπροσωπεύουν διαφορετικού είδους δεδομένα. Για παράδειγμα μελετάμε ένα δίκτυο για τα σχόλια που αφήνουν χρήστες σε διάφορα κανάλια του YouTube, τότε θα υπάρχουν δύο ομάδες κόμβων. Η μία ομάδα θα αποτελείται από άτομα που αφήνουν σχόλια σε κάποιο κανάλι και η δεύτερη ομάδα θα αποτελείται από τα ίδια τα κανάλια τα οποία λαμβάνουν τα σχόλια. Το συγκεκριμένο δίκτυο θεωρείται ένα διμερές (bipartite ή 2-partite) δίκτυο, αφού αποτελείται από δύο ομάδες κόμβων (τους χρήστες και τα κανάλια). Γενικότερα, τέτοιου είδους δίκτυα είναι κατάλληλα για να αντιπροσωπεύσουν σύνολα δεδομένων τα οποία μπορούν να περιγράψουν σύνολα, όπου οι κόμβοι ανήκουν σε k διαφορετικές ομάδες. Ως παράδειγμα ενός πολυμερούς δικτύου θα μπορούσαμε να θεωρήσουμε ένα σύνολο ταινιών το οποίο περιέχει τους χρήστες που το παρακολούθησαν, τους ηθοποιούς της ταινίας, το σύνολο των ταινιών, το είδος των ταινιών κ.λπ. Όπως αντιλαμβανόμαστε, δεν χρειάζεται όλες οι ομάδες να έχουν ενώσεις μεταξύ τους. Στο παράδειγμα που προαναφέραμε οι χρήστες που παρακολουθούν ταινίες δεν χρειάζεται να έχουν απαραίτητα κάποια σύνδεση με τους ηθοποιούς.



Σχήμα 2.4. Διμερής γράφος

e. Προσημικοί γράφοι (Singed)

Η ανάλυση των σχέσεων των ανθρώπων, είτε σε προσωπικό είτε σε επαγγελματικό πλαίσιο, έχει μεγάλη σημασία στη κοινωνία μας. Η κατανόηση και η αξιολόγηση αυτών των σχέσεων είναι θεμελιώδους σημασίας για τη βελτίωση της επικοινωνίας, της συνεργασίας και της συνολικής ευημερίας. Στην προσωπική μας ζωή, η ποιότητα των σχέσεών μας έχει βαθύτατο αντίκτυπο στη συναισθηματική μας υγεία και ευτυχία, καθιστώντας ουσιαστικό τον εντοπισμό τομέων προς βελτίωση, την επίλυση συγκρούσεων και τη διατήρηση υγιών δεσμών. Στον εργασιακό χώρο, η αποτελεσματική ανάλυση των σχέσεών είναι ζωτικής σημασίας για τη συνεργασία της ομάδας, την ηγεσία και την παραγωγικότητα. Βοηθά στην ενίσχυση της εμπιστοσύνης, στην ευθυγράμμιση των συμφερόντων και στη βελτιστοποίηση των διαδικασιών λήψης αποφάσεων. Η ανάλυση σχέσεων είναι επίσης απαραίτητη σε τομείς πολύ διαφορετικούς όσο οι σχέσεις με τους πελάτες, η πολιτική, η διπλωματία και η επίλυση συγκρούσεων, επιτρέποντάς μας να περιηγηθούμε σε πολύπλοκες κοινωνικές δυναμικές, να προβλέψουμε τις προκλήσεις και να λάβουμε τεκμηριωμένες αποφάσεις. Οι προσημικοί γράφοι αποτυπώνουν την ποιότητα αλληλεπίδρασης μεταξύ ατόμων. Αν δύο άτομα συνεχώς διαφωνούν και καβγαδίζουν τότε θεωρούμε πως η μεταξύ τους σχέση είναι αρνητική, απεναντίας αν έχουν κοινές απόψεις και θετικές αλληλεπιδράσεις, τότε η μεταξύ τους σχέση είναι θετική. Για την οπτικοποίηση αυτών των σχέσεων οι προσημικοί γράφοι χρησιμοποιούν τα αντίστοιχα μαθηματικά σύμβολα (¨-¨,¨+¨)



Σχήμα 2.5. Προσημικός γράφος

Πέραν των βασικών τύπων γράφων, μπορούμε να χωρίσουμε τους γράφους και σε κάποιες βασικές υποκατηγορίες. Η πρώτη βασική υποκατηγορία που πρέπει να γνωρίζουμε είναι το αν υπάρχει συνένωση ή όχι μεταξύ όλων των κόμβων. Εάν από κάθε κορυφή υπάρχει μονοπάτι που μπορεί να μας οδηγήσει σε οποιαδήποτε άλλη κορυφή του γράφου τότε τον γράφο αυτόν, τον ονομάζουμε συνεκτικό (connected). Η δεύτερη υποκατηγορία είναι για να γνωρίζουμε το πόσο συνδεδεμένοι είναι οι κόμβοι μεταξύ τους. Αν το πλήθος των ακμών είναι πολύ μικρό, τότε θα ονομάζουμε τον γράφο αυτόν αραιό (Sparse). Αντίθετα, αν ο γράφος έχει ένα πολύ μεγάλο πλήθος ακμών (πχ όταν όλες σχεδόν οι κορυφές ενώνονται από μια ακμή) τότε ο γράφος θεωρείται πυκνός (Dense). Οι κατηγορίες αυτές είναι χρήσιμες να αναφερθούν μόνο σε κατευθυνόμενο, μη-κατευθυνόμενο ή γράφο με βάρη.

2.5 Τα κοινωνικά δίκτυα ως γράφοι

Τα κοινωνικά δίκτυα μπορούν να μοντελοποιηθούν και να αναπαρασταθούν ως γράφοι, τους οποίους μπορεί να συναντήσουμε με τον όρο κοινωνικοί γράφοι. Σε ένα κοινωνικό γράφο, οι χρήστες αναπαριστώνται ως κόμβοι, ενώ οι σχέσεις μεταξύ τους αναπαριστώνται από ακμές. Οι ακμές συνδέουν δύο κόμβους, αντιπροσωπεύοντας τη σχέση που τους ενώνει στο δίκτυο. Ο βαθμός της σχέσης συχνά αναπαρίσταται με την επισήμανση των ακμών. Στις περισσότερες περιπτώσεις, οι κοινωνικοί γράφοι είναι μη-κατευθυνόμενοι, όπως ο γράφος των φίλων στο Facebook. Ωστόσο, υπάρχουν και κατευθυνόμενοι γράφοι στα κοινωνικά δίκτυα, όπως οι γράφοι των ακολούθων στο Instagram, όπου η κατεύθυνση της σχέσης είναι σημαντική.

Πολλά άλλα φαινόμενα μπορούν να μελετηθούν και μοιάζουν με κοινωνικούς γράφους, ιδίως όταν παρουσιάζουν τοπικότητα. Χαρακτηριστικά τέτοια παραδείγματα αποτελούν τα δίκτυα πληροφοριών (έγγραφα, διαδικτυακοί γράφοι, πατέντες), δίκτυα υποδομών (δρόμοι, σωλήνες νερού, δίκτυα ηλεκτρικής ενέργειας), βιολογικά δίκτυα (γονίδια, πρωτεΐνες, τροφικά δίκτυα ζώων που τρώνε το ένα το άλλο), καθώς και άλλα είδη, όπως δίκτυα αγοράς προϊόντων όπου συνδέουν τον αγοραστή με τοπικούς εμπόρους (Airbnb, Groupon).

ΚΕΦΑΛΑΙΟ 3
Χρήσιμα μέτρα

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε τη δομή των κοινωνικών γράφων. Η ανάλυση δομής προσφέρει το πλαίσιο για την ανίχνευση προτύπων, την αναγνώριση επιρροής και την ανακάλυψη κρίσιμων σημείων στα κοινωνικά δίκτυα. Αναδεικνύει τις δομικές ιδιότητες των σχέσεων, επιτρέποντας μας να κατανοήσουμε πώς διαμορφώνονται οι κοινότητες, πώς επιδρούν οι κλίκες και ποιοι είναι οι κεντρικοί παράγοντες που καθορίζουν τη ροή των πληροφοριών.

3.2 Ασυμπτωτική ανάλυση χρόνου

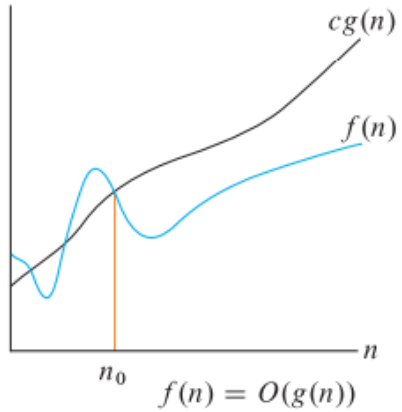
Προτού ξεκινήσουμε με την ανάλυση χρήσιμων μέτρων πρέπει να αναφέρουμε ότι σήμερα τα σύνολα δεδομένων κοινωνικών δικτύων είναι τεράστια. Αυτό δημιουργεί μεγάλο πρόβλημα κατά την ανάλυση τους, αφού οι αλγόριθμοι απαιτούν πολύ χρόνο και μνήμη για να μπορέσουν να χρησιμοποιηθούν. Για τον λόγο αυτό, δεν μας ενδιαφέρει μόνο η αποτελεσματικότητα ενός αλγορίθμου αλλά και η ταχύτητα και οι πόροι που απαιτεί ώστε να είναι εφαρμόσιμος.

Οι ασυμπτωτικοί συμβολισμοί παρέχουν ένα ισχυρό πλαίσιο για την ανάλυση και τη σύγκριση της αποδοτικότητας και της απόδοσης των αλγορίθμων, ιδίως όταν το μέγεθος του συνόλου δεδομένων αυξάνεται σημαντικά. Οι συμβολισμοί αυτοί παρέχουν έναν τρόπο περιγραφής της συμπεριφοράς συναρτήσεων και αλγορίθμων καθώς πλησιάζουν ορισμένα όρια. Με τη χρήση ασυμπτωτικών συμβολισμών μπορούμε να εκφράσουμε συνοπτικά την πολυπλοκότητα των αλγορίθμων, εστιάζοντας στις βασικές πτυχές της απόδοσής τους. Οι συμβολισμοί αυτοί παίζουν σημαντικό ρόλο στην επιστήμη των υπολογιστών, βοηθώντας στην αξιολόγηση, σύγκριση και επιλογή αλγορίθμων με βάση τη χρονική και χωρική πολυπλοκότητά τους. Η κατανόηση των

ασυμπτωτικών συμβολισμών είναι θεμελιώδης για την αποτελεσματική αξιολόγηση της αποδοτικότητας των αλγορίθμων και τη λήψη τεκμηριωμένων αποφάσεων στο σχεδιασμό και την ανάλυση αλγορίθμων.

Το σύμβολο O συμβολίζει ότι η συνάρτηση που μελετάμε είναι ασυμπτωτικά άνω φραγμένη. Πιο συγκεκριμένα, αν f και g είναι δύο συναρτήσεις τότε ορίζουμε

$$O(g(n)) = \{f(n): \alpha\nu\ \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} \mid 0 \leq f(n) \leq cg(n)\ \forall n \geq n_0\}$$

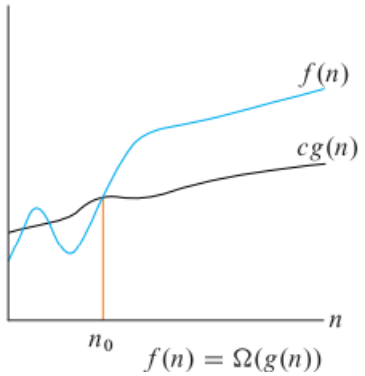


Σχήμα 3.1. Γραφική αναπαράσταση σύμβολου O (Cormen et al. 2022).

και σε αυτή τη περίπτωση θα λέμε ότι η f ανήκει στην $O(g)$ $(f(n) \in O(g(n)))$. Αντίστοιχα, με το σύμβολο Ω συμβολίζουμε τις συναρτήσεις οι οποίες είναι κάτω φραγμένες. Αν f και g δύο συναρτήσεις. Ορίζουμε

$$\Omega(g(n)) = \{f(n): \alpha\nu\ \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} \mid 0 \leq cg(n) \leq f(n)\ \forall n \geq n_0\}$$

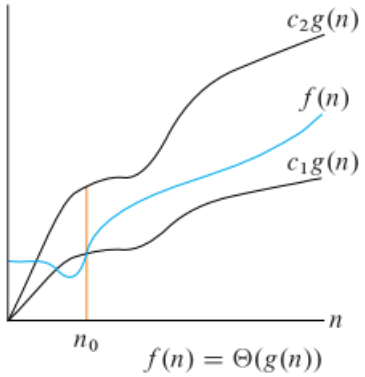
και τότε λέμε ότι η f ανήκει στην $\Omega(g)$ $(f(n) \in \Omega(g(n)))$.



Σχήμα 3.2. Γραφική αναπαράσταση σύμβολου Ω (Cormen et al. 2022).

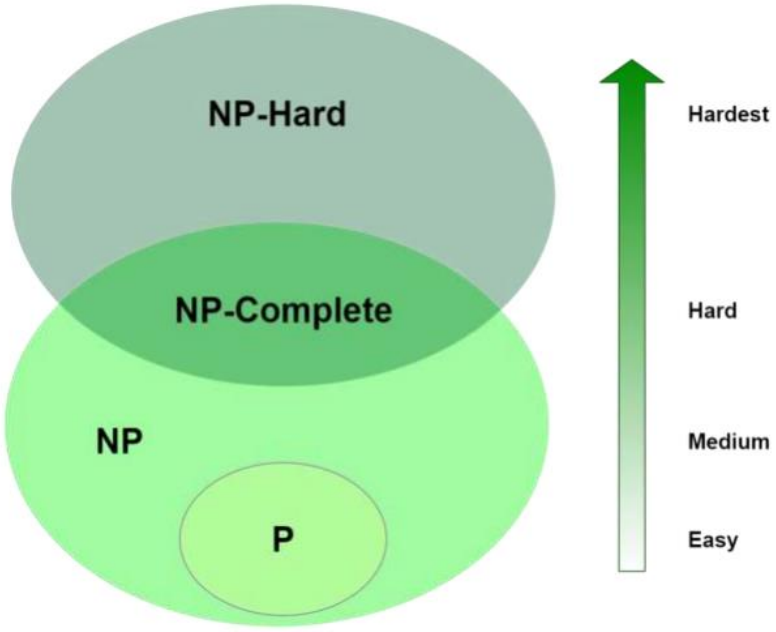
Τέλος, θα παρουσιάσουμε το σύμβολο Θ. Μια συνάρτηση f ανήκει στην $\Theta(g)$ ($f(n) \in \Theta(g(n))$), όταν είναι ταυτόχρονα ασυμπτωτικά άνω και κάτω φραγμένη. Έτσι αν $f(n)$ και $g(n)$ είναι δύο συναρτήσεις ορίζουμε

$$\Theta(g(n)) = \{f(n): \alpha\nu\ \exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N} \mid c_1 g(n) \leq f(n) \leq c_2 g(n)\ \forall n \geq n_0\}.$$



Σχήμα 3.3. Γραφική αναπαράσταση σύμβολου Θ (Cormen et al. 2022).

Στην επιστήμη των υπολογιστών τα προβλήματα μπορούν να κατηγοριοποιηθούν βάσει της δυσκολίας επίλυσής τους (Σχήμα 3.4). Αρχικά έχουμε τα προβλήματα της κατηγορίας P (polynomial), όπου ανήκουν προβλήματα για τα οποία μπορούμε να βρούμε λύση, σε πολυωνυμικό χρόνο. Στη κατηγορία αυτή ανήκουν προβλήματα όπως απλές μαθηματικές πράξεις (π.χ. πολλαπλασιασμός), εύρεση κοντινότερου μονοπατιού σε έναν γράφο (αλγόριθμος Dijkstra) όπως και διάφορα ακόμα προβλήματα. Αντίθετα έχουμε τους NP (non-polynomial) προβλήματα των οποίων η λύση μπορεί να επιβεβαιωθεί σε πολυωνυμικό χρόνο, αλλά δεν μπορεί να βρεθεί λύση του, σε πολυωνυμικό χρόνο. Η NP-complete κατηγορία εμπεριέχει όλα τα προβλήματα, τα οποία ικανοποιούν την ιδιότητα ότι, για κάθε πρόβλημα που ανήκει στην κατηγορία NP-complete υπάρχει ένας αλγόριθμος πολυωνυμικού χρόνου ο οποίος μπορεί να μετατρέψει ένα πρόβλημα σε ένα άλλο NP-complete πρόβλημα. Η τελευταία κατηγορία είναι η NP-Hard της οποίας τα προβλήματα δεν είναι απλώς δύσκολο να λυθούν αλλά είναι δύσκολη ακόμα και η επιβεβαίωση της λύσης τους. Ουσιαστικά τα πρόβλημα που ταξινομούνται στην κατηγορία NP-Hard, είναι τουλάχιστον δύσκολο να λυθούν όσο ένα NP-Complete πρόβλημα ή ακόμα περισσότερο (Pokharel M, 2020).



Σχήμα 3.4. Ταξινόμηση προβλημάτων σε κλάσεις (Pokharel M, 2020).

3.3 Μέτρα απόστασης

Τα μέτρα αποστάσεων μεταξύ δύο παρατηρήσεων, μπορούν να μας βοηθήσουν να καθορίσουμε την ομοιότητα μεταξύ αυτών. Όσο πλησιέστερα βρίσκονται δύο σημεία θεωρούμε πως έχουν μεγαλύτερη πιθανότητα να ενωθούν. Εμπειρικά, αν δυο σημεία βρίσκονται αρκετά κοντά μεταξύ τους θωρούμε πως είναι πιο όμοια από δύο στοιχεία που έχουν μεγαλύτερη απόσταση μεταξύ τους. Υπάρχουν διάφορα μέτρα που μπορούν να χρησιμοποιηθούν για τον υπολογισμό της απόστασης μεταξύ σημείων. Παρακάτω θα αναφερθούν κάποια από τα μέτρα αυτά.

- **Ευκλείδεια απόσταση**

Το πιο γνωστό μέτρο για τον υπολογισμό αποστάσεων, είναι η ευκλείδεια απόσταση, η οποία υπολογίζει το μήκος της ευθείας γραμμής μεταξύ δύο σημείων. Έστω δύο σημεία $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \dots, x_{in})$ και $\boldsymbol{x_j} = (x_{j1}, x_{j2}, \dots, x_{jn})$, τότε η απόσταση μεταξύ των δύο αυτών σημείων δίνεται από τον τύπο

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{\sum_{r=1}^{n} (x_{ir} - x_{jr})^2} \tag{3.1}$$

- **City block**

Μια ακόμη χρήσιμη απόσταση είναι η απόσταση Manhattan (ή City block). Η απόσταση Manhattan είναι παρόμοια με την ευκλείδεια, με τη διαφορά ότι δεν βασίζεται σε τετραγωνικές αποκλίσεις, αλλά σε απόλυτες. Η απόσταση αυτή είναι πιο ανθεκτική σε ακραίες παρατηρήσεις και ορίζεται από τον τύπο

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sum_{r=1}^{n} |x_{ir} - x_{jr}|. \tag{3.2}$$

- **Minkowski**

Η απόσταση Minkowski αναφέρεται σε μια γενικευμένη μορφή μέτρου απόστασης που ενσωματώνει την ευκλείδεια απόσταση και την απόσταση Manhattan ως ειδικές περιπτώσεις. Ο μαθηματικός τύπος για τον υπολογισμό του είναι

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\sum_{r=1}^{n} |x_{ir} - x_{jr}|^{\lambda} \right)^{\frac{1}{\lambda}}. \tag{3.3}$$

Για $\lambda = 1$ έχουμε την απόσταση Manhattan, ενώ για $\lambda = 2$ την ευκλείδεια. Η απόσταση Minkowski χρησιμοποιείται ευρέως σε διάφορους τομείς, όπως η μηχανική μάθηση, η ανάλυση δεδομένων και η αναγνώριση μοτίβων για να μετρηθεί η ομοιότητα σε πολυδιάστατους χώρους. Η επιλογή του κατάλληλου λ εξαρτάται από τη φύση των δεδομένων και τους στόχους της ανάλυσης.

3.4 Μέτρα ομοιότητας

Πολλές φορές τα δεδομένα που μελετάμε δεν έχουν την έννοια του χώρου. Για παράδειγμα, αν θέλουμε να δούμε πως "ενώνονται" φίλοι σε μια εφαρμογή, δεν μπορούμε πάντα να ξέρουμε σε ποια περιοχή ζουν τα άτομα αυτά. Για το λόγο αυτό, δεν μπορούμε να υπολογίσουμε το πόσο όμοια είναι τα άτομα με βάση την περιοχή που ζουν, αλλά θα χρησιμοποιήσουμε άλλα χαρακτηριστικά όπως τους κοινούς φίλους, παρόμοιες σελίδες που ακολουθούν κλπ. Σε τέτοιες περιπτώσεις τα

μέτρα απόστασης που αναλύσαμε παραπάνω δεν μπορούμε να τα αξιοποιήσουμε και αντί αυτών χρησιμοποιούμε άλλου είδους μέτρα. Τα μέτρα ομοιότητας προσπαθούν βάσει των χαρακτηριστικών των στοιχείων, να αναγνωρίσουν ποια στοιχεία είναι όμοια μεταξύ τους.

- **Κοινοί γείτονες**

Το μέτρο αυτό δεν είναι τίποτε άλλο από πλήθος κοινών γειτόνων δύο στοιχείων. Αν A, B είναι δύο κόμβοι τότε το πλήθος των κοινών γειτόνων είναι

$$CN(A,B) = |N(A \cap B)| \tag{3.4}$$

όπου με $N(A)$ συμβολίζεται το πλήθος των γειτονικών στοιχείων που έχει ο κόμβος A. Η πολυπλοκότητα του αλγορίθμου είναι πολύ μικρή αφού ο χρόνος που απαιτεί για να λειτουργήσει περιορίζεται σε $O\big(d(A) + d(B)\big)$, όπου με $d(v)$ συμβολίζουμε το πλήθος των γειτόνων το κόμβου v. Ο δείκτης αυτός βασίζεται στη διαπίστωση πως, αν δύο κόμβοι έχουν μεγάλο πλήθος κοινών γειτόνων, τότε οι κόμβοι αυτοί είναι πιο όμοιοι. Βέβαια το πλήθος και μόνο των κοινών γειτόνων δεν είναι πάντα το πιο αξιόπιστο μέτρο για τον υπολογισμό της ομοιότητας δύο στοιχείων. Για τον λόγο αυτό, έχουν δημιουργηθεί και αξιοποιούνται διάφορα άλλα μέτρα ομοιότητας.

- **Ομοιότητα Jaccard**

Ο πιο απλός δείκτης που έχει δημιουργηθεί για να υπολογίζεται η ομοιότητα δύο στοιχείων είναι αυτός του Paul Jaccard. Ο δείκτης αυτός υπολογίζει το πλήθος των κοινών γειτόνων δύο στοιχείων και έπειτα το διαιρεί με το συνολικό πλήθος γειτόνων των δύο στοιχείων.

$$J(A,B) = \frac{|N(A \cap B)|}{|N(A \cup B)|}. \tag{3.5}$$

Το μεγάλο μειονέκτημα του δείκτη είναι η ευαισθησία του στο πλήθος των γειτονικών στοιχείων των κόμβων. Για παράδειγμα έστω ότι έχουμε δύο στοιχεία, Α και Β τα οποία έχουν και τα δύο 10 γειτονικούς κόμβους και το πλήθος των κοινών γειτονικών κόμβων είναι 5. Σε αυτή τη περίπτωση ο δείκτης Jaccard ισούται με 0.33 (=5/(10+10-5)). Εάν στο δίκτυο ο κόμβος Α χάσει έναν γείτονα, ενώ αντίθετα στον κόμβο Β προστεθεί ένας και το πλήθος κοινών γειτόνων παραμείνει ίσο, τότε ο δείκτης Jaccard θα ισούται και πάλι με 0.33 (=5/(11+9-5)). Αν κάποιος κόμβος είναι γειτονικός με τα περισσότερα στοιχεία του συνόλου, τότε δεν μπορεί να μας προσφέρει κάποια πληροφορία, σε αντίθεση με κόμβους που έχουν λίγους γείτονες. Για παράδειγμα, σε ένα δίκτυο αν δύο άτομα ακολουθούν έναν διάσημο (πχ κάποιον τραγουδιστή)

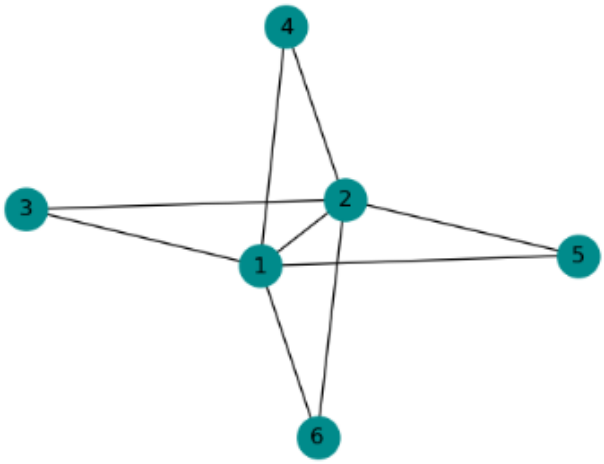
τότε αυτός ο γειτονικός κόμβος δεν θα πρέπει να έχει το ίδιο βάρος για στην ομοιότητα των στοιχείων, όσο ένας κόμβος που έχει λίγους γείτονες. Αν οι κόμβοι είναι ταξινομημένοι, τότε ο χρόνος που απαιτεί ο αλγόριθμος είναι της τάξεως $O\big(d(A)+d(B)\big)$. Σε αντίθετη περίπτωση, αν οι κόμβοι δεν είναι ταξινομημένοι τότε ο μέγιστος χρόνος που μπορεί να χρειαστεί ο αλγόριθμος είναι της τάξεως $O(m^2)$, όπου $m=\max\{d(A),d(B)\}$. Βέβαια θα μπορούσαμε προτού χρησιμοποιήσουμε τον αλγόριθμο, να ταξινομήσουμε τα δεδομένα το οποίο έχει κόστος τάξης $O\big(nlog(n)\big)$.

- **SimRank ομοιότητα**

Η SimRank ομοιότητα (Jeh and Widom, 2002) είναι μια μέθοδος για την αναπαράσταση ομοιότητας μεταξύ κόμβων. Η μέθοδος βασίζεται στη πεποίθηση ότι δύο κόμβοι είναι όμοιοι όταν έχουν παρόμοιους γείτονες. Έτσι, όσο πιο πολλές κοινές συνδέσεις έχουν δύο κόμβοι, τόσο πιο παρόμοιοι θα θεωρούμε πως είναι. Για τον υπολογισμό της ομοιότητας δύο κόμβων A,B χρησιμοποιούμαι τον παρακάτω τύπο

$$s(A,B)=\begin{cases}\dfrac{C}{|I(A)||I(B)|}\displaystyle\sum_{i=1}^{|I(A)|}\sum_{j=1}^{|I(B)|}s\left(I_i(A),I_j(B)\right), & \alpha\nu\ A\neq B\\ 1, & \alpha\nu\ A=B\end{cases} \tag{3.6}$$

όπου με $|I(X)|$ συμβολίζουμε το πλήθος των εσωτερικών γειτόνων του κόμβου X. Σε περίπτωση που ο ένας από τους δύο υπό μελέτη κόμβους δεν έχει γείτονες, τότε $s(A,B)=0$. Το C ονομάζεται σταθερά φθοράς με $C\in(0,1)$. Η τιμή της σταθεράς C που προτάθηκε αρχικά ήταν $C=0.8$. Βέβαια, έπειτα από δοκιμές που έγιναν για τη βελτίωση του αλγορίθμου αποφάνθηκε πως η τιμή αυτή δεν θα μας δίνει ικανοποιητικά αποτελέσματα για αρκετά σύνολα δεδομένων, αφού κόμβοι που θα θέλαμε να θεωρούνται παρόμοιοι έπαιρναν πολύ μικρές τιμές και έτσι πλέον συνηθίζεται να παίρνει την τιμή $C=0.9$. Η σταθερά φθοράς χρησιμοποιείται, διότι δεν θέλουμε να έχουμε στοιχεία όπου το μέτρο ομοιότητας τους ισούται με τη μονάδα.



Σχήμα 3.5. Κόμβοι συνδεδεμένοι με όλο το δίκτυο.

Στο παραπάνω παράδειγμα παρατηρούμε ότι όλοι οι κόμβοι είναι συνδεδεμένοι με τους κόμβους 1 και 2. Σε αυτή τη περίπτωση εάν η σταθερά φθοράς ήταν ίση με τη μονάδα θα είχαμε $s(1,2) \approx 1$, ενώ για $C = 0.9$ η ομοιότητα μεταξύ των ίδιων κόμβων είναι $s(1,2) \approx 0.64$. Για να υπολογίσουμε το SimRank μπορούμε να χρησιμοποιήσουμε διάφορους από τους αλγόριθμους που έχουν προταθεί. Παρακάτω θα παρουσιάσουμε έναν επαναληπτικό αλγόριθμο για τον υπολογισμό της SimRank ομοιότητας μεταξύ δύο στοιχείων. Αρχικά, ο αλγόριθμος θεωρεί πως όλα τα στοιχεία είναι πλήρως ανόμοια μεταξύ τους.

$$s_0(A,B) = \begin{cases} 0, & \alpha\nu\ A \neq B \\ 1, & \alpha\nu\ A = B \end{cases}$$

Για να υπολογίσει το $s_{k+1}(A,B)$ σε κάθε βήμα, χρησιμοποιεί τον τύπο

$$s_{k+1}(A,B) = \frac{C}{|I(A)||I(B)|} \sum_{i=1}^{|I(A)|} \sum_{j=1}^{|I(B)|} s_k\left(I_i(A), I_j(B)\right).$$

Ο αλγόριθμος επαναλαμβάνεται για k βήματα, το οποίο πλήθος μπορούμε να το προσαρμόσουμε ανάλογα με τις ανάγκες μας, αλλά συνήθως χρησιμοποιείται ένα πλήθος $k = 1000$ βημάτων. Βέβαια, το πλήθος των επαναλήψεων έχει σημαντικό ρόλο στη πρακτικότητα του αλγορίθμου και συνήθως δεν χρειάζονται τόσα πολλά βήματα, για το υπολογισμό της ομοιότητας των στοιχείων. Έτσι, για να μειώσουμε το πλήθος επαναλήψεων του αλγορίθμου με σκοπό τη γρηγορότερη απόκτηση της πληροφορίας, χρησιμοποιούμε ένα σημείο τερματισμού. Συνηθίζεται να σταματάει η διαδικασία όταν το μέτρο ομοιότητας συγκλίνει, δηλαδή όταν

$$\frac{|s_k(a,b) - s_{k-1}(a,b)|}{s_{k-1}(a,b)} < \varepsilon$$

όπου το ε προτείνεται να παίρνει τιμές μικρότερες του 10^{-6}. Χρησιμοποιώντας τον επαναληπτικό τύπο στο παράδειγμα του Σχήματος 3.5 για τον υπολογισμό της ομοιότητας κόμβων, θα βρούμε ότι οι υψηλότερες τιμές ομοιότητας είναι μεταξύ των εξωτερικών κόμβων (π.χ. 3 και 4), διότι έχουν ως κοινούς γείτονες μόνο τους κόμβους 1 και 2 οι οποίοι είναι αρκετά όμοιοι μεταξύ τους.

3.5 Μέτρα κεντρικότητας

Η κεντρικότητα του κάθε κόμβου αποτελεί ένα μέτρο σημαντικότητας, το οποίο αντικατοπτρίζει τη σημασία του κάθε κόμβου σε ένα δίκτυο. Η ανάλυση της σημαντικότητας των κόμβων, εξαρτάται από τον τύπο του εξεταζόμενου δικτύου. Για παράδειγμα, σε ένα κοινωνικό δίκτυο, είναι λογικό να θεωρηθούν σημαντικοί οι κόμβοι που ενώνονται με ένα μεγάλο αριθμό κόμβων. Σε αυτή την περίπτωση, ο μεγάλος αριθμός των συνδέσεων ενός κόμβου αντιστοιχεί σε ένα μεγάλο αριθμό σχέσεων φιλίας και φαίνεται να δίνει στο άτομο, μια κοινωνική θέση στο δίκτυο. Ωστόσο, σε ένα δίκτυο ροής πληροφοριών, μόνο το πλήθος των συνδέσεων ενός κόμβου ενδέχεται να μην είναι αρκετό για να τον χαρακτηρίσουν ως σημαντικό. Είναι λογικό να υποστηρίξει κανείς, ότι στα δίκτυα πληροφοριών ένας κόμβος είναι σημαντικός όταν μπορεί να ελέγξει τις πληροφορίες που διακινούνται εντός του δικτύου. Επομένως, η κεντρικότητά του δεν κρίνεται από την άμεση σύνδεσή του με άλλους κόμβους, αλλά από τη συχνή παρεμβολή του μεταξύ των μονοπατιών που χρησιμοποιούν τα ζεύγη κόμβων για τη μεταφορά πληροφοριών εντός του δικτύου. Έτσι, έχει προταθεί η χρήση διαφόρων μέτρων κεντρικότητας, τα οποία μπορούν να εφαρμοστούν ανάλογα με τον τύπο του δικτύου και τις ανάγκες της εκάστοτε έρευνας. Έστω πως υπάρχει μια εταιρεία που θέλει να διαφημίσει ένα προϊόν σε συνεργασία με ένα άτομο στο Facebook. Ο στόχος της εταιρείας είναι να δουν τη διαφήμιση όσο το δυνατόν περισσότερα άτομα. Όταν κάνει μια ανάρτηση ένας χρήστης που έχει πολλές συνδέσεις, αλλά δεν είναι κεντρικός, τότε θα δουν τη δημοσίευση μόνο οι φίλοι του. Αντίθετα ένας κεντρικός χρήστης που μπορεί να έχει λιγότερους φίλους, όταν θα αναρτήσει μια δημοσίευση, θα τη δουν και άτομα με τα οποία δεν είναι άμεσα συνδεδεμένος (πχ φίλοι φίλων). Για το λόγο αυτό, η εταιρεία που θέλει να κάνει μια

διαφήμιση πρέπει να επιλέξει το άτομο, βάσει της ικανότητάς του να εμφανίσει τη διαφήμιση (ανάρτηση που θα δημοσιεύσει) σε όσο το δυνατό περισσότερα άτομα.

- **Κεντρικότητα βαθμού**

Το μέτρο κεντρικότητας βαθμού (Degree Centrality) είναι ίσως το πιο απλό μέτρο που μπορεί να αναλύσει κάποιος. Το μέτρο αυτό αναπαριστά το πλήθος των συνδέσεων που έχει κάποιος, δηλαδή όσο πιο κεντρικό θα είναι το άτομο τόσο περισσότερες συνδέσεις θα έχει. Ο κόμβος, ο οποίος έχει τις περισσότερες συνδέσεις σε σύγκριση με τους υπόλοιπους στο δίκτυο, θεωρείται ότι είναι σε πλεονεκτική θέση να μάθει μια πληροφορία που κινείται στο δίκτυο (για παράδειγμα μια φήμη). Η κεντρικότητα βαθμού όμως δεν είναι πάντα αντιπροσωπευτική, αφού το μόνο που προσφέρει είναι το πλήθος των συνδέσεων του κόμβου με το υπόλοιπο δίκτυο. Για τον υπολογισμό της κεντρικότητας βαθμού, υπολογίζουμε το σύνολο των γειτόνων του κόμβου και το διαιρούμε με το συνολικό πλήθος των κόμβων, για να έχουμε μια πιο εύχρηστη κλιμάκωση των τιμών.

$$DC(u) = \frac{d(u)}{n-1} \tag{3.7}$$

όπου το $d(u) = |N(u)|$ συμβολίζει το πλήθος των γειτόνων του κόμβου u. Εάν ο γράφος που μελετάμε είναι κατευθυνόμενος, η κεντρικότητα βαθμού μπορεί να μελετηθεί με δύο διαφορετικούς τύπους. Αρχικά, μπορούμε να μελετήσουμε τις εισερχόμενες ακμές, όπου κεντρικός θεωρείται ένας κόμβος ο οποίος μπορεί να πάρει πληροφορία από πολλούς κόμβους (In-Degree). Αντιθέτως, μπορούμε να χαρακτηρίσουμε ως κεντρικό έναν κόμβο ο οποίος έχει τη δυνατότητα να αποστείλει πληροφορία σε πολλούς κόμβους (Out-Degree). Η κεντρικότητα βαθμού μπορεί να υπολογιστεί για να αξιολογήσει τη κεντρικότητα ενός κόμβου βάσει μόνο των εισερχόμενων ή των εξερχόμενων ακμών ως

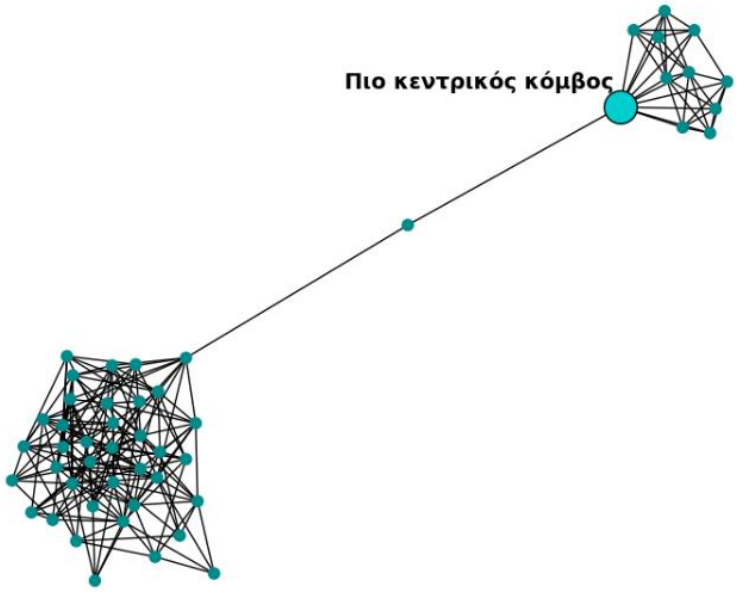
$$DC_{in}(u) = \frac{d^{-}(u)}{n-1} \tag{3.8}$$

$$DC_{out}(u) = \frac{d^{+}(u)}{n-1} \tag{3.9}$$

όπου $d^{-}(u) = |I(u)|$ συμβολίζει το πλήθος γειτόνων με εισερχόμενες ακμές και $d^{+}(u) = |O(u)|$ να συμβολίζει το πλήθος των γειτόνων με βάση των εξερχόμενων ακμών τους.

Βέβαια, ο κόμβος που έχει το μεγαλύτερο πλήθος συνδέσεων δεν είναι πάντα ο πιο κεντρικός. Για παράδειγμα αν οι περισσότερες συνδέσεις που έχει, είναι με κόμβους χωρίς άλλες συνδέσεις ή υπάρχουν στο δίκτυο κόμβοι που βρίσκονται πολύ μακριά από τον κόμβο αυτόν, τότε η

πληροφορία δεν θα διαδοθεί σε όλο το δίκτυο (βλέπε Σχήμα 3.6). Για το λόγο αυτό, η χρήση της κεντρικότητας βαθμού δεν είναι πάντα κατάλληλη για την ανάλυση σημαντικότητας των κόμβων. Ο χρόνος που απαιτείται για να υπολογιστεί την κεντρικότητα βαθμού όλων των κόμβων είναι τάξης $O(n)$ και είναι η μικρότερη σε απαίτηση πόρων από τις υπόλοιπες.



Σχήμα 3.6. Δίκτυο με μια ομάδα ατόμων απομακρυσμένη από το υπόλοιπο δίκτυο.

- **Κεντρικότητα εγγύτητας**

Ένα άλλο μέτρο για τη κεντρικότητα του δικτύου είναι η κεντρικότητα εγγύτητας (Closeness centrality). Η κεντρικότητα εγγύτητας θεωρεί πιο κεντρικό τον κόμβο από το οποίο έχουμε την ελάχιστη απόσταση από όλους τους υπολοίπους κόμβους. Για παράδειγμα, αν θέλουμε να διαδώσουμε μια πληροφορία σε ένα δίκτυο και τη δώσουμε στο κόμβο που έχει την ελάχιστη απόσταση από όλους τους κόμβους, τότε θα γνωρίζουν όλοι οι κόμβοι τη πληροφορία στο μικρότερο δυνατό χρόνο. Ο υπολογισμός της κεντρικότητας εγγύτητας έτσι όπως την έθεσε ο Beauchamp (1965) δίνεται από τον παρακάτω τύπο

$$Cl(u) = \frac{N-1}{\sum_{u=1}^{u=n} dist(u,v)} \tag{3.10}$$

όπου το $dist(u,v)$ συμβολίζει την απόσταση (μέγεθος μικρότερου μονοπατιού) μεταξύ των κόμβων u και v, η οποία υπολογίζεται βάσει του αλγορίθμου του Dijkstra (Javaid, 2013). Το n συμβολίζει τον συνολικό αριθμό των κόμβων του υπό-μελέτη δικτύου. Η κεντρικότητα εγγύτητας

μπορεί επίσης να θεωρηθεί ως η ικανότητα κάθε κόμβου να μεταδίδει πληροφορίες στους υπόλοιπους κόμβους του δικτύου. Η ύπαρξη μικρής μέσης απόστασης ενός κόμβου από τους υπόλοιπους, συνεπάγεται αυτόματα και καλύτερη ικανότητα στη διάδοση πληροφορίας. Είναι επομένως φυσικό επακόλουθο να αποδίδουμε υψηλή κεντρικότητα σε έναν τέτοιο κόμβο. Ο χρόνος που απαιτεί για να υπολογιστεί η κεντρικότητα είναι τάξεως $O(n^2)$.

- **Αρμονική κεντρικότητα**

Η αρμονική κεντρικότητα (Harmonic Centrality) είναι αρκετά παρόμοια με τη κεντρικότητα εγγύτητας και η υπολογιστική ισχύς που απαιτεί είναι ίση με αυτή της κεντρικότητας εγγύτητας Boldi and Sebastiano (2014). Όπως η προηγούμενη έτσι και αυτή, χρησιμοποιεί τις αποστάσεις μεταξύ κόμβων για να καθορίσει την σημαντικότητα τους. Η κεντρικότητα αυτή είναι λιγότερο ευαίσθητη σε απομακρυσμένες ή εκτός δικτύου παρατηρήσεις (outliers) αφού δεν ¨τιμωρεί¨ σε τόσο μεγάλο βαθμό τους κόμβους που έχουν μερικά πολύ μεγάλα μονοπάτια. Ο τύπος της κανονικοποιημένης αρμονικής κεντρικότητας είναι

$$HC(u) = \frac{1}{n-1}\sum_{u \neq v}\frac{1}{dist(u,v)} \tag{3.11}$$

όπου το $dist(u,v)$ συμβολίζει την απόσταση μεταξύ των κόμβων u και v από τον αλγόριθμο του Dijkstra. Η κεντρικότητα αυτή όπως και η κεντρικότητα εγγύτητας είναι οι καλύτερες όταν θέλουμε να μάθουμε ποιο άτομο μπορεί να διαδώσει μια πληροφορία στο μεγαλύτερο μέρος του δικτύου στο μικρότερο χρονικό διάστημα.

- **Κεντρικότητα ενδιαμεσότητας**

Η κεντρικότητα εγγύτητας θεωρεί πιο κεντρικό τον κόμβο, ο οποίος είναι πιο κοντά στους υπόλοιπους κόμβους αλλά δεν μας παρέχει πληροφορίες για την επιρροή που ασκεί στο δίκτυο γενικότερα. Με σκοπό να εντοπίσει τον κόμβο που έχει τη μεγαλύτερη επιρροή στο δίκτυο ο Freeman (1977) εισήγαγε ένα νέο μέτρο κεντρικότητας, αυτό της ενδιαμεσότητας (Betweenness Centrality). Με τη κεντρικότητα ενδιαμεσότητας προσπαθούμε ουσιαστικά να εντοπίσουμε το κόμβο που θα μας διευκολύνει στην επικοινωνία με τους υπόλοιπους κόμβους αλλά και στη ροή της πληροφορίας. Για να το καταφέρει αυτό, υπολογίζει τον αριθμό των φορών που ένας κόμβος ανήκει στη συντομότερη διαδρομή μεταξύ των υπολοίπων κόμβων και το διαιρεί με το συνολικό

αριθμό συντομότερων μονοπατιών μεταξύ των κόμβων του δικτύου. Ο τύπος κεντρικότητας ενδιαμεσότητας είναι

$$CB(u) = \sum_{s,t \epsilon V} \frac{\sigma(s,t|u)}{\sigma(s,t)} \tag{3.12}$$

όπου με $\sigma(s,t)$ συμβολίζεται ο αριθμός των συντομότερων μονοπατιών μεταξύ των κόμβων s και t. Με $\sigma(s,t|u)$ συμβολίζουμε το πλήθος των συντομότερων μονοπατιών μεταξύ των κόμβων s και t, όπου το μονοπάτι εμπεριέχει τον κόμβο u. Για $s = t$, έχουμε $\sigma(s,t) = 1$. Για τη κανονικοποίηση του μέτρου της κεντρικότητας ενδιαμεσότητας, τη διαιρούμε με $\frac{(n-1)(n-2)}{2}$ σε περίπτωση μη κατευθυνόμενου δικτύου. Αντίθετα, εάν το δίκτυο είναι κατευθυνόμενο τότε τη διαιρούμε με $(n-1)(n-2)$. Έτσι ο τύπος με τον οποίο θα υπολογίσουμε τη κετρικότητα ενδιαμεσότητας είναι

$$CB(u) = \frac{\alpha \sum_{s,t \epsilon V} \frac{\sigma(s,t|u)}{\sigma(s,t)}}{(n-1)(n-2)} \tag{3.13}$$

όπου $\alpha = 1$ όταν το δίκτυο είναι κατευθυνόμενο και $\alpha = 2$ σε αντίθετη περίπτωση. Ο χρόνος που απαιτεί για να υπολογιστεί η κεντρικότητα είναι της τάξεως $O(n^3)$.

- **Ιδιοδιανυσματική κεντρικότητα**

Η ιδιοδιανυσματική κεντρικότητα (Eigenvector centrality) ακολουθεί περίπου την ίδια λογική με αυτή της κεντρικότητας βαθμού. Όπως προαναφέρθηκε, η κεντρικότητα βαθμού το μόνο που λαμβάνει υπόψη είναι το πλήθος των γειτόνων που έχει ένας κόμβος, όμως κάθε κόμβος δεν είναι εξίσου σημαντικός. Για το λόγο αυτό, δημιουργήθηκε η ιδιοδιανυσματική κεντρικότητα η οποίο προσδιορίζει ένα βαθμό σημαντικότητας σε κάθε γειτονικό κόμβο, ανάλογα με το πλήθος των γειτόνων που έχει ο ίδιος. Κατά αυτόν τον τρόπο, έχουμε ένα μέτρο κεντρικότητας το οποίο συνυπολογίζει το πλήθος των γειτόνων αλλά και τη σημαντικότητα τους. Αν $\boldsymbol{A} = [a_{ij}]$ συμβολίζει τον πίνακα γειτνίασης ενός γράφου $G = (V, E)$ ($a_{ij} = 1$ αν οι κόμβοι i και j είναι γειτονικοί, ενώ σε αντίθετη περίπτωση, $a_{ij} = 0$) τότε η ιδιοδιανυσματική κεντρικότητα των κόμβων i (CE_i) υπολογίζεται ως εξής

$$\boldsymbol{CE'} = \frac{1}{\lambda_1}\boldsymbol{A} \cdot \boldsymbol{CE} \tag{3.14}$$

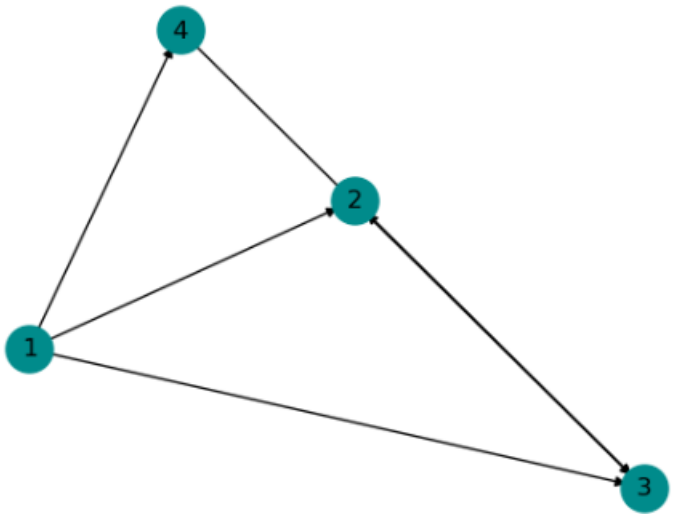
όπου το λ_1είναι η πρώτη ιδιοτιμή του πίνακα γειτνίασης A και $\mathbf{CE} = [CE_1, \dots CE_n]^T$. Για να μπορέσουμε να επιτύχουμε όσο καλύτερα αποτελέσματα, κάνουμε χρήση της επαναληπτικής διαδικασίας, δηλαδή

$$\boldsymbol{CE}^{\kappa+1} = \frac{1}{\lambda_1} A \cdot \boldsymbol{CE}^{\kappa} \tag{3.15}$$

Η διαδικασία αυτή επαναλαμβάνεται μέχρι η κεντρικότητα να συγκλίνει, δηλαδή έως ότου $\boldsymbol{CE}^{\kappa+1} \approx \boldsymbol{CE}^{\kappa}$.

Κόμβοι με υψηλή ιδιοδιανυσματική κεντρικότητα είναι κόμβοι οι οποίοι έχουν την ικανότητα να ασκήσουν άμεση επιρροή στο δίκτυο (έχουν πολλούς γείτονες) ή μπορούν να ασκήσουν επιρροή σε κόμβους που έχουν πολλούς γείτονες. Για το λόγο αυτό η κεντρικότητα αυτή μπορεί να χρησιμοποιηθεί σε διάφορα δίκτυα και αναλύσεις. Η ιδιοδιανυσματική κεντρικότητα μπορεί να χρησιμοποιηθεί για κατευθυνόμενους και μη γράφους. Βέβαια, στη περίπτωση των κατευθυνόμενων γράφων, επειδή ο πίνακας δεν είναι συμμετρικός μπορεί να δημιουργηθούν προβλήματα και κάποιος κόμβος που έχει αλληλεπιδράσεις μπορεί να φανεί πως δεν είναι σημαντικός.

Ας υποθέσουμε ότι μελετάμε ένα δίκτυο με ακολούθους, όπου οι ακμές συμβολίζουν ότι ένας κόμβος ακολουθεί κάποιον άλλον.




```
Ιδιοδιανυσματική κεντρικότητα
      Κόμβος 1: 0.0
      Κόμβος 2: 0.71
      Κόμβος 3: 0.71
      Κόμβος 4: 0.0
```

Σχήμα 3.7. Μη αποσυνδεμένο δίκτυο με κόμβους με μηδενική κεντρικότητα.

Στο παραπάνω παράδειγμα βλέπουμε ότι δύο κόμβοι έχουν μηδενική ιδιοδιανυσματική κεντρικότητα παρόλο που δεν είναι πλήρως αποσυνδεμένα από το υπόλοιπο δίκτυο. Για παράδειγμα, το άτομο 1 ακολουθεί όλους τους κόμβους όμως κανένα άτομο δεν τον ακολουθεί. Σε αυτή τη περίπτωση βλέπουμε ότι ο κόμβος έχει μηδενική κεντρικότητα το οποίο δεν θα έπρεπε να συμβαίνει. Ένα άλλο πρόβλημα που παρατηρούμε να υπάρχει είναι το μέτρο κεντρικότητας του τέταρτου ατόμου. Το άτομο αυτό ακολουθεί το άτομο 3 και ακολουθείτε από το άτομο 1. Βέβαια, παρατηρούμε ότι η κεντρικότητα του είναι μηδενική. Αυτό συμβαίνει διότι, το βάρος του πρώτου ατόμου είναι μηδενικό, έτσι ο αλγόριθμος θεωρεί πως δεν υπάρχει κόμβος που να συνδέετε μαζί του και θεωρεί πως η κεντρικότητα του πρέπει να είναι ίση με το μηδέν. Αν γενικεύσουμε το παραπάνω παράδειγμα, μπορούμε να καταλήξουμε σε κόμβους με μηδενική κεντρικότητα, που ακολουθούνται από κόμβους, όπου και αυτοί έχουν μηδενική κεντρικότητα. Βέβαια, τέτοιες περιπτώσεις σπανίζουν και θεωρείτε μια από τις καλύτερες επιλογές για τη μελέτη κοινωνικών δικτύων.

- **Κεντρικότητα Katz**

Όπως αναφέραμε προηγουμένως, μερικές φορές η ιδιοδιανυσματική κεντρικότητα μπορεί να έχει κάποια προβλήματα. Για να λύσει τα προβλήματα αυτά ο Katz (1953) πρότεινε κάποια μετατροπή στον τύπο της ιδιοδιανυσματικής κεντρικότητας με κύριο χαρακτηριστικό τη προσθήκη μια σταθερής τιμής σε όλες τις κεντρικότητες των κόμβων. Ο τύπος του Katz είναι

$$\boldsymbol{CK'} = a\boldsymbol{A} \cdot \boldsymbol{CK} + b\mathbf{1} \qquad (3.16)$$

όπου το a και το b είναι θετικές σταθερές και το $\mathbf{1}$ συμβολίζει το διάνυσμα (1,1…,1). Το b το προσθέτουμε με σκοπό να μην υπάρχουν κόμβοι με μηδενική κεντρικότητα, ενώ το πρώτο μέρος είναι παρόμοιο με αυτό της ιδιοδιανυσματικής. Η προσθήκη μιας σταθεράς στη κεντρικότητα κάθε κόμβου, δεν επηρεάζει τις εναλλαγές στις κεντρικότητες αφού όλοι οι κόμβοι θα έχουν ένα επιπρόσθετο μέτρο κεντρικότητας. Το διάνυσμα $\boldsymbol{CK}$ μπορούμε να το γράψουμε στη μορφή $\boldsymbol{CK} = b(\boldsymbol{I} - a\boldsymbol{A})^{-1} \cdot 1$ όπου με $\boldsymbol{I}$ συμβολίζουμε το ταυτοτικό πίνακα, όπου η κύρια διαγώνιος είναι ίση με τη μονάδα και όλα τα υπόλοιπα στοιχεία του είναι μηδενικά. Στα μέτρα κεντρικότητας δεν μας ενδιαφέρει η τιμή που έχουμε σε κάθε στοιχείο αλλά μας ενδιαφέρει ποιοι κόμβοι έχουν τη μεγαλύτερη ή τη μικρότερη τιμή, οπότε η τιμή που θα έχει η σταθερά b δεν μας ενδιαφέρει και

μπορούμε να την καθορίσουμε με όποια τιμή επιλέξουμε. Χάριν ευκολίας, μπορούμε να θέσουμε το b ίσο με τη μονάδα. Με αυτό τον τρόπο θα έχουμε το τύπο

$$\boldsymbol{CK} = (\boldsymbol{I} - a\boldsymbol{A})^{-1} \cdot 1 \tag{3.17}$$

Έτσι για να μπορέσουμε να υπολογίσουμε τη κεντρικότητα των κόμβων μένει να υπολογίσουμε τη σταθερά a. Για να καθορίσουμε τη σταθερά πρέπει να κατανοήσουμε ότι, δεν μπορούμε να βάλουμε μια οποιαδίποτε αυθαίρετη τιμή διότι μπορεί να δημιουργηθούν διάφορα προβλήματα. Αν για παράδειγμα $a \to 0$ τότε το μόνο που θα παραμείνει στη σχέση (3.16) είναι η σταθερά b και όλα οι κόμβοι θα έχουν ίση κεντρικότητα. Στη αντίθετη περίπτωση, όπου η σταθερά b είναι πολύ μεγάλή, η σχέση (3.17) θα αποκλίνει. Αυτό συμβαίνει όταν η ορίζουσα $\det(\boldsymbol{I} - a\boldsymbol{A})$ θα ξεπεράσει τη τιμή μηδέν, το οποίο συμβαίνει πρώτη φορά όταν $a = \frac{1}{\lambda_1}$ (με λ_1συμβολίζεται η μεγαλύτερη ιδιοτιμή του πίνακα γειτνίασης). Για τον λόγο αυτόν θα πρέπει να επιλέξουμε ένα στοιχείο μικρότερο από τη τιμή αυτή (δηλαδή $0 < a < \frac{1}{\lambda_1}$). Η κεντρικότητα Katz δημιουργήθηκε με σκοπό να αντιμετωπίσει κάποια προβλήματα που μπορεί να μας δημιουργούσε η ιδιοδιανυσματική, σε κατευθυνόμενα δίκτυα. Βέβαια, η χρήση της σε μη κατευθυνόμενους γράφους δεν είναι πάντοτε αρνητική. Για παράδειγμα μπορεί να θέλουμε να δώσουμε μεγαλύτερη βαρύτητα σε κόμβους που κάποιο χαρακτηριστικό τους το θεωρούμε πιο σημαντικό. Για τέτοιου είδους αναλύσεις, θα χρησιμοποιήσουμε το τύπο

$$\boldsymbol{CK_i} = a\sum_j \boldsymbol{A_{ij}CK_j} + \boldsymbol{b_i} \tag{3.18}$$

όπου με $\boldsymbol{b_i}$ θα συμβολίζεται η προκαθορισμένη αύξηση που θα έχει ο κάθε κόμβος. Αν για παράδειγμα υπολογίσουμε τη κεντρικότητα αυτή, στα δεδομένα του προηγούμενου παραδείγματος (βλέπε Σχήμα 3.7) θα δούμε ότι έχουμε αρκετά πιο ικανοποιητικά αποτελέσματα.

```
Κεντρικότητα Katz
Κόμβος 1: 0.43
Κόμβος 2: 0.53
Κόμβος 3: 0.57
Κόμβος 4: 0.47
```

Πίνακας 3.1. Αποτελέσματα κεντρικότητας Katz.

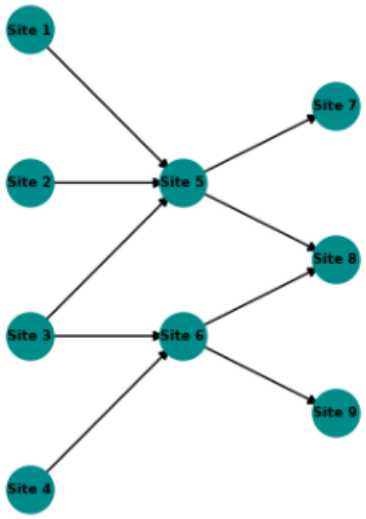
Βλέπουμε πως πλέον, οι κόμβοι 1 και 4 δεν έχουν μηδενική κεντρικότητα όπως στην ιδιοδιανυσματική κεντρικότητα. Τα αποτελέσματα αυτά φαίνονται πιο αντικειμενικά, αφού κανένας κόμβος δεν ήταν αποσυνδεμένος από το υπόλοιπο δίκτυο.

- **PageRank Κεντρικότητα**

Ο αλγόριθμος PageRank δημιουργήθηκε από τους Page et al. (1999), από όπου πήρε και το όνομα της. Ο αλγόριθμος σχεδιάστηκε για να αξιολογεί τη σημαντικότητα ιστοσελίδων και ήταν ο πρώτος αλγόριθμος που χρησιμοποιήθηκε από τη Google για τη μέτρηση της σημαντικότητας των ιστοσελίδων. Ο PageRank βασίστηκε στην ιδέα ενός τυχαίου περίπατου στους γράφους. Σε ένα τυχαίο περίπατο, υπάρχει ένα άτομο το οποίο πλοηγείται τυχαία σε σελίδες του διαδικτύου και ονομάζεται τυχαίος σέρφερ. Ουσιαστικά, οι κόμβοι είναι οι διάφορες ιστοσελίδες που μπορεί να επισκεφτεί, οι οποίες έχουν συνδέσμους που τον μεταφέρουν σε μια άλλη σελίδα και οι σύνδεσμοι αυτοί αποτελούν τις ακμές του δικτύου. Ο σέρφερ επιλέγει τυχαία την ιστοσελίδα στην οποία θα ξεκινήσει και έπειτα επιλέγει τυχαία τους υπερσυνδέσμους που θα ακολουθήσει. Σε περίπτωση όπου κάποια σελίδα που επισκέπτεται, δεν έχει υπερσυνδέσμους που θα τον προωθήσουν σε μια άλλη ιστοσελίδα, τότε θα επισκεφτεί άλλη τυχαία. Ακολουθώντας τους υπερσυνδέσμους υπάρχει μια μικρή πιθανότητα ο σέρφερ να μπει σε ένα loop (θα επισκέπτεται συνεχώς τις ίδιες ιστοσελίδες), όπου σε αυτή τη περίπτωση θεωρούμε πως δεν θα συνεχίσει, αλλά αντιθέτως θα επιλέξει τυχαία κάποια άλλη σελίδα. Αρχικά, παρουσιάζεται ένας απλός τύπος για το PageRank

$$\boldsymbol{R(u)} = c \sum_{v \epsilon \boldsymbol{B(u)}} \frac{\boldsymbol{R(v)}}{N_v} \qquad (3.19)$$

όπου με u συμβολίζουμε την ιστοσελίδα (κόμβο) στην οποία βρισκόμαστε αυτή τη στιγμή. Με $\boldsymbol{F_u}$ συμβολίζουμε το σύνολο των ιστοσελίδων στο οποίο μπορούμε να μεταφερθούμε από την ιστοσελίδα u και με $N_u = |\boldsymbol{F_u}|$. Αντίθετα, με $\boldsymbol{B_u}$ συμβολίζουμε το σύνολο των ιστοσελίδων, από τους οποίους μπορούμε να μεταφερθούμε στη u. Για παράδειγμα, στο Σχήμα 3.8 για τον κόμβο $u = 5$ το σύνολο $\boldsymbol{B_u}$ θα αποτελείτε από τις πρώτες τρεις ιστοσελίδες ($Site\ 1, Site\ 2, Site\ 3$) ενώ το σύνολο $\boldsymbol{F_u}$ από τις ιστοσελίδες $Site\ 7, Site\ 8$ και $Site\ 9$. Το $c < 1$ συμβολίζει μια σταθερά με την οποία μπορούμε να κανονικοποιήσουμε τα αποτελέσματα του PageRank.



Σχήμα 3.8. Σχηματική αναπαράσταση ενός υποσυνόλου 9 ιστοσελίδων.

Βασισμένος στον τύπο (3.19), ο αλγόριθμος ακολουθεί τα επόμενα βήματα

1 Αναθέτει σε όλους τους κόμβους του δικτύου μια τιμή κεντρικότητας αυτή η τιμή μπορεί να είναι $\frac{1}{N}$ όπου με N συμβολίζεται το πλήθος των κόμβων.

2 Αναθέτει μια τιμή στο στοιχείο c, όπου συνήθως είναι ίση με 0.85.

3 Για κάθε κόμβο του δικτύου επαναυπολογίζει το μέτρο κεντρικότητας $\boldsymbol{R'(u)}$ του κόμβου u. Εδώ πρέπει να σημειώσουμε πως το συνολικό άθροισμα του βαθμού κεντρικότητας των στοιχείων ισούται με τη μονάδα σε κάθε βήμα. Το μέτρο κεντρικότητας υπολογίζεται από τον τύπο

$$\boldsymbol{R'(u)} = c \sum_{v \epsilon \boldsymbol{B(u)}} \frac{\boldsymbol{R'(v)}}{N_v} \tag{3.20}$$

4 Σε αυτό το βήμα ο αλγόριθμος σταματά σε περίπτωση που τα μέτρα κεντρικότητας κάθε κόμβου, έχουν πολύ μικρή απόκλιση από το προηγούμενο βήμα ή έχουμε ξεπεράσει μια προκαθορισμένη τιμή επαναλήψεων, σε αντίθετη περίπτωση πηγαίνει ξανά στο προηγούμενο βήμα.

Χρησιμοποιώντας τον τύπο (3.19), μπορούμε να αντιμετωπίσουμε κάποια προβλήματα. Υποθέτουμε ότι υπάρχουν δύο κόμβοι σε ένα γράφο, μεταξύ των οποίων υπάρχει κατεύθυνση, προκειμένου ο ένας να δείχνει στον άλλο. Ωστόσο, αυτοί οι δύο κόμβοι δεν έχουν συνδέσεις προς άλλους κόμβους στον γράφο. Επιπλέον, υπάρχει κόμβος στο δίκτυο μας που δείχνει τουλάχιστον

έναν από τους δύο αυτούς. Σε αυτή τη περίπτωση κατά την διάρκεια επαναλήψεων του αλγορίθμου, οι δύο αυτοί κόμβοι θα πάρουν αρκετά μεγάλο βαθμό κεντρικότητας, το οποίο φυσικά και δεν πρέπει να έχουν, το γεγονός αυτό το ονομάζουμε βύθιση βαθμού. Για να αντιμετωπίσουν το πρόβλημα αυτό, οι δημιουργοί χρησιμοποίησαν μια ακόμη μεταβλητή διανύσματος στη συνάρτηση. Παρουσίασαν το $\boldsymbol{E}(\boldsymbol{u})$, το οποίο το χρησιμοποίησαν ως πηγή για να δίνει έξτρα κεντρικότητα και να αντιμετωπίζει προβλήματα όπως η βύθιση βαθμού ή κύκλους κόμβων που δεν έχουν υπερσυνδέσμους εκτός του κύκλου. Αυτό το κάνουν ώστε ο περίπατος του τεχνητού σέρφερ να είναι πιο ρεαλιστική. Εάν ένα άτομο στη πραγματικότητα έμπαινε σε ένα loop (βλέπε Σχήμα 3.9) κατά τη διάρκεια που περιηγούταν στο διαδίκτυο, θα έβγαινε από αυτό, αφού δεν έχει νόημα να δει το ίδιο περιεχόμενο, πολλές φορές.

Σχήμα 3.9. Αναπαράσταση ενός loop.

Έτσι πλέον, χρησιμοποιείται ο τύπος

$$\boldsymbol{R}'(\boldsymbol{u}) = c \sum_{v \in \boldsymbol{B}(\boldsymbol{u})} \frac{\boldsymbol{R}'(\boldsymbol{v})}{N_v} + c\boldsymbol{E}(\boldsymbol{u}). \tag{3.21}$$

Ο αλγόριθμος είναι φτιαγμένος για να εφαρμόζεται σε κατευθυνόμενους γράφους, βέβαια μπορεί να χρησιμοποιηθεί και σε μη κατευθυνόμενους. Σε μια τέτοια περίπτωση η σύνδεση που φαίνεται να υπάρχει μεταξύ δύο κόμβων την αναπαριστά ως σύνδεση, όπου οι δύο κόμβοι "δείχνουν" ο ένας τον άλλον. Προβλήματα μπορούν να μας δημιουργήσουν οι σύνδεσμοι που έχουν ακμές, μόνο προς συνδέσμους χωρίς εξερχόμενες ακμές. Τους συνδέσμους αυτούς του ονομάζουμε dangling links και είναι κόμβοι που δεν επηρεάζουν τον βαθμό κάποιου άλλου κόμβου άμεσα. Για τον λόγο αυτό, τους αφαιρούμε έως ότου υπολογίσουμε το PageRank των υπολοίπων κόμβων και έπειτα τους επανατοποθετούμε στο δίκτυο μας. Με αυτό τον τρόπο θα επηρεαστεί ελαφρώς ο βαθμός των υπολοίπων κόμβων, αλλά η διαφορά θα είναι αμελητέα. Ο αλγόριθμος PageRank αναγνωρίζει τη σημασία των κόμβων που συνδέονται με κεντρικούς κόμβους, χωρίς ωστόσο να δίνει υπερβολική έμφαση σε τέτοιες συνδέσεις. Ένα απλό παράδειγμα για κατανόηση

αυτής της έννοιας είναι μια σχολική τάξη. Όλα τα παιδιά σε μια τάξη επικοινωνούν με τον δάσκαλό τους, αλλά ο δάσκαλος, λόγω περιορισμένου χρόνου, δεν μπορεί να αλληλεπιδράσει με κάθε μαθητή ισάξια. Συνεπώς, η επικοινωνία ενός παιδιού με τον δάσκαλο δεν οδηγεί αυτόματα σε υψηλότερη τιμή κεντρικότητας, αντίθετα με ένα παιδί που έχει σταθερή σχέση με άλλους μαθητές, αλλά όχι επικοινωνία με τον δάσκαλο. Αυτό το παράδειγμα επισημαίνει ότι η κεντρικότητα PageRank δημιουργήθηκε αρχικά για την αξιολόγηση ιστοσελίδων, αλλά αποτελεί χρήσιμο εργαλείο για την ανάλυση κοινωνικών δικτύων.

ΚΕΦΑΛΑΙΟ 4

Απαρίθμηση τριγώνων

4.1 Εισαγωγή

Η κατανόηση των αλληλεπιδράσεων και των συνδέσεων μεταξύ των στοιχείων ενός δικτύου είναι πολύ βασικό εργαλείο για την ανάλυση τους. Η μελέτη δομικών προτύπων αποκτά κεντρική σημασία, αποκαλύπτοντας βασικά χαρακτηριστικά που διαμορφώνουν τη συμπεριφορά και τη λειτουργικότητα των δικτύων. Ο υπολογισμός του αριθμού των τριγώνων σε ένα γράφο αποτελεί θεμελιώδες πρόβλημα για διάφορες εφαρμογές. Σε αυτό το κεφάλαιο θα αναλύσουμε διάφορους αλγορίθμους για τον υπολογισμό των τριγώνων ενός γράφου όπως επίσης κάποια βασικά χαρακτηριστικά τα οποία μπορούμε να αξιοποιήσουμε για την κατανόηση της δομής ενός κοινωνικού δικτύου. Σε κοινωνικούς γράφους και κυρίως σε δίκτυα φίλων, αναμένουμε ο αριθμός των τριγώνων να είναι αρκετά μεγαλύτερος από άλλους είδους δίκτυα. Ο λόγος είναι ότι, αν έχουμε δύο φίλους Α και Β, όπου ο Α είναι επίσης φίλος με τον C, θα αναμένουμε ότι η πιθανότητα ο C να είναι (ή να γίνει) φίλος με τον Β είναι αρκετά μεγαλύτερες. Έτσι η καταμέτρηση και καταγραφή τριγώνων μας βοηθάει να αντιληφθούμε τη σχέση μεταξύ των κόμβων εντός του δικτύου.

4.2 Αλγόριθμοι καταμέτρησης και καταχώρησης τριγώνων

Δεδομένης της σημαντικότητας της καταμέτρησης τριγώνων, έχουν δημιουργηθεί ποικίλοι αλγόριθμοι για την καταμέτρηση τους. Σε αυτή την ενότητα θα αναλυθούν διάφοροι από τους αλγορίθμους που έχουν αναπτυχθεί για την καταμέτρηση τριγώνων. Οι αλγόριθμοι χωρίζονται σε δύο κατηγορίες. Στη πρώτη κατηγορία ανήκουν αλγόριθμοι οι οποίοι, καταμετρούν τα τρίγωνα, ενώ στη δεύτερη αλγόριθμοι που καταμετρούν αλλά και αποθηκεύουν σε λίστες (καταχωρούν) του κόμβους που δημιουργούν το κάθε τρίγωνο. Προτού προχωρήσουμε στην ανάλυση τον αλγορίθμων θα δώσουμε τον ορισμό ενός τριγώνου.

Έστω ότι έχουμε ένα μη-κατευθυνόμενο γράφο $G = (V, E)$. Ένα τρίγωνο Δ_{uvw} (ή Δ) του γράφου G ορίζεται ως ένας πλήρης υπό-γράφος τριών κόμβων u, v και w όπου $V(\Delta_{uvw}) = \{u, v, w\}$ και $E(\Delta_{uvw}) = \{\{u, v\}, \{u, w\}, \{v, w\}\}$. Έτσι το πλήθος των τριγώνων, στους οποίους ανήκει ένας κόμβος u είναι

$$\Delta(u) = |E(G[N(u)])| \tag{4.1}$$

όπου με $N(u)$ συμβολίζουμε το σύνολο των γειτόνων του κόμβου u. Βάσει του παραπάνω τύπου, μπορούμε να υπολογίσουμε το συνολικό πλήθος των τριγώνων του δικτύου. Αν αθροίσουμε όλα τα σύνολα τριγώνων όλων των κόμβων, θα έχουμε τρεις φορές το συνολικό πλήθος τριγώνων, αφού κάθε τρίγωνο θα καταμετρηθεί τρείς φορές (μια για κάθε κόμβο στον οποίο ανήκει). Έτσι το συνολικό πλήθος τριγώνων υπολογίζεται ως εξής

$$\Delta(G) = \frac{1}{3}\sum_{u \epsilon V} \Delta(u) \tag{4.2}$$

Παρακάτω θα παρουσιαστούν επτά βασικοί αλγόριθμοι εύρεσης τριγώνων. Κάθε αλγόριθμος εφαρμόζει διαφορετικές μεθοδολογίες και τεχνικές, οι οποίες επιδρούν άμεσα στην αποδοτικότητά και τη πολυπλοκότητά τους. Η κατανόηση αυτών των αλγορίθμων είναι απαραίτητη για τη μελέτη δομών και τη βελτιστοποίηση της ανίχνευσης τριγώνων σε ποικίλα σύνολα δεδομένων.

α. Αλγόριθμος try-all

Ο αλγόριθμος try-all (γνωστός και ως Brute Force) χρησιμοποιείται για τον υπολογισμό του πλήθους των τριγώνων που ανήκουν σε έναν γράφο. Ενώ είναι μια μέθοδος αρκετά απλή και κατανοητή από τον καθένα δεν είναι πρακτική, αφού απαιτεί πολύ χρόνο για την εκτέλεσή της. Ο συνολικός χρόνος που απαιτεί ο αλγόριθμος για την εκτέλεσή του είναι $\Theta(n^3)$ (ή ο μέγιστος χρόνος ισούται με $O(n^3)$) και αποτελεί τον πιο αργό αλγόριθμο για τον υπολογισμό τριγώνων που έχουμε διαθέσιμο. Ο αλγόριθμος επισκέπτεται όλους τις κόμβους ξεχωριστά και εντοπίζει όλες τις πιθανές δυάδες γειτονικών κόμβων. Αν υπάρχουν ακμές μεταξύ της τριάδας αυτής τότε ο αλγόριθμος αναγνωρίζει πως η τριάδα σχηματίζει τρίγωνο και ο αριθμός τριγώνων που ανήκει ο κόμβος αυτός αυξάνεται κατά μία μονάδα.

```
Function Try-All(Graph G):
1:  triangle_list= []                                   ▸Λίστα τριγώνων
2:  triangle_count= 0
3:  for u ∈ V do:
4:      for v ∈ N(u) do:
5:          for w ∈ N(v) do:
6:              if w! = u and w ∈ N(u):                 ▸Αν οι κόμβοι u, v και w γειτονικοί
7:                  triangle= {u, v, w}
8:                  triangle_list.append(triangle)      ▸Πρόσθεσε στη λίστα το τρίγωνο
9:                  triangle_count=triangle_count+1
10: return triangle_list, triangle_count/3              ▸Επειδή κάθε τρίγωνο υπολογίζεται τρεις φορές
```

Αλγόριθμος 4.1. Try-All

Ενώ ο αλγόριθμος Try-All είναι εννοιολογικά απλός και εξασφαλίζει ακριβή καταμέτρηση τριγώνων, λόγω της υπολογιστικής ισχύος που απαιτεί, θεωρείται πολύ δαπανηρός. Ως εκ τούτου, για πρακτικές εφαρμογές σε μεγάλα σύνολα δεδομένων (όπως είναι αυτά των κοινωνικών δικτύων), δεν θεωρείται πρακτικός και δεν συνηθίζεται να χρησιμοποιείται.

β. Αλγόριθμος Matrix Multiplication

Όπως προαναφέραμε, ο πίνακας γειτνίασης A ενός γράφου $G = (V, E)$ είναι ένας τρόπος για να αποθηκεύσουμε τις συνδέσεις μεταξύ των κόμβων του δικτύου. Βέβαια, είναι ένα εργαλείο που μπορούμε να χρησιμοποιήσουμε για να βρούμε διάφορες άλλες πληροφορίες για τον γράφο που μελετάμε.

Έστω A πίνακας γειτνίασης ενός γράφου $G = (V, E)$. Τότε το συνολικό πλήθος μονοπατιών μήκους k, για να μεταφερθούμε από τον κόμβο i στον κόμβο j, δίνεται από το στοιχείο (i, j) του πίνακα A^k.Ένα τρίγωνο μπορούμε να το εκφράσουμε ως ένα μονοπάτι τριών βημάτων, από έναν κόμβο στον εαυτό του. Οπότε, για να βρούμε το πλήθος τριγώνων στο οποίο ανήκει ένας κόμβος i, αρκεί να υπολογίσουμε το στοιχείο (i, i) του πίνακα A^3.

```
Function Matrix-Multiplication (Graph G)
1:  triangles_count= 0
2:  compute A^2
3:  for i = 1,2, … , n do
4:      triangles_count = triangles_count + A_ii · A_ii^2      ▸ Στοιχείο (i, i) του πίνακα A^3
5:  return triangle_count/3
```

Αλγόριθμος 4.2. Matrix-Multiplication

Για τον υπολογισμό του πίνακα A^3 ό συνολικός χρόνος που απαιτείται είναι $\Theta(n^3)$ (ή μέγιστος χρόνος $O(n^3)$). Βέβαια, κάνοντας χρήση ταχύτερων μεθόδων (Coppersmith and Winograd, 1987) για τον πολλαπλασιασμών πινάκων, ο αλγόριθμος μπορεί να τρέξει σε χρόνο $\Theta(n^\gamma)$, με $\gamma \leq 2.376$ (Condon and Karp, 2001) να συμβολίζει τη πολυπλοκότητα του πολλαπλασιασμού πινάκων.

γ. Αλγόριθμος Node-Iterator

Ο αλγόριθμος αυτός είναι αρκετά αποτελεσματικός και πολύ πιο γρήγορος από τον αλγόριθμος Try-All. Ο αλγόριθμος επαναλαμβάνεται για όλους τους κόμβους, όπου ελέγχει για κάθε ζεύγος γειτόνων του κόμβου u, αν είναι γείτονες μεταξύ τους.

```
Function Node-Iterator (Graph G)
1:  triangle_list= []
2:  triangle_count= 0
3:  for u ∈ V do:
4:      for {v, w} ∈ N(u) do:
5:          if (v, w) ∈ E(u) then:
6:              if u < v < w then:                ▸Κάθε τρίγωνο καταμετρείται μια φορά
7:                  triangle= {u, v, w}
8:                  triangle_list.append(triangle)
9:                  triangle_count=triangle_count+1
10: return triangle_list, triangle_count
```

Αλγόριθμος 4.3. Node-Iterator (Schank, 2007)

Ο ασυμπτωτικός χρόνος που απαιτείται για την εκτέλεση του αλγορίθμου, δίνεται από την έκφραση

$$\sum_{u\epsilon V} \binom{d(u)}{2} \tag{4.3}$$

το οποίο φράζεται από το $O(n \cdot d_{max}^2)$ (d_{max} είναι ο μέγιστος αριθμός γειτόνων του γράφου, βλέπε 2.5) και είναι αρκετά γρηγορότερο από το $O(n^3)$.

δ. Αλγόριθμοι ayz και Listing-ayz

Ο αλγόριθμος ayz συνδυάζει τεχνικές από τους αλγόριθμους node-iterator και matrix-multiplication (Alon, Yuster and Zwick, 1994). Ο αλγόριθμος χωρίζει το δίκτυο σε δύο σύνολα. Στο πρώτο σύνολο εντάσσονται κόμβοι με μικρό βαθμό (μικρό αριθμό γειτόνων) $V_l = \{u\epsilon V: d(u) \leq \beta\}$, ενώ οι κόμβοι με μεγάλο βαθμό εντάσσονται στο δεύτερο σύνολο $V_h = V \backslash V_l$, όπου $\beta = m^{(\gamma-1)/(\gamma+1)}$ και $\gamma \leq 2.376$ η παράμετρος πολυπλοκότητας πολλαπλασιασμού πινάκων. Ο αλγόριθμος έπειτα χρησιμοποιεί τη μέθοδο node-iterator για να υπολογίσει τον αριθμό των τριγώνων στο σύνολο τον μικρών σε βαθμό κόμβων και τη μέθοδο (γρήγορου) πολλαπλασιασμού πινάκων στο σύνολο V_h. Για τη μέθοδο Listing-ayz χρησιμοποιείται ο node-iterator και για τα δύο σύνολα.

Function ayz(Graph $G, \gamma\ parameter$)

```
1:  β = m^((γ−1)/(γ+1))
2:  for u ∈ V do:
3:      triangle_count(u)←0
4:      if d(u) < β then:
5:          V_l←V_l ∪ {u}
6:      else:
7:          V_h←V_h∪{u}
8:  for ueVlow do:
9:      for {v, w} ∈ N(u) do:
10:         if (v, w) ∈ E then:
11:             if v, w ∈ Vlow then:
12:                 triangle_count(u)←triangle_count(u) + 1/3
13:             else if u, v ∈ V_h then:
14:                 triangle_count(u)←triangle_count(u)+1
15:             else:
16:                 triangle_count(u)←triangle_count(u)+1/2
17: A<-adjacency Matrix of V_h
18: compute A^3
19: for u ∈ V_h do:
20:     triangle_count(u)←triangle_count(u) + A^3_{i,i}/2
21: triangle_count= sum(triangle_count(u), u ∈ G)/6
22: return triangle_count
```

Αλγόριθμος 4.4. Counting ayz (Schank, 2007)

Ο αλγόριθμος καταμέτρησης τριγώνων ayz χρειάζεται χρόνο τάξης $O(m^{2\gamma/(\gamma+1)})$ για τον οποίο οι συγγραφείς έχουν υπολογίσει πως έχει ανώτατο χρόνο εκτέλεσης $O(m^{1.41})$ (Alon, Yuster and Zwick, 1994) για τον υπολογισμό τριγώνων. Ο αλγόριθμος ο οποίος πέραν της καταμέτρησης μπορούμε να έχουμε και τα σύνολα των τριγώνων (listing-ayz) έχει ασυμπτωτικό χρόνο εκτέλεσης $O(m^{3/2})$, χρησιμοποιώντας τη μέθοδο node-iterator και για τα δύο σύνολα.

ε. Αλγόριθμος Edge-Iterator

Ο αλγόριθμος edge-iterator επαναλαμβάνεται για όλες τις ακμές και ελέγχει αν οι δύο προσκείμενες ακμές σχηματίζουν ένα τρίγωνο. Για κάθε ακμή $(v, w)\epsilon E$, οι κόμβοι $\{u, v, w\}$ σχηματίζουν τρίγωνα, μόνο στη περίπτωση όπου ο κόμβος u, είναι γειτονικός με τους $\{v, w\}$.

```
Function edge-iterator (Graph G)
triangle_list= []
triangle_count= 0
for u ∈ V do:
    for v ∈ N(u) do:
        j ← FirstNeighborIndex(u)
        k ← FirstNeighborIndex(v)
        while j < n and k < n do:
            if j < k then:
                j ← NextNeighborIndex(u, j)
            else if k < j then:
                K ← NextNeighborIndex(v, k)
            else:
                if i < k < l then:
                    triangle_list=triangle_list.append(triangle)
                    triangle_count+= 1
                j ← NextNeighborIndex(u, j)
                k ← NextNeighborIndex(v, k)
return triangle_list, triangle_count
Function FirstNeighborIndex(u)
minIndex ← n
for v ∈ N(u) do:
    if index(v)< minIndex then:
        minIndex ← index(v)
return minIndex
Function NextNeighborIndex(u, j)
minIndex <-n
for v ∈ N(u) do:
    if index(v) < minIndex and index(v) > j then:
        minIndex ← index(v)
return minIndex
```

Αλγόριθμος 4.5. Edge-Iterator (Schank, 2007)

Εάν ο πίνακας γειτνίασης είναι ήδη στοιχισμένος μπορούμε να βρούμε τις γειτονικές ακμές $(v,w)\epsilon E$ σε χρόνο ίσο με $d(u)+d(w)$. Η στοίχιση του πίνακα γειτνίασης, όπως έχουμε προαναφέρει, μπορεί να επιτευχθεί σε χρόνο $\sum_{u\epsilon V} d(u)ln(d(u))$. Βάσει των παραπάνω ο αλγόριθμος του edge-iterator μπορεί να επιτευχθεί σε ασυμπτωτικό χρόνο ίσο με $O(m \cdot d_{max}) \subseteq O(mn)$. Στη πραγματικότητα ο πίνακας γειτνίασης δεν θα είναι πάντα σε στοίχιση, με αποτέλεσμα ο αλγόριθμος να χρειάζεται χρόνο, περίπου ίσο με αυτόν του node-iterator.

στ. Αλγόριθμος Forward

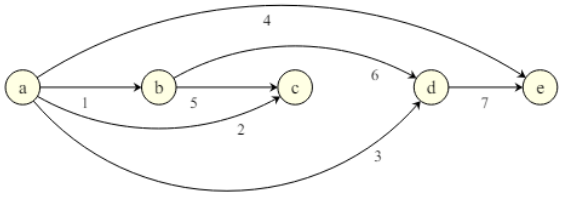
Ο αλγόριθμος Forward αναπτύχθηκε από τους Schank and Wagner (2005) ως βελτίωση της μεθόδου του edge-iterator. Αντίθετα με τον προηγούμενο αλγόριθμο, ο forward δεν χρησιμοποιεί τον πίνακα γειτνίασης αλλά ένα καινούργιο δυναμικό σύνολο $A(u)$ το οποίο στην αρχή είναι άδειο στο οποίο προσθέτει τα σύνολα των τριγώνων. Ουσιαστικά ο αλγόριθμος απαριθμεί και αποθηκεύει μόνο τους προς τα εμπρός γείτονες $A(u)$ του u, δηλαδή τους κόμβους που έχουν έρθει και αποθηκευτεί μετά τον κόμβο u. Με τον τρόπο αυτό ο αλγόριθμος διασφαλίζει ότι κάθε πιθανή τριάδα θα ελεγχθεί μία μόνο φορά, με αποτέλεσμα να μειώνει τον απαραίτητο χρόνο που χρειάζεται ο αλγόριθμος για να τρέξει.

```
Function Forward
triangle_count= 0
for u ∈ V do:
    A(u)<- ∅
for v ∈ V do:
    for w ∈ N(v) do:
        if index(v)<index(w) then:
            for u ∈ A(v) ∩ A(w) do:
                tringle_count+= 1
            A(w) < −A(w)U{v}
```

Αλγόριθμος 4.6. Forward (*Schank and Wagner, 2005)*

Ο χρόνος που απαιτεί ο αλγόριθμος για να τρέξει φράζεται από την έκφραση $\sum_{\{u,w\}\epsilon E} d_{in}(u) + d_{in}(w)$ το οποίο ασυμπτωτικά μπορεί να εκφραστεί ως $O(m^{3/2})$. Παρακάτω παρουσιάζεται ένα παράδειγμα για το πως λειτουργεί ο αλγόριθμος forward.

edge	$A(b)$	$A(c)$	$A(d)$	$A(e)$	triangles
1 (a,b)	a				
2 (a,c)	a	a			
3 (a,d)	a	a	a		
4 (a,e)	a	a	a	a	
5 (b,c)	a,b	a	a	a	$\{a,b,c\}$
6 (b,d)	a,b	a	a,b	a	
7 (d,e)	a,b	a	a,b	a,d	$\{a,d,e\}$



Σχήμα 4.1 Παράδειγμα αλγόριθμου forward (*Schank and Wagner, 2005)*

ζ. Αλγόριθμοι με δομή hash

Με τον όρο Hash αναφερόμαστε σε διαδικασίες μετατροπής των δεδομένων σε μια διαφορετική μορφή συγκεκριμένου μεγέθους. Ένας υπολογιστής δεν λειτουργεί και καταλαβαίνει όπως ένας άνθρωπος, έτσι με το Hashing αποθηκεύουμε τα δεδομένα μας σε μια μορφή πιο κατανοητή και εύκολα προσβάσιμη για τον υπολογιστή. Με τον τρόπο αυτό ο υπολογιστείς μπορεί να ανακτά και να επεξεργάζεται τα δεδομένα πολύ πιο εύκολα και γρήγορα. Σε σύνολα γράφων για να το καταφέρουμε αυτό αποθηκεύουμε τα σύνολα γειτόνων κάθε κόμβου σε λίστες $A[u] = (0,1,1,..)$ όπου με 0 συμβολίζεται ότι οι κόμβοι δεν είναι γειτονικοί, ενώ με 1 ότι είναι δύο κόμβοι είναι γειτονικοί. Στη καταμέτρηση τριγώνων χρησιμοποιούμε τις μορφές Hash για να βρούμε ταχύτερα τους κοινούς γείτονες δύο κόμβων. Ένας από τους πρώτους αλγορίθμους ο οποίος έκανε χρήση της μορφής Hash είναι ο αλγόριθμος edge-iterator hashed (Chiba and Nishizeki, 1985) ο οποίος λειτουργεί με τον ίδιο τρόπο όπως ο edge iterator με τη διαφορά ότι για να βρει την τομή των κοινών γειτόνων δύο κόμβων αποθηκεύει τους γείτονες των κόμβων σε λίστες hashed μορφής και έπειτα υπολογίζει την τομή τους. Ο ασυμπτωτικός χρόνος που απαιτεί ο αλγόριθμος για να τρέξει φράζεται στην ίδια τιμή με αυτό του edge-iterator $O(n \cdot d_{max})$. Μπορούμε επίσης να συνδυάσουμε τον αλγόριθμο forward με τις μορφές hashed και να πάρουμε τον forward hashed (Schank and Wagner, 2005) ο οποίος φράζεται στον ίδιο ασυμπτωτικό χρόνο με αυτό της μεθόδου forward. Σε σύνολα δεδομένων τα οποία δεν έχουν αποθηκευτεί σε μορφή κατανοητή για ένα υπολογιστή, όπως για παράδειγμα χαρακτήρες με ονόματα, αναμένουμε ότι οι μέθοδοι hashed θα χρειάζονται λιγότερο χρόνο και μνήμη για να μπορούν να τρέξουν. Οι αλγόριθμοι που αναφέρουμε μπορούν να τρέξουν με διάφορες παραμετροποιήσεις (Bader, 2023).

4.3 Χρησιμότητα εύρεσης τριγώνων σε δίκτυα

Όπως προαναφέραμε τα τρίγωνα είναι υποσύνολα ενός δικτύου τα οποία είναι πλήρως συνδεδεμένα μεταξύ τους (3-clique). Σε κοινωνικά δίκτυα αναμένουμε να έχουμε μεγαλύτερο πλήθος τριγώνων αφού αν ένα άτομο έχει δύο φίλους, τότε οι φίλοι αυτοί, έχουν μεγαλύτερη πιθανότητα να γίνουν φίλοι μεταξύ τους. Η εύρεση του πλήθους τριγώνων είναι ζωτικής σημασίας κατά την ανάλυση κοινωνικών δικτύων αφού μπορεί να χρησιμοποιηθεί για να κατανοήσουμε τη

δομή του δικτύου. Οι αλγόριθμοι εύρεσης τριγώνων μπορούν να αξιοποιηθούν σε διάφορους τομείς ανάλυσης κοινωνικών γράφων.

a) Συντελεστές συσταδοποίησης και μεταβατικότητας

Ο συντελεστής συσταδοποίησης (Clustering Coefficient) ενός κόμβου, αναπτύχθηκε από τους Watts and Strogatz (1998) και ορίζεται ως το μέτρο που διερευνά την τάση δημιουργίας κλικών (ομάδων) ανάμεσα στα στοιχεία. Ουσιαστικά το μέτρο αυτό μπορεί να μας υποδείξει την πυκνότητα ενός υποσυνόλου του γράφου. Ο συντελεστής συσταδοποίησης για έναν κόμβο v ορίζεται ως εξής

$$cc(v) = \frac{\Delta_v}{\binom{d_v}{2}}$$

όπου το Δ_v συμβολίζει το πλήθος των τριγώνων στο οποίο ανήκει ο κόμβος v, ενώ με d_v συμβολίζουμε το βαθμό του κόμβου v. Για τους κόμβους που έχουν μόνο έναν γείτονα ορίζουμε τον συντελεστή συσταδοποίησης ως 0 ή 1 (Rochat, 2009). Ο συντελεστής συσχέτισης ενός γράφου $G = (V, E)$ ή ολικός συντελεστής συσχέτισης (Global Clustering Coefficient) γνωστός και ως μεταβατικότητα (Transitivity) ορίζεται ως

$$T(G) = \frac{3 \cdot \Delta_v}{\sum_{v \in V'} \binom{d_v}{2}}$$

όπου $V' = \{v \in V : d(v) \geq 2\}$. Όπως προαναφέραμε, για τον συντελεστή συσταδοποίησης ενός κόμβου τον ορίζουμε με 0 ή 1, αν έχει έναν ή κανέναν γείτονες, αλλά στον συντελεστή συσταδοποίησης ολόκληρου γράφου τα στοιχεία αυτά δεν τα συνυπολογίζουμε. Με τους συντελεστές αυτούς μπορούμε να κατανοήσουμε καλύτερα τους τη δομή ενός γράφου. Ο συντελεστής ομαδοποίησης ενός κόμβου μας βοηθάει να αντιληφθούμε στο πόσο κοντά βρίσκεται ένας κόμβος να σχηματίσει ένα πλήρη υπογράφο (κλίκα), με το οποίο μπορούμε να αντιληφθούμε την τάση που έχουν οι γείτονες ενός κόμβου να ενώνονται μεταξύ τους. Για παράδειγμα αν στο Facebook όλοι οι φίλοι ενός ατόμου είναι φίλοι μεταξύ τους, τότε ο ολικός συντελεστής συσχέτισης θα είναι ίσος με την μονάδα. Με αυτό το τρόπο μπορούμε να καταλάβουμε ότι, αν υπάρξει ένας καινούργιος φίλος του ατόμου αυτού, τότε θα περιμένουμε μελλοντικά να γίνει φίλος και με όλη την υπόλοιπη ‘παρέα’. Ο συντελεστής ομαδοποίησης παρέχει μια συνολική εικόνα για τη τάση ομαδοποίησης του δικτύου. Το μέτρο αυτό δηλώνει την πιθανότητα δύο φίλοι Α και Β

ενός κόμβου Γ να είναι φίλοι μεταξύ τους και μας υποδεικνύει το πόσο συγκεντρωμένο είναι το δίκτυο συνολικά, αντί να εστιάζει σε μεμονωμένους κόμβους.

b) Εύρεση κοινοτήτων

Οι Berry et al. (2011) χρησιμοποιούν το πλήθος τριγώνων σε αλγόριθμο εύρεσης κοινοτήτων. Ουσιαστικά λόγου του resolution limit (Fortunato and Barthelemy, 2007) που μπορεί να δημιουργηθεί λόγο της συνάρτησης ποιοτικής αναπαράστασης modularity (βλέπε 5.3.1) οι συγγραφείς χρησιμοποιούν το πλήθος των τριγώνων για να αναπροσαρμόσουν τα βάρη των ακμών. Με το τρόπο αυτό επιτυγχάνουν να βρίσκει ο αλγόριθμος καλύτερα αποτελέσματα για την εύρεση κοινοτήτων.

c) Ανίχνευση spam ιστοσελίδων

Η ταχεία αύξηση χρήσης μηχανών αναζήτησης έχει δημιουργήσει το φαινόμενο ανάπτυξης ιστοσελίδων spam (webspam). Ως spam χαρακτηρίζεται μια ιστοσελίδα που 'παραπλανεί' τις μηχανές αναζήτησης για να οδηγήσουν τους χρήστες (λανθασμένα) σε ορισμένους ιστότοπους. Κατά την έρευνά τους, οι Becchetti et al. (2008) εντόπισαν πως η κατανομή του πλήθους τριγώνων ανάμεσα σε spam και όχι-spam ιστοσελίδες, μπορεί να συμβάλει στην ταξινόμηση μιας ιστοσελίδας σε μια από τις δύο κλάσεις.

ΚΕΦΑΛΑΙΟ 5
Κοινότητες

5.1 Εισαγωγή

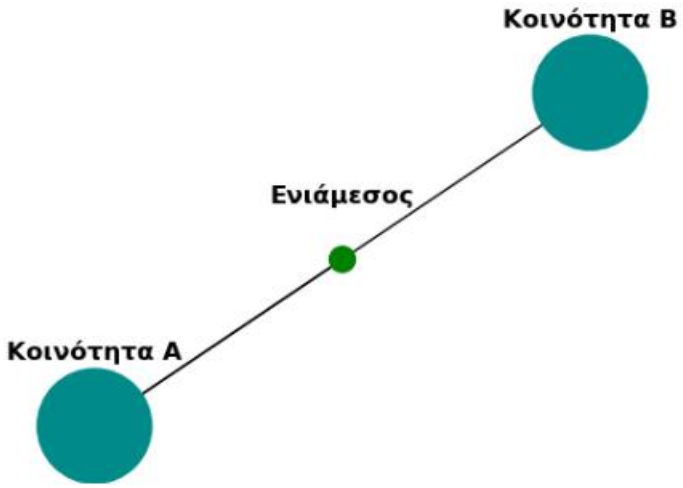
Σε δίκτυα γράφων οι κόμβοι δημιουργούν πυκνά συνδεδεμένες ομάδες τις οποίες τις αποκαλούμε κοινότητες. Τις ομάδες αυτές συνήθως δεν τις γνωρίζουμε πριν την ανάλυση των υπό μελέτη δεδομένων. Μια ιδανική δομή κοινοτήτων είναι να υπάρχουν πολλές ακμές μεταξύ κόμβων της ίδιας κοινότητας και ελάχιστες ακμές μεταξύ κοινοτήτων. Στα κοινωνικά δίκτυα, η ανίχνευση κοινοτήτων κατέχει κεντρικό ρόλο στην αποκάλυψη της δομής των κοινωνικών αλληλεπιδράσεων και της δυναμικής εντός των ομάδων. Η κατανόηση κοινοτήτων μέσα στα κοινωνικά δίκτυα προσφέρει πολύτιμες γνώσεις σχετικά με τον τρόπο ροής των πληροφοριών, τον τρόπο διαμόρφωσης των απόψεων με τον οποία τα άτομα διαμορφώνουν σχέσεις και συνεργασίες. Για την εύρεση κοινοτήτων έχουν αναπτυχθεί διάφορες μέθοδοι (Fortunato, 2010), ποικίλων τύπων, ανά τα χρόνια. Στο κεφάλαιο αυτό θα σχολιαστούν αλγόριθμοι, οι οποίοι μετά από κάποιους ελέγχους μας δίνουν πολύ καλά αποτελέσματα σε γρήγορο χρόνο. Τους τύπους αλγορίθμων που θα αναπτυχθούν, μπορεί κάποιος να τους διακρίνει σε ιεραρχικούς αλγορίθμους, αλγορίθμους που βασίζονται στη μεγιστοποίηση μιας ποιοτικής συνάρτησης, αλγορίθμους που κάνουν χρήση μεθόδων τυχαίων περιπάτων και σε αλγορίθμους που βρίσκουν αλληλοκαλυπτόμενες κοινότητες.

5.2 Ιεραρχικές μέθοδοι

Οι πιο κλασσικές μέθοδοι εύρεσης κοινοτήτων είναι παραλλαγές ιεραρχικών μεθόδων που είναι πολύ χρήσιμες στη μελέτη κοινωνικών δικτύων. Σε αυτή τη κατηγορία εμπίπτουν αλγόριθμοι που παράγουν μια ιεραρχία δενδροειδούς μορφής. Ουσιαστικά, μπορούν να μετατρέψουν και να παρουσιάσουν ένα υπάρχον δίκτυο σε ένα δενδρόγραμμα όπου βάσει κάποιας συνάρτησης που χρησιμοποιούν, θεωρούν το κάθε κόμβο ως πιο σημαντικό ή πιο ασήμαντο. Υπάρχουν δύο βασικές

κατηγορίες ιεραρχικών μεθόδων, οι συσωρευτικές (agglomerative) και οι διαιρετικές (divisive). Στις συσωρευτικές μεθόδους θεωρούμε αρχικά πως ο κάθε κόμβος αποτελεί μια ξεχωριστή κοινότητα. Έπειτα, βάσει της συνάρτησης που χρησιμοποιούμε στην μελέτη υπολογίζουμε την ομοιότητα (απόσταση) μεταξύ των κοινοτήτων και ενώνουμε τους κόμβους που έχουν τη μεγαλύτερη ομοιότητα, όπου πλέον τους θεωρούμε ως μια ομάδα. Μετά τη συνένωση των κόμβων ξανά-υπολογίζουμε της ομοιότητες μεταξύ των κοινοτήτων. Επαναλαμβάνουμε τη διαδικασία που προαναφέραμε έως ότου όλοι οι κόμβοι να έχουν ενωθεί σε μια ομάδα. Ένα μεγάλο μειονέκτημα των συσωρευτικών μεθόδων είναι υψηλή απαίτηση πόρων και χρόνου. Σε μικρά σύνολα δεδομένων δεν μας προβληματίζει η ταχύτητα του αλγορίθμου αφού όποια μέθοδο και να χρησιμοποιήσουμε δεν θα χρειαστεί πολύς χρόνος για να τρέξει ο αλγόριθμος. Βέβαια, τα κοινωνικά έχοντας τεράστιο όγκο πληροφορίας, κάνει τους αλγορίθμους που απαιτούν πολύ χρόνο απρόσιτους. Επίσης, κατά τη διάρκεια των συνενώσεων οι κόμβοι που έχουν ομαδοποιηθεί, δεν πρόκειται να διασπαστούν σε μετέπειτα βήματα, αλλά θα παραμείνουν ενωμένα. Η δεύτερη κατηγορία ιεραρχικών αλγορίθμων είναι αυτή των διαιρετικών. Οι αλγόριθμοι αυτού του τύπου, ξεκινούν με μία μόνο ομάδα και θεωρούν πως όλοι οι κόμβοι ανήκουν στην ομάδα αυτή. Η λογική στην οποία βασίζονται οι αλγόριθμοι αυτού του είδους είναι, ότι βρίσκουν μέσα στις ομάδες τα πιο ανόμοια στοιχεία και τα αφαιρούν από την ομάδα που βρίσκονται. Το στοιχείο που απομακρύνεται θεωρείτε πλέον μια ομάδα αποτελούμενη από το στοιχείο αυτό. Οι διαιρετικοί αλγόριθμοι με το τρόπο που κάνουν τους υπολογισμούς για την απομάκρυνση των στοιχείων γίνονται περισσότερο απαιτητικοί σε θέμα χρόνου και μνήμης για τον υπολογιστή με αποτέλεσμα να μη χρησιμοποιούνται σχεδόν ποτέ σε δεδομένα κοινωνικών δικτύων.

Θα παρουσιαστεί στη συνέχεια ένας αρκετά διαδεδομένος ιεραρχικός αλγόριθμος εύρεσης κοινοτήτων, ο οποίος κατασκευάστηκε από τους Girvan-Newman (2001). Ο αλγόριθμος Girvan-Newman είναι ένας διαιρετικός ιεραρχικός αλγόριθμος, ο οποίος έχει σαν στόχο να εντοπίζει κοινότητες, μέσω της αφαίρεσης ενδιάμεσων σε κοινότητες κόμβων. Η μέθοδος, χρησιμοποιώντας την κεντρικότητα ενδιαμεσότητας (βλέπε 3.5.4), εντοπίζει τα στοιχεία τα οποία είναι ενδιάμεσα σε δύο κοινότητες και τα αφαιρεί ώστε να τις αποκόψει. Όπως είχαμε προαναφέρει, το μέτρο αυτό εντοπίζει τους κόμβους που είναι πιο κεντρικοί στη ροή της πληροφορίας, τους οποίους θεωρούμε ως κόμβους που δεν είναι κεντρικοί εντός κάποιας κοινότητας.

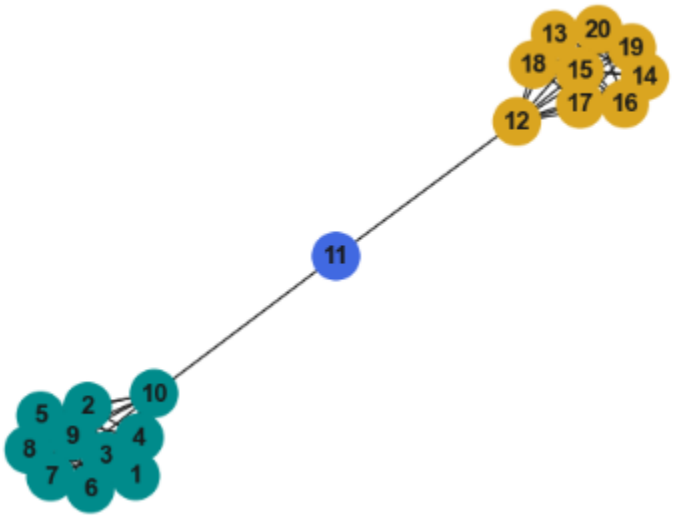


Σχήμα 5.1. Ενδιάμεσος σε δύο κοινότητες κόμβος.

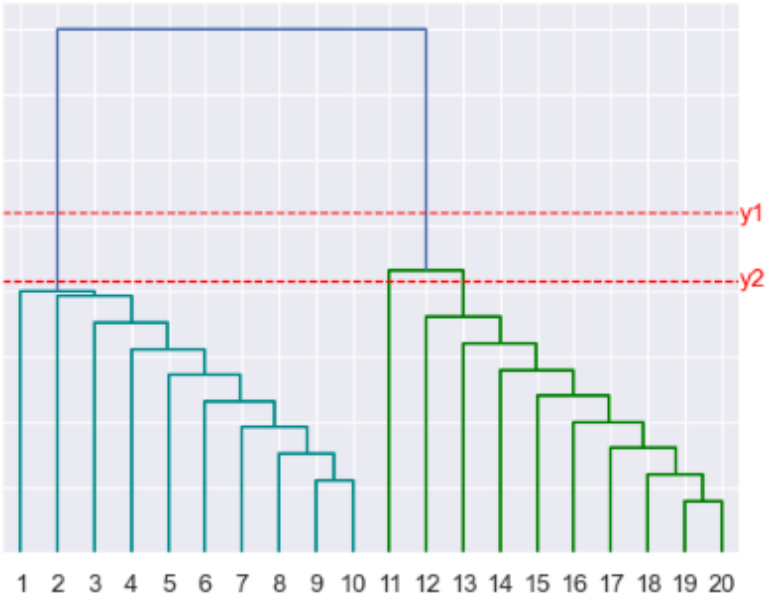
Τα βήματα που ακολουθεί ο αλγόριθμος είναι τα εξής:

1. Υπολογίζει τη κεντρικότητα ενδιαμεσότητας των κόμβων.
2. Βρίσκει τον κόμβο με την υψηλότερη τιμή κεντρικότητας. Αν υπάρχουν περισσότεροι από ένα κόμβοι, που έχουν τιμή κεντρικότητας, ίση με αυτό του μεγίστου, επιλέγει έναν από τους κόμβους αυτούς τυχαία.
3. Αφαιρεί όλες τις ακμές του κόμβου που έχει επιλέξει στο προηγούμενο βήμα.
4. Σε αυτό το σημείο ο αλγόριθμος επιστρέφει στον πρώτο βήμα έως ότου να μην υπάρχουν ακμές στο σύνολο.
5. Σε κάθε βήμα οι κόμβοι που παραμένουν συνδεμένοι με κάποια ακμή θεωρούνται ως κοινότητα μεταξύ τους.

Ο αλγόριθμος αφαιρεί σταδιακά τα πιο ενδιάμεσα στοιχεία. Οι κόμβοι που αφαιρούνται σε αρχικά στάδια είναι κόμβοι που είναι σημαντικοί για να διατηρείται συνδεδεμένος ο γράφος. Αυτοί οι κόμβοι αποτελούν συνήθως μέρος πυκνών υπογράφων ή λειτουργούν ως γέφυρες που συνδέουν διαφορετικές κοινότητες. Αντίθετα, οι κόμβοι που παραμένουν συνδεμένοι σε μεταγενέστερα στάδια του αλγορίθμου, ακόμα και μετά την αφαίρεση πολλών ακμών, είναι ζωτικής σημασίας για τη συγκράτηση του δικτύου. Με τον τρόπο αυτό, μπορούμε να δημιουργήσουμε μια ιεραρχία και να παρουσιάσουμε το γράφο σε δενδροειδή μορφή. Παρακάτω παρουσιάζεται ένα παράδειγμα ενός γράφου με είκοσι κόμβους. Οι πρώτοι δέκα κόμβοι συνενώνονται πλήρως μεταξύ τους, οι κόμβοι 12-20 συνδέονται πλήρως μεταξύ τους, ενώ ο κόμβος 11 έχει ως γείτονες μόνο τους κόμβους 10 και 12.



Σχήμα 5.2. Παράδειγμα 2 πλήρων γράφων που ενώνονται με ένα κόμβο.



Σχήμα 5.3. Παράδειγμα δενδροδιαγράμματος.

Το πρόβλημα που δημιουργείται πλέον, είναι η επιλογή του πλήθους των κοινοτήτων. Για παράδειγμα αν χωρίσουμε το δίκτυο σε δύο κοινότητες (βλέπε y_1 στο Σχήμα 5.3), τότε θα έχουμε τον κόμβο 11 να ανήκει στο στην ίδια κοινότητα με τους κόμβους 12-20, ενώ έχει μόνο έναν 'φίλο' στη κοινότητα αυτή. Αντίθετα, αν χωρίσουμε το δίκτυο σε τρεις κοινότητες (y_2) θα έχουμε μια ολόκληρη κοινότητα να αποτελείται από μόνο έναν κόμβο. Για να λύσουν το πρόβλημα αυτό, οι συγγραφείς παρουσίασαν μετέπειτα το ποσοτικό μέτρο modularity βάσει του οποίου επιλέγουμε

το βέλτιστο διαχωρισμό. Σε κάθε επανάληψη ο αλγόριθμος υπολογίζει και το modularity του εκάστοτε διαχωρισμού και επιλέγεται αυτός για τον οποίο η τιμή του modularity είναι μέγιστη. Το ποσοτικό αυτό μέτρο χρησιμοποιείται σε πολλούς αλγορίθμους και θα αναλυθεί διεξοδικά, στη συνέχεια αυτού του κεφαλαίου. Το υπολογιστικό πρόβλημα που δημιουργείται είναι ότι πρέπει να υπολογίσουμε τη κεντρικότητα ενδιαμεσότητας σε κάθε αφαίρεση κόμβου το οποίο έχει υπολογιστικό κόστος χρόνου $O(nm)$. Για τον λόγο αυτό, ο συνολικός χρόνος που απαιτεί ο αλγόριθμος για να τρέξει μπορεί να φτάσει έως και $O(nm^2)$. Το πρόβλημα που υπάρχει στα κοινωνικά δίκτυα είναι ότι έχουν πάρα πολύ μεγάλο πλήθος στοιχείων, οπότε αυτό που θέλουμε είναι όχι μόνο να βρούμε αλγορίθμους που θα βρίσκουν σωστά της κοινότητες αλλά θα έχουν και χαμηλό κόστος υπολογισμού. Για να μειώσει το υπολογιστικό κόστος θα μπορούσε κάποιος να υπολογίσει την κεντρικότητα ενδιαμεσότητας των κόμβων μόνο μια φορά και να αφαιρεί σταδιακά τους κόμβους, ξεκινώντας με αυτόν που έχει την υψηλότερη τιμή ενδιαμεσότητας. Επανυπολογίζοντας την ενδιαμεσότητα σε κάθε αφαίρεση κόμβου, θα έχουμε καλύτερη προσέγγιση στο πρόβλημα, αφού είναι πιο σίγουρο πως κάποιοι από τους ενδιάμεσους κόμβους θα έχουν υψηλή τιμή κεντρικότητας σε κάθε στάδιο του αλγορίθμου. Σε περίπλοκα δίκτυα μπορεί να υπάρξουν κόμβοι, με το ίδιο μεγαλύτερο μέτρο ενδιαμεσότητας. Σε τέτοιου είδους περιπτώσεις ο αλγόριθμος αφαιρεί έναν και μόνο κόμβο, από αυτούς τυχαία. Με σκοπό τη μείωση του χρόνου που απαιτεί ο αλγόριθμος και δεδομένου ότι η επιλογή αφαίρεσης του κόμβου με υψηλότερη τιμή ενδιαμεσότητας γίνεται τυχαία, οι συγγραφείς πρότειναν, να μην αφαιρούμε μόνο έναν από τους κόμβους με την υψηλότερη ενδιαμεσότητας αλλά να αφαιρεθούν όλοι οι κόμβοι που έχουν ίση μέγιστη τιμή. Αντίστοιχα με τον αλγόριθμο των Girvan-Newman λειτουργεί και ο αλγόριθμος των Fortunato, Latora and Marchiori (2004), ο οποίος αντί να χρησιμοποιεί την κεντρικότητα ενδιαμεσότητας, χρησιμοποιεί την κεντρικότητα πληροφορίας (Information Centrality) (Latora and Marchiori, 2004). Οι δυο αλγόριθμοι δίνουν πολύ παρόμοια αποτελέσματα, αλλά λόγω της υψηλότερης απαίτηση πόρων για να υπολογιστεί η κεντρικότητα πληροφορίας σε κάθε βήμα, προτιμήθηκε η χρήση του αλγορίθμου που παρουσιάστηκε από τους Girvan-Newman.

5.3 Μέθοδοι μεγιστοποίησης ποιοτικών συναρτήσεων

Το πρόβλημα της ανίχνευσης κοινοτήτων αναφέρεται στην ανάγκη να διαχωρίσουμε ένα δίκτυο σε κοινότητες με πυκνή σύνδεση μεταξύ των κόμβων εντός της κάθε κοινότητας. Αντίθετα, οι κόμβοι που ανήκουν σε διαφορετικές κοινότητες έχουν αραιές συνδέσεις μεταξύ τους. Στις περισσότερες των περιπτώσεων, οι αλγόριθμοι εύρεσης κοινοτήτων χρησιμοποιούνται σε δίκτυα στα οποία δεν γνωρίζουμε ποιες είναι οι κοινότητες, με αποτέλεσμα να μη μπορούμε να αξιολογήσουμε τα αποτελέσματα που παράγονται. Για να μπορέσουμε να αξιολογήσουμε αλλά και να συγκρίνουμε τη δομή κοινοτήτων που παράγονται από διάφορους αλγόριθμους, έχουν παραχθεί διάφορα ποσοτικά μέτρα. Στην ενότητα αυτή, θα αναφερθούμε στο modularityτο οποίο είναι το πιο δημοφιλές μέτρο αξιολόγησης ομαδοποιήσεων, αλλά θα μελετήσουμε και αλγορίθμους, οι οποίοι το αξιοποιούνε το μέτρο αυτό για να βρούνε κοινότητες σε ένα δίκτυο.

5.3.1 Modularity

Για να μπορούμε να αξιολογήσουμε τα αποτελέσματα αλγορίθμων εύρεσης κοινοτήτων, οι Newman και Girvan (Newman and Girvan, 2004) προτείναν ένα καινούργιο ποσοτικό μέτρο, το modularity. Υποθέτοντας ότι οι υψηλές τιμές modularity υποδεικνύουν καλές διαμερίσεις σε κοινότητες, μπορούμε να θεωρήσουμε πως η διαμέριση που αντιστοιχεί στη μέγιστή τιμή του, μπορεί να θεωρηθεί ως η καλύτερη διαμέριση του δικτύου. Η λογική αυτή αποτέλεσε κίνητρο για τη δημιουργία αλγορίθμων που στηρίζονται στη μεγιστοποίηση της συνάρτησης modularity για τη διαμέριση δικτύων σε κοινότητες.

Το modularity είναι η πιο δημοφιλής συνάρτηση ποιότητας αναπαράστασης μιας διαμέρισης. Όπως προαναφέραμε, τέτοιες συναρτήσεις αξιολογούν την διαμέριση που έχει γίνει στο δίκτυο που μελετάμε σε μικρότερες συστάδες. Κατά τη διαμέριση ενός δικτύου ο κύριος στόχος είναι να έχουμε μεγάλο πλήθος συνενώσεων μεταξύ των κόμβων της ίδιας κοινότητας. Αντίθετα, θέλουμε να υπάρχει μικρό πλήθος συνενώσεων μεταξύ των κόμβων διαφορετικών κοινοτήτων. Το modularity είναι μια συνεχής αριθμητική τιμή, που παίρνει τιμές στο διάστημα [0,1]. Πρακτικά είναι πολύ δύσκολο να πάρει τιμές ίσες με τα άκρα του διαστήματος, διότι αυτό θα σήμαινε ότι ο διαχωρισμός που έχει γίνει είναι ιδανικός (ή το αντίθετο). Συνήθως οι τιμές του modularity κυμαίνονται στο εύρος [0.3,0.7], ενώ είναι σπάνιο να πάρουμε τιμές που δεν ανήκουν στο

διάστημα αυτό. Στη βιβλιογραφία υπάρχουν διάφορες εκδόσεις για τη μαθηματική αναπαράσταση του modularity, οι οποίες διαφέρουν ανάλογα με τον τύπο γράφου που μελετάμε. Για μη κατευθυνόμενους γράφους ο τύπος που χρησιμοποιείται

$$Q = \frac{1}{2m}\Sigma_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right](s_v s_w + 1) \qquad (5.1)$$

όπου το m είναι ο συνολικός αριθμός ακμών που έχει το δίκτυο. Με k_i συμβολίζουμε το βαθμό της i-στης συστάδας, δηλαδή το πλήθος των ακμών των κόμβων εντός της συστάδας. Το $s_i = 1$ σε περίπτωση που ο κόμβος ανήκει στην συστάδα i, ενώ όταν δεν ανήκει στην συστάδα θέτουμε το $s_i = -1$. Με A συμβολίζουμε τον πίνακα γειτνίασης του δικτύου. Αντίστοιχα, μπορούμε να παρουσιάσουμε τον τύπο (5.1) με τη χρήση της συνάρτησης του Kronecker ως εξής

$$Q = \frac{1}{2m}\Sigma_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\delta(c_v, c_w) \qquad (5.2)$$

όπου τα c_v, c_w συμβολίζουν τις συστάδες. Η συνάρτηση δ του Kronecker δίνεται ως εξής

$$\delta(c_v, c_w) = \begin{cases} 1, & \alpha\nu\ c_v = c_w \\ 0, & \alpha\nu\ c_v \neq c_w. \end{cases}$$

5.3.2 Βελτιστοποίηση υπολογισμού modularity

Το modularity ποσοτικοποιεί την ποιότητα της διαμέρισης ενός δικτύου σε κοινότητες, αξιολογώντας τη διαφορά μεταξύ του παρατηρούμενου και του αναμενόμενου αριθμού ακμών μεταξύ των κοινοτήτων. Ουσιαστικά οι αλγόριθμοι που βασίζονται στο modularity και έχουν ως σκοπό τη διαμέριση ενός δικτύου σε κοινότητες, υπολογίζουν τη διαφορά του modularity πριν και μετά τη διαμέριση, με στόχο τη μεγιστοποίηση του. Ο ακριβής υπολογισμός του modularity περιλαμβάνει τον επανυπολογισμό της συνάρτησης για κάθε πιθανή συγχώνευση. Όπως έχει αποδειχθεί, το πρόβλημα του υπολογισμού του μεγίστου modularity είναι NP-Complete (Brandes et al. 2006) (δεν υπάρχει αυτή τη στιγμή αλγόριθμος ο οποίος να μας δίνει τη μέγιστη τιμή modularity για όλους τους πιθανούς συνδυασμούς διαμερίσεων). Ως αποτέλεσμα, οι αλγόριθμοι που βασίζονται στην μεγιστοποίηση της συνάρτησης δεν μας δίνουν πάντα τη βέλτιστη λύση, αλλά προσπαθούν με διάφορες τεχνικές να προσεγγίσουν μια πολύ ικανοποιητική λύση.

Για τη βελτιστοποίηση του modularity έχουν προταθεί πολλοί αλγόριθμοι και διάφορες τεχνικές. Οι αλγόριθμοι που θα παρουσιαστούν, χρησιμοποιούν άπληστες (greedy) στρατηγικές για τη μεγιστοποίηση του modularity. Με τον όρο ‘άπληστη’ αναφερόμαστε σε τεχνικές (αλγορίθμους), όπου οι επιλογές σε κάθε βήμα γίνονται σύμφωνα με τα άμεσα οφέλη, χωρίς να

λαμβάνονται υπόψη πιθανές μελλοντικές συνέπειες. Μια άπληστη τεχνική, δεν παράγει την βέλτιστη λύση αλλά δίνει τοπικά βέλτιστη λύση η οποία προσεγγίζει την ολικά βέλτιστη, σε εύλογο χρόνο. Ο πρώτος αλγόριθμος που αναπτύχθηκε για τη μεγιστοποίηση του modularity είναι αυτή του Newman (Newman, 2004). Πρόκειται για μια ιεραρχική μέθοδο συσσώρευσης ομάδων όπου οι κοινότητες κόμβων συνδέονται διαδοχικά για να δημιουργήσουν μεγαλύτερες κοινότητες, με την προϋπόθεση ότι το modularity αυξάνεται μετά από κάθε συγχώνευση. Αρχικά, ο αλγόριθμος θεωρεί πως υπάρχουν n κοινότητες (κάθε κόμβος θεωρείται μια ξεχωριστή ομάδα). Σταδιακά, ενώνει τις δύο κοινότητες για τις οποίες η αύξηση της ποιοτικής συνάρτησης, θα είναι η μέγιστη. Η ένωση των πρώτων δύο κοινοτήτων μεταξύ τους θα μειώσει το συνολικό πλήθος από n σε $n-1$, παρέχοντάς μας μια νέα διαμέριση του δικτύου. Οι λοιπές κοινότητες, ενώνονται βάσει της ίδιας πρακτικής. Για να απλοποιήσουμε τη περιγραφή του αλγορίθμου ας ορίσουμε τις ακόλουθες δύο ποσότητες (Clauset, Newman and Moore, 2004)

$$e_{ij} = \frac{1}{2m}\sum_{vw} A_{vw}\delta(c_u, i)\delta(c_w, j) \tag{5.3}$$

η οποία δίνει το ποσοστό των ακμών του δικτύου που συνδέουν τις κορυφές της κοινότητας i με τις κορυφές της κοινότητας j. Επιπλέον ορίζουμε τη ποσότητα

$$a_i = \frac{1}{2m}\sum_{v} k_v\delta(c_v, i) \tag{5.4}$$

η οποία είναι το αναμενόμενο ποσοστό ακμών μεταξύ των κόμβων στην κοινότητα i. Με αυτό τον τρόπο και έχοντας ότι $\delta(c_v, c_w) = \sum_i \delta(c_v, i)\delta(c_w, i)$, από τον τύπο (5.2), μπορούμε να γράψουμε

$$Q = \frac{1}{2m}\Sigma_{vw}\left[A_{vw} - \frac{k_v k_w}{2m}\right]\sum_i \delta(c_u, i)\delta(c_w, i)$$

$$= \sum_i \left[\frac{1}{2m}\sum_{vw} A_{vw}\delta(c_u, i)\delta(c_w, i) - \frac{1}{2m}\sum_v k_v\delta(c_v, i)\frac{1}{2m}\sum_w k_w\delta(c_w, i)\right]$$

$$= \sum_i (e_{ij} - a_i^2). \tag{5.6}$$

Αν θεωρήσουμε ότι θα χωρίσουμε το σύνολο των ακμών μας σε k συστάδες, τότε το e_{ij} συμβολίζει ποσοστό των ακμών που συνδέουν τις κορυφές της κοινότητας i με τις κορυφές της κοινότητας j. Έτσι, η διαφορά του modularity σε κάθε βήμα υπολογίζεται ως εξής

$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j). \tag{5.7}$$

Ωστόσο, το modularity των διαμερίσεων που διερευνάται κατά τη διαδικασία, υπολογίζεται πάντα για όλο τον γράφο και όχι απλώς για τα διαμερισμένα τμήματα του, διότι ο αλγόριθμος προσπαθεί να υπολογίσει τη μέγιστη τιμή σε ολόκληρο το δίκτυο. Το πρόβλημα της προσέγγισης

αυτής είναι ότι ο αλγόριθμος απαιτεί υψηλό υπολογιστικό χρόνο σε κάθε βήμα ώστε να επαναυπολογίζει τη συνάρτηση ποιότητας. Σε αντίθεση με άλλες τεχνικές, οι αλγόριθμοι που βελτιώνουν το modularity συνήθως δεν μας περιορίζουν από θέμα χρόνου, αλλά από θέμα χώρου. Αυτό συμβαίνει διότι για να υπολογιστεί το modularity πρέπει να έχουμε αποθηκεύσει τον πίνακα γειτνίασης, αλλά και τις αλλαγές που γίνονται σε αυτό με το πέρασμα των βημάτων των αλγορίθμων. Ένα βασικό πλεονέκτημα των αλγορίθμων αυτών είναι ότι, έχοντας τερματίσει, έχουμε ήδη το modularity της τελικής ομαδοποίησης και μπορούμε πολύ εύκολα να αποφασίσουμε ποια διαμέριση θα επιλέξουμε. Οι περιορισμοί που έχει το modularity είναι ότι κάποιες φορές, δεν μπορεί να ξεχωρίσει μικρές κοινότητες μεταξύ τους, με αποτέλεσμα να ενώνει μικρές κοινότητες μεταξύ τους οι οποίες δεν θα έπρεπε να θεωρηθούν ως μια. Οι περιορισμοί αυτοί είναι γνωστοί ως resolution limit (Fortunato and Barthelemy, 2007), το οποίο στην ουσία σημαίνει πως μεροληπτεί προς την ανίχνευση μεγαλύτερων κοινοτήτων ενώ αντίθετα μικρότερες κοινότητες που θα τις θεωρούσαμε ως ξεχωριστές κοινότητες, μπορεί να τις ενώσει.

5.3.3 Αλγόριθμος Louvain

Ο αλγόριθμος Louvain (Αλγόριθμος 5.2) είναι ένας άπληστος αλγόριθμος εύρεσης κοινοτήτων. Δημιουργήθηκε από τους Blondel et al. (2008) του πανεπιστημίου του Louvain, από όπου πήρε και το όνομα του. Ο αλγόριθμος κατατάσσεται στην κατηγορία των άπληστων αλγορίθμων, η οποία όπως προαναφέραμε, ακολουθεί τοπικά την επιλογή της βέλτιστης λύσης σε κάθε βήμα με σκοπό την εύρεση του ολικού βέλτιστου. Η μέθοδος Louvain είναι κατάλληλη για χρήση σε σταθμισμένα αλλά και μη-σταθμισμένα δίκτυα. Η μέθοδος περιλαμβάνει δύο βήματα τα οποία επαναλαμβάνονται συνεχώς (βλέπε Σχήμα 5.4). Έστω ότι έχουμε ένα δίκτυο αποτελούμενο από N κόμβους. Αρχικά, κάθε κόμβος θα θεωρηθεί μια ξεχωριστή κοινότητα (Σχήμα 5.4.a), έτσι σε αυτό το πρώτο βήμα έχουμε πλήθος κοινοτήτων ίσο με το πλήθος κόμβων. Έπειτα, για κάθε κόμβο i ελέγχονται οι γειτονικοί κόμβοι j και υπολογίζεται το κέρδος που θα υπάρχει στο modularity, εάν αφαιρέσουμε τον κόμβο i από τη κοινότητα του και τον τοποθετήσουμε στη κοινότητα του κόμβου j. Ο κόμβος i θα τοποθετηθεί στη κοινότητα του γειτονικού κόμβου j (Σχήμα 5.4.b) για την οποία θα έχουμε το υψηλότερο κέρδος (στη τιμή της modularity), με τη προϋπόθεση ότι το κέρδος είναι θετικό. Στη περίπτωση όπου υπάρχουν παραπάνω από ένας κόμβοι j που μας δίνουν τη μέγιστη

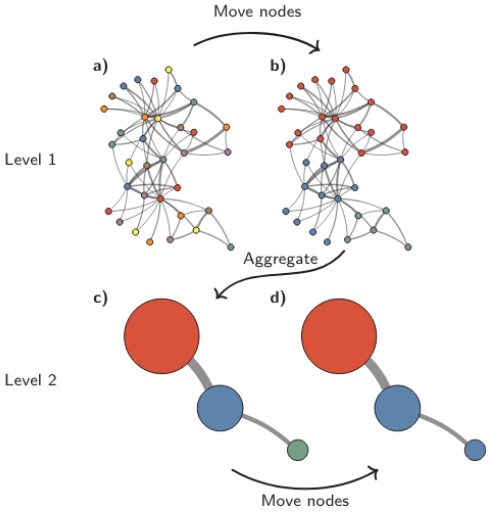
βελτίωση της ποιοτικής συνάρτησης, θα πρέπει να χρησιμοποιήσουμε ένα κανόνα για την επιλογή του κόμβου, ο οποίος θα συνδυαστεί με τον υπό-μελέτη κόμβο i. Στη παρούσα διπλωματική έχουμε θέσει ως κανόνα να επιλέγεται ο πρώτος κόμβος j ο οποίος μας δίνει μέγιστη τιμή. Σε περίπτωση που το κέρδος δεν είναι θετικό για τη μεταφορά του κόμβου i σε οποιαδήποτε κοινότητα των γειτόνων του (j) τότε ο κόμβος παραμένει στη ίδια κοινότητα. Η φάση αυτή επαναλαμβάνεται με τη σειρά για όλους τους κόμβους έως ότου να μη μπορεί να επιτευχθεί περεταίρω βελτίωση. Όταν έχει πλέον καθορίσει τις ενώσεις που πρέπει να γίνουν μεταξύ των κοινοτήτων, ο αλγόριθμος τερματίζει την πρώτη φάση.

Θα πρέπει να σημειωθεί ότι, τα αποτελέσματα του αλγορίθμου εξαρτώνται από τη σειρά με την οποία ελέγχονται οι κόμβοι. Μελέτες σε διάφορα δίκτυα έχουν υποδείξει πως η σειρά με την οποία ελέγχονται οι κόμβοι, δεν επιδρούν σημαντικά στα αποτελέσματα που μας δίνει ο αλγόριθμος. Ωστόσο, η σειρά ελέγχου των κόμβων μπορεί να επιδράσει σημαντικά στον χρόνο υπολογισμού (Blondel et al. 2008). Με αποτέλεσμα, να συνηθίζεται να γίνεται τυχαία η επιλογή της σειράς των κόμβων, αφού δεν έχει βρεθεί κανόνας ο οποίος να μας βοηθάει στην επιλογή αυτή. Για να υπολογίσει τη διαφορά στο modularity ο αλγόριθμος χρησιμοποιεί έναν τύπο αρκετά όμοιο με αυτό του Newman (5.7), ο οποίος έχει παρόμοιο υπολογιστικό κόστος, πιο συγκεκριμένα

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{5.8}$$

όπου το Σ_{in} είναι το άθροισμα των βαρών των ακμών εντός της κοινότητας C, το Σ_{tot} είναι το άθροισμα των βαρών των ακμών που συνδέονται με κόμβους της κοινότητας C. Το $k_{i,in}$ συμβολίζει το άθροισμα των βαρών των ακμών που είναι γείτονες με τον κόμβο i και ανήκουν στη κοινότητα C. Αντίθετα, το k_i είναι το άθροισμα των ακμών που είναι γείτονες με τον κόμβο i ασχέτως της κοινότητας που βρίσκονται. Έχοντας τερματίσει τη πρώτη φάση ο αλγόριθμος έχει αποθηκεύσει σε λίστες τις κοινότητες που έχουν δημιουργηθεί και από ποιους κόμβους αποτελούνται. Έχοντας την πληροφορία αυτή, συσσωματώνει τις κοινότητες σε υπερμεγέθη κόμβους (Σχήμα 5.4.c). Δηλαδή, οι νέοι κόμβοι που έχει δημιουργήσει, έχουν στο εσωτερικό τους μια ολόκληρη κοινότητα από κόμβους. Τα βάρη των συνδέσμων μεταξύ των νέων αυτών κόμβων ισούνται με το άθροισμα των βαρών μεταξύ των κοινοτήτων από τις οποίες αποτελούνται. Οι

ακμές μεταξύ των κοινοτήτων δημιουργούν βρόγχους[1] για τους νέους αυτούς κόμβους. Όταν τερματίζεται και το δεύτερο βήμα, τότε είναι πιθανό ο αλγόριθμος να επαναλάβει το πρώτο βήμα έχοντας ως κόμβους νέους υπερμεγέθη κόμβους που έχει σχηματίσει. Εάν βέβαια επαναληφθεί το πρώτο βήμα τότε θα περάσουμε ξανά στο δεύτερο όπου θα ενώσει τους 'υπερμεγέθεις κόμβους' σε ακόμα μεγαλύτερους, οι οποίοι θα αποτελούνται από ακόμα περισσότερους κόμβους του αρχικού δικτύου. Το ερώτημα πλέον φτάνει στο πότε τερματίζει ο αλγόριθμος. Μια προφανής επιλογή είναι, όταν έχουν πλέον ενωθεί όλοι οι κόμβοι και έχει δημιουργηθεί μια μόνο κοινότητα, αποτελούμενη από όλο το δίκτυο. Προφανώς, κάτι τέτοιο δεν συνεισφέρει σε καμιά είδους έρευνα και δεν επιλέγουμε να τερματίζεται ποτέ με αυτόν το τρόπο ο αλγόριθμος. Για το λόγο αυτό χρησιμοποιούμε ένα σημείο τερματισμού, δηλαδή επιλέγουμε μια τιμή ε και ο αλγόριθμος θα σταματάει αν $\Delta Q < \varepsilon$ αφού πλέον, το να ενώσουμε κόμβους (κοινότητες) μεταξύ τους δεν θα μας δίνει καμιά ουσιαστική διαφορά στο πόσο ικανοποιητικά έχει διαχωριστεί το δίκτυο σε κοινότητες. Όταν το modularity αυξάνεται ελάχιστα κατά τη συνένωση κοινοτήτων, σημαίνει πως η ένωση δεν είναι πρακτικά καλή και ότι οι κοινότητες αυτές δεν είναι αρκετά όμοιες για να θεωρηθούν ως μία. Το υπολογιστικό κόστος που έχει ο αλγόριθμος, δεν μπορεί να υπολογιστεί με ακρίβεια, αλλά από έρευνες που έχουν γίνει φαίνεται να τρέχει σε χρόνο $O(nln(n))$.



Σχήμα 5.4. Σχηματική αναπαράσταση βημάτων του αλγόριθμου Louvain. (Traag, Waltman and Van Eck, 2019)

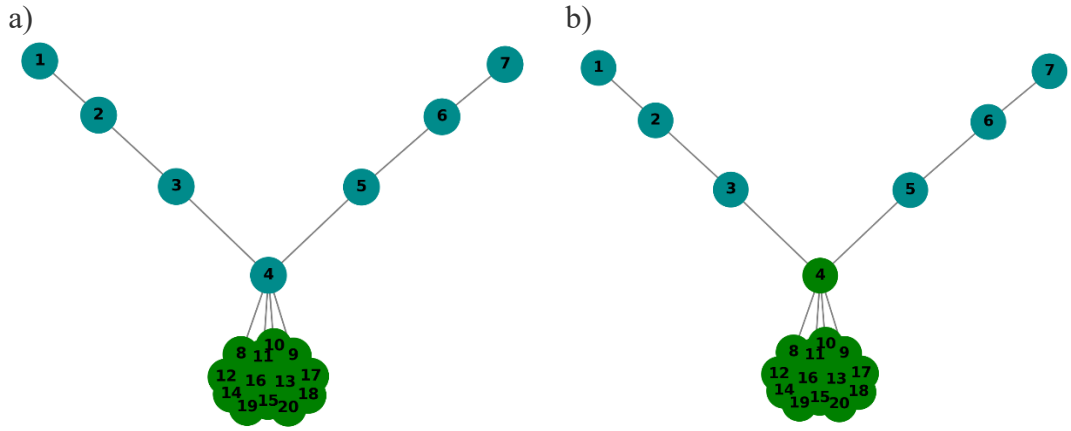
[1] Βρόγχος στη θεωρία γράφων θεωρείται μια ακμή που ενώνει έναν κόμβο με τον εαυτό του.

```
function Louvain (Graph G, Partition P)
    Do
        P ← MoveNodes(G,P)                ▸Μετακίνησε τους κόμβους μεταξύ κοινοτήτων
        done ← |P| = |V(G)|               ▸Τερμάτισε όταν όλες οι κοινότητες εμπεριέχουν ένα μόνο κόμβο
        if not done then
            G ← AggregateGraph (G, P)     ▸Δημιούργησε ένα συσσωματωμένο δίκτυο βάσει της διαμέρισης P
            P ← SingletonPartion(G)       ▸Ανάθεσε κάθε κόμβο στο συσσωρευμένο δίκτυο σε ξεχωριστή κοινότητα
        end if
    while not done
    return flat*(P)
end function
function MoveNodes (Graph G, Partition P)
    Do
        Q_old = Q(P)
        for u ϵ V(G) do                   ▸Επισκέψου όλους τους κόμβους με τυχαία σειρά
            C' ← arg max_{C∈P∪∅} ΔQ_P(u → C)   ▸Καθορίζει τη καλύτερη κοινότητα για τον κόμβο u
            if ΔQ_P(u → C) > 0 then       ▸Εκτέλεσε μόνο θετικές μετακινήσεις κόμβων
                u → C'                    ▸Μετακίνησε τον κόμβο u στη κοινότητα C
            end if
        end for
    while Q(P) > Q_old                    ▸Συνέχισε μέχρι να μην μπορούν να μετακινηθούν άλλοι κόμβοι
    return P
end function

function AggregateGraph
    V ← P                                 ▸ Οι κοινότητες μετατρέπονται σε υπερμεγέθη κόμβους στο συσσωρευμένο δίκτυο
    E← {(C, D)|(u, v) ϵE(G), u ϵC ϵP, v ϵD ϵP}
    return Graph(V, E)
end function

function SingletonPartion (Graph G)
    return {{u}|uϵV(G)}                   ▸Θεώρησε κάθε κόμβο ως ξεχωριστή κοινότητα
end function
```

Αλγόριθμος 5.2. Louvain (Traag, Waltman and Van Eck, 2019)

5.3.4 Αλγόριθμος Leiden

Ένα πρόβλημα που μπορεί να δημιουργηθεί από τον αλγόριθμο Louvain, είναι ότι μπορεί να δημιουργήσει κοινότητες που είναι εσωτερικά αποσυνδεδεμένες. Δηλαδή, θα έχουμε κόμβους που ανήκουν στην ίδια κοινότητα αλλά δεν θα υπάρχει μονοπάτι που να τους ενώνει και στο οποίο θα ανήκουν μόνο κόμβοι της κοινότητας αυτής.



Σχήμα 5.5. Αποσυνδεδεμένες κοινότητες.

Το Σχήμα 5.5 είναι ένα παράδειγμα που αντιπροσωπεύει ένα γενικότερο πρόβλημα που μπορεί να παράξει ο αλγόριθμος Louvain. Αν ο αλγόριθμος ελέγξει πρώτα τους κόμβους 3 και 5 θα τους τοποθετήσει στην ίδια κοινότητα με αυτή του κόμβου 4 και έτσι θα έχουμε τις κοινότητες του Σχήματος 5.5 (a). Όταν ελέγξει τον κόμβο 4 θα τον μεταφέρει σε άλλη κοινότητα και έτσι το δίκτυο θα είναι, όπως στο Σχήμα 5.5 (b). Το αποτέλεσμα που έχουμε είναι οι κόμβοι 1-3 και 5-7 να ανήκουν στην ίδια κοινότητα, όμως δεν υπάρχει μονοπάτι που ενώνει αυτούς του κόμβους μεταξύ τους και περνάει μόνο από κόμβους της κοινότητας αυτής. Παραδείγματος χάριν για να μεταφερθούμε από τον κόμβο 3 στον 5 πρέπει να περάσουμε αναγκαστικά από τον κόμβο 4, ο οποίος δεν ανήκει στην ίδια κοινότητα. Οι κόμβοι 1-3 και 5-7 θα μπορούσαν να ανήκουν στην ίδια κοινότητα ασχέτως που είναι αποσυνδεμένοι μεταξύ τους. Βέβαια, θα ήταν προτιμότερο να τους διαχωρίσουμε σε δύο κοινότητες, αλλά ο αλγόριθμος Louvain δεν εξετάζει αυτή τη πιθανότητα, αφού εξετάζει μεμονωμένες κινήσεις κόμβων. Όταν δεν μπορούν να μετακινηθούν άλλοι κόμβοι, ο αλγόριθμος ενώνει τους κόμβους της κοινότητας σε έναν. Όταν μια αποσυνδεδεμένη κοινότητα έχει μετατραπεί σε υπερμεγέθη κόμβο, δεν πρόκειται να διασπαστεί μελλοντικά. Όπου έχει ως

αποτέλεσμα να παραμένει αποσυνδεμένη στη τελική εξαγωγή των ομάδων, εκτός εάν συγχωνευθεί με κάποια άλλη κοινότητα που θα λειτουργήσει ως γέφυρα των αποσυνδεδεμένων κόμβων.

Για να αποφύγουν να έχουν κοινότητες που είναι αποσυνδεμένες μεταξύ τους, οι Traag, Waltman and Van Eck (2019) δημιούργησαν έναν νέο αλγόριθμο, τον Leiden (Αλγόριθμος 5.3). Ο αλγόριθμος Leiden είναι αρκετά παρόμοιος με αυτόν του Louvain, αφού εκτελεί την ίδια διαδικασία, με τη διαφορά ότι προσθέτει ένα ακόμα βήμα. Κατά τον αλγόριθμο Louvain όταν ένας κόμβος μεταφέρεται σε μια κοινότητα, δεν μπορεί να ξανά μετακινηθεί. Για το λόγο αυτό οι συγγραφείς έχουν προσθέσει το επιπλέον αυτό βήμα όπου επιτρέπει στους κόμβους να μετακινηθούν, ασχέτως που έχουν είδη καταχωρηθεί σε μια κοινότητα. Αρχικά, όπως και στον αλγόριθμο Louvain κάθε κόμβος θα θεωρηθεί μια ξεχωριστή κοινότητα (Σχήμα 5.6.a), με αποτέλεσμα το πλήθος των κοινοτήτων να είναι ίσο με το πλήθος κόμβων. Έπειτα, για κάθε κόμβο i ελέγχονται οι γειτονικοί κόμβοι j και υπολογίζεται το κέρδος που θα υπάρχει στη συνάρτηση ποιότητας, εάν αφαιρέσουμε τον κόμβο i από τη κοινότητα που ανήκει και τον τοποθετήσουμε στη κοινότητα του κόμβου j. Ο κόμβος i θα τοποθετηθεί στη κοινότητα του γειτονικού κόμβου j για την οποία θα έχουμε το υψηλότερο κέρδος (της ποιοτικής τιμής), με τη προϋπόθεση ότι το κέρδος είναι θετικό. Εάν ο αλγόριθμος θεωρήσει ότι δεν θα υπάρξει αρκετό κέρδος με τη μεταφορά του κόμβου i, τότε ο κόμβος θα παραμείνει στην αρχική του κοινότητα. Στο τέλος αυτού του βήματος, ο αλγόριθμος μας δίνει το σύνολο της βέλτιστης διαμέρισης $P = \{C_1, C_2, \dots\}$ (Σχήμα 5.6.b).

Μια ακόμη βασική διαφορά μεταξύ των δύο αλγορίθμων είναι η εφαρμογή διαφορετικής μεθόδου για τη μεταφορά των κόμβων από μια κοινότητα σε μια άλλη. Σε αντίθεση με τον αλγόριθμο Louvain ο αλγόριθμος Leiden κάνει χρήση μια ταχύτερης διαδικασίας στη μετακίνηση των κόμβων από μια κοινότητα σε μια άλλη. Η διαδικασία αυτή προτάθηκε από τους Bae et al. (2017). Κατά τη μετακίνηση κόμβων ο Louvain επισκέπτεται όλους τους κόμβους εντός του δικτύου, έως ότου να μην υπάρχουν πλέον μεταφορές που θα αυξήσουν την συνάρτηση ποιότητας. Με αυτό τον τρόπο όμως, ο αλγόριθμος επισκέπτεται κόμβους οι οποίοι δεν μπορούν να μετακινηθούν σε διαφορετική κοινότητα. Αντίθετα, κατά την αντίστοιχη φάση, ο αλγόριθμος Leiden επισκέπτεται μονάχα κόμβους των οποίων οι γείτονες έχουν μετακινηθεί, με αποτέλεσμα να μειώσει αρκετά το πλήθος των κόμβων το οποίο θα επισκεφτεί. Ουσιαστικά, κατά τη διαδικασία ταχείας μετακίνησης των κόμβων ο αλγόριθμος τοποθετεί τους κόμβους σε μια ουρά με τυχαία σειρά. Στη συνέχεια αφαιρεί τον πρώτο κόμβο από το μπροστινό μέρος της ουράς και καθορίζει

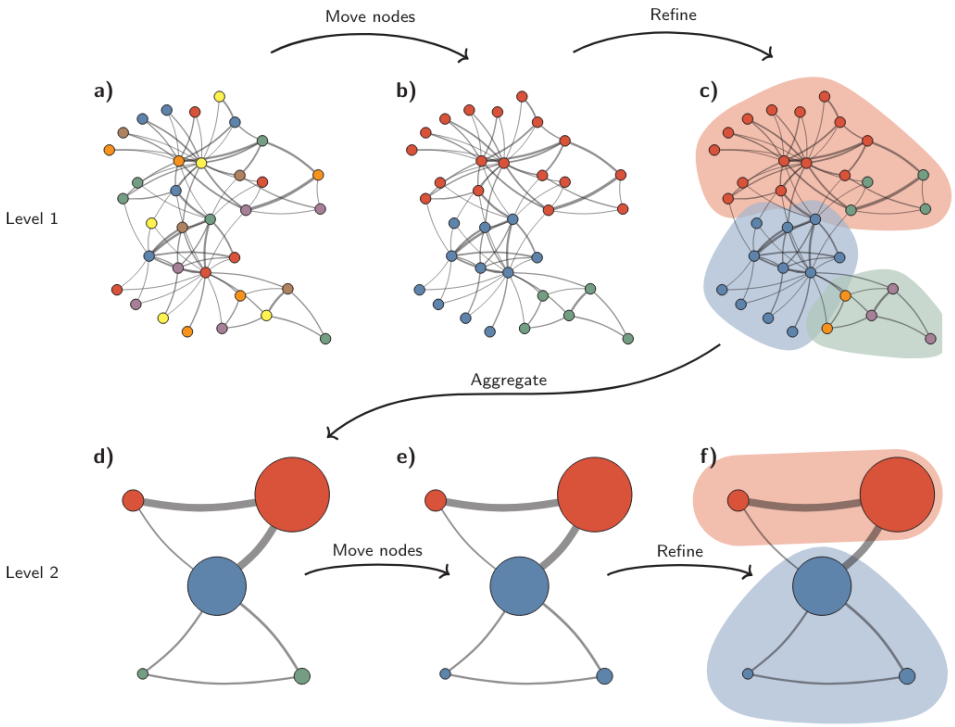
αν μπορεί να αυξηθεί η συνάρτηση ποιότητας με τη μετακίνηση του κόμβου αυτού σε μια άλλη κοινότητα. Εάν ο κόμβος μετακινηθεί σε διαφορετική κοινότητα, προσθέτουμε στο πίσω μέρος της ουράς όλους τους γείτονες του κόμβου αυτού οι οποίοι δεν ανήκουν στη νέα κοινότητα και δεν βρίσκονται ακόμα στην ουρά. Συνεχίζουμε να αφαιρούμε τους μπροστινούς κόμβους της ουράς έως ότου αδειάσει. Η πρώτη επίσκεψη όλων των κόμβων είναι ίδια για τους δύο αλγορίθμους. Ωστόσο, αφού όλοι οι κόμβοι έχουν επισκεφθεί μια φορά, ο αλγόριθμος Leiden θα ξανά επισκεφτεί κόμβους των οποίων οι γείτονες έχουν μεταφερθεί (μετά την επίσκεψη του κόμβου αυτού), ενώ αντίθετα ο αλγόριθμος Louvain επισκέπτεται ξανά όλους τους κόμβους. Με τον τρόπο αυτό ο Leiden υλοποιεί τη φάση της μετακίνησης αποτελεσματικότερα.

Το βήμα βελτίωσης που ακολουθεί αυτό της βελτιστοποίησης του modularity, αποσκοπεί στη μεγιστοποίηση της ποιότητας της ακρίβειας ανάθεσης των κόμβων σε κοινότητες. Κατ' ουσίαν στο βήμα αυτό, ο αλγόριθμος προσπαθεί να βρει μια καλύτερη διαμέριση $P_{refinement}$ (Σχήμα 5.6.c) της αρχικής δομής κοινοτήτων που έχει δημιουργήσει ο αλγόριθμος, κατά το πρώτο βήμα. Οι αρχικές κοινότητες (P) μπορεί να χωριστούν σε πολλές υπό-κοινότητες στο $P_{refinement}$. Σκοπός της δημιουργίας μιας νέας διαμέρισης είναι η βελτίωση της είδη υπάρχουσας, το οποίο σημαίνει πως ένας κόμβος δεν μπορεί να μετακινηθεί από μια κοινότητα C σε μια κοινότητα C' όπου και οι δύο ανήκουν στη διαμέριση P. Αντίθετα, κατά το στάδιο αυτό οι κοινότητες της αρχικής διαμέρισης, μελετιούνται ξεχωριστά. Ουσιαστικά ο αλγόριθμος για κάθε κοινότητα $C \epsilon P$ διασπά όλους τους κόμβους σε ξεχωριστές κοινότητες και μελετά όλους τους κόμβους, υπολογίζοντας αν θα ενωθούν ή όχι μεταξύ τους. Κατά τη φάση της βελτιστοποίησης οι κόμβοι δεν ενώνονται με άπληστο τρόπο, δηλαδή οι ενώσεις δεν αποδίδουν στη μέγιστη αύξηση της συνάρτησης ποιότητας. Η επιλογή της ένωσης γίνεται τυχαία, με την αύξηση της ποιοτικής συνάρτησης να χρησιμοποιείται ως παράγοντας, για τη τυχαία αυτή επιλογή. Όσο μεγαλύτερη η αύξηση της συνάρτησης ποιότητας, τόσο μεγαλύτερη η πιθανότητα να επιλεγεί μια κοινότητα. Ο βαθμός τυχαιότητας στην επιλογή της κοινότητας βασίζεται σε μια παράμετρο $\theta > 0$. Η τυχαιότητα επιλογής, επιτρέπει μια εκτενέστερη εξερεύνηση μιας διαμέρισης. Στη περίπτωση όπου η ένωση δύο ομάδων θα μειώσει τη συνάρτηση ποιότητας, η πιθανότητα ένωσης των δύο κοινοτήτων είναι μηδενική. Σε αυτή τη φάση ο αλγόριθμος μπορεί να ενώσει κοινότητες που είναι καλά συνδεδεμένες μεταξύ τους, με σκοπό να αποφύγει τη δημιουργία αποσυνδεδεμένων

κοινοτήτων. Η πιθανότητα ένωσης δύο κοινοτήτων κατά τη φάση της βελτιστοποίησης δίνεται από τον τύπο

$$\Pr(C' = C) \approx \begin{cases} \exp\left(\frac{1}{\theta}\Delta Q\right), & \text{αν } \Delta Q > 0 \\ 0, & \text{αλλιώς.} \end{cases} \quad (5.9)$$

Κατά το τελευταίο βήμα, ο αλγόριθμος δημιουργεί τους υπερμεγέθη (Σχήμα 5.6.d). κόμβους βάσει του συνόλου που έχει, από το βήμα βελτίωσης ($P_{refinement}$) και όχι από αυτό του πρώτου βήματος (P). Με το να δημιουργεί τις κοινότητες βάσει της διαμέρισης $P_{refinement}$ αντί της αρχικής P, ο αλγόριθμος Leiden αποκτά μια ευελιξία στην αναγνώριση και δημιουργία ποιοτικών διαμερίσεων. Στη πραγματικότητα, με την κατάλληλη εφαρμογή της φάσης βελτίωσης, μπορούν να δοθούν αρκετές ικανοποιητικές διασφαλίσεις για τις διαμερίσεις που παράγονται από τον αλγόριθμο.



Σχήμα 5.6. Σχηματική αναπαράσταση βημάτων του αλγόριθμου Leiden (Traag, Waltman and Van Eck,2019).

Λόγω των περιορισμών που έχει η συνάρτηση modularity (Fortunato and Barthelemy, 2007) οι συγγραφείς χρησιμοποιούν το Constant Potts Model (CPM) (Traag, Van Dooren and Nesterov,

2011) ως ποιοτικό μέτρο για την αξιολόγηση του αλγορίθμου. Το CPM ξεπερνά κάποιους από τους περιορισμούς του modularity. Το CPM ορίζεται ως

$$H = \sum_{C} \left[e_c - \gamma \binom{n_C}{2} \right] \tag{5.10}$$

όπου n_C συμβολίζει το πλήθος των κόμβων της κοινότητας C. Η παράμετρος ανάλυσης γ χρησιμοποιείται ως όριο όπου οι κοινότητες θα πρέπει να έχουν πυκνότητα τουλάχιστον ίση με γ εντός της κοινότητας και το πολύ ίση με γ μεταξύ κοινοτήτων. Όσο μεγαλύτερη είναι η παράμετρος ανάλυσης, τόσο περισσότερες κοινότητες αναμένουμε να μας δώσει ο αλγόριθμος. Ο αλγόριθμος Leiden έχει ένα παραπάνω βήμα από αυτό του Louvain. Βέβαια, λόγω αυτού του βήματος, συνήθως θέλει λιγότερες επαναλήψεις για να βρει τη βέλτιστη κοινοτική δομή. Έπιπλέον, ο αλγόριθμος χρησιμοποιεί τεχνικές οι οποίες βελτιώνουν συμαντικά τη ταχυτητα του σε σύγκριση με αυτή του Louvain (πχ στο βήμα μετακίνησης κόμβων). Λόγω των παραπάνω ο αλγόριθμος τρέχει σε ταχύτερο χρόνο, ο οποίος όμως δεν μπορεί να υπολογιστεί ακριβώς. Βάσει των εφαρμογών που πραγματοποίησαν οι συγγραφείς, διαπίστωσαν πως ο αλγόριθμος μπορεί να είναι μέχρι και είκοσι φορές ταχύτερος από αυτόν του Louvain (Traag, Van Dooren and Nesterov, 2011).

```
1:  function Leiden (Graph G ,Partition P)
2:      do
3:          P ← MoveNodesFast(G, P)          ▸Μετακίνησε τους κόμβους μεταξύ κοινοτήτων
4:          done ← |P| = |V(G)|              ▸Τερμάτισε όταν όλες οι κοινότητες εμπεριέχουν ένα μόνο κόμβο
5:          if not done then
6:              P_refined ← RefinePartition(G, P)   ▸Βελτίωσε τη διαμέριση P
7:              G ← AggregateGraph (G, P_refined)   ▸Δημιούργησε συσσωμάτωμα της διαμέρισης P_refined
8:              P ← {{u|u ⊆ C, uϵV(G)}|CϵP}         ▸Διατήρησε τη διαμέριση P
9:          end if
10:     while not done
11:     return flat*(P)
12: end function
```

13: **function** MoveNodesFast($Graph$ G, Partition P)
14: $Q \leftarrow$ Queue$(V(G))$ ▸Βεβαιώσου ότι θα επισκεφτούν όλοι οι κόμβοι (με τυχαία επιλογή)
15: **do**
16: $u \leftarrow Q$.remove() ▸Αποφάσισε ποιον κόμβο θα επισκεφτείς στο επόμενο βήμα
17: $C' \leftarrow arg\ max_{C \epsilon P \cup \emptyset} \Delta H_P(u \rightarrow C)$ ▸Καθορίζει τη καλύτερη κοινότητα για τον κόμβο u
18: **if** $\Delta H_P(u \rightarrow C') > 0$ **then** ▸Εκτέλεσε μόνο αυστηρά θετικές κινήσεις κόμβων
19: $u \rightarrow C'$ ▸Μετακίνησε τον κόμβο u στη κοινότητα C'
20: $N \leftarrow \{u | (u, v)\ \epsilon E(G), u \notin C'\}$ ▸Βρες τους γείτονες του u που δεν ανήκουν στη κοινότητα C'
21: Q.add$(N - Q)$ ▸Βεβαιώσου ότι θα επισκεφτούν όλοι οι γείτονες
22: **end if**
23: **while** $Q \neq \emptyset$ ▸Συνέχισε μέχρι να μην υπάρχουν άλλοι κόμβοι για να επισκεφτούν
24: **return** P
25: **end function**

26: **function** RefinePartion(Graph G, Partition P)
27: $P_{refined} \leftarrow$ SingletonPartition(P) ▸Θεώρησε κάθε κόμβο ως ξεχωριστή κοινότητα
28: **for** $C \epsilon P$ **do**
29: $P_{refined} \leftarrow$ MergeNodesSubset$(G, P_{refined}, C)$
30: **end for**
31: **return** $P_{refined}$
32: **end function**

33: **function** MergeNodesSubset(Graph G, Partition P, Subset S)
34: $R = \{u |\ u\ \epsilon\ S, E(u, S - u) \geq \gamma \|u\| \cdot (\|S\| - \|C\|)\}$
35: **for** $u\ \epsilon\ R$ **do** ▸Επισκέψου όλους τους κόμβους με τυχαία σειρά
36: **if** u in singleton community **then**
37: $T \leftarrow \{C | C \epsilon P, C \subseteq S, E(C, S - C) \geq \gamma \|C\| \cdot (\|S\| - \|C\|)\}$ ▸Μόνο καλά συνδεδεμένες κοινότητες
$Pr(C' = C) \sim \begin{cases} exp\left(\frac{1}{\theta} \Delta H_P(u \rightarrow C)\right), για\ C \epsilon T \\ 0 \qquad\quad , διαφορετικά \end{cases}$ ▸Διάλεξε τυχαία κοινότητα C'
38:
39: $u \rightarrow C'$ ▸Μετέφερε τον κόμβο u στη κοινότητα C'
40: **end if**
41: **end for**
42: **return** P
43: **end function**

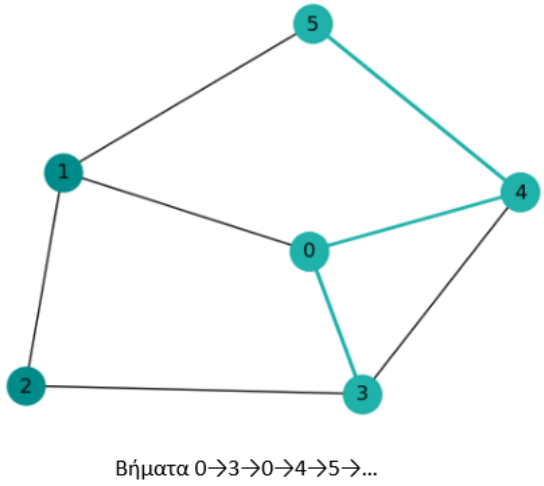
```
function AggregateGraph(Graph G, Partition P)
    V ← P                          ▸ Οι κοινότητες μετατρέπονται σε υπερμεγέθη κόμβους στο συσσωρευμένο δίκτυο
    E ← {(C, D)|(u, v) ϵE(G), u ϵC ϵP, v ϵD ϵP
    return Graph (V, E)
end function

function SingletonPartition(Graph G)
    return {{u}|uϵV(G)}            ▸Θεώρησε κάθε κόμβο ως ξεχωριστή κοινότητα
end function
```

Αλγόριθμος 5.2. Leiden (Traag, Waltman and Van Eck, 2019)

5.4 Τυχαίοι περίπατοι

Οι τυχαίοι περίπατοι (γνωστοί και ως random walks) είναι θεμελιώδεις στοχαστικές διεργασίες που περιγράφουν την κίνηση μίας οντότητας (πχ. άτομο) καθώς αυτή κινείται εντός ενός δικτύου. Σε κάθε βήμα το άτομο που κινείται επιλέγει τυχαία την θέση στην οποία θα μετακινηθεί. Παρά την απλότητά τους, οι τυχαίοι περίπατοι παρουσιάζουν πολύπλοκες και ενδιαφέρουσες συμπεριφορές καθιστώντας τους χρήσιμους για διάφορες αλγοριθμικές διαδικασίες. Στην επιστήμη της ανάλυσης κοινωνικών δικτύων οι τυχαίοι περίπατοι χρησιμοποιούνται σε ένα ευρύ φάσμα αναλύσεων παρέχοντας πολύτιμες γνώσεις για τη συμπεριφορά των υπό μελέτη δικτύων που μπορούν να αναλυθούν χρησιμοποιώντας έννοιες όπως η τυχαιότητα. Σε αυτό το κεφάλαιο θα αναλύσουμε αλγορίθμους τυχαίων περιπάτων, για την εύρεση κοινοτήτων εντός ενός δικτύου.

Η ιδέα στην οποία βασίζονται οι τυχαίοι περίπατοι είναι ότι κατά τη μετακίνηση του, ένας περιπατητής θα τείνει να ¨ παγιδευτεί ¨ εντός μιας πυκνά ενωμένης περιοχής μέσα στον γράφο, την οποία μπορούμε να θεωρήσουμε ως μια κοινότητα. Αναλύοντας τα μοτίβα μετακίνησης και τη συχνότητα επισκέψεων σε διαφορετικές κορυφές που δημιουργεί το κάθε άτομο κατά τη διάρκεια της διαδικασίας, μπορούμε να βγάλουμε συμπεράσματα για τη δομή του δικτύου. Έχουν δημιουργηθεί ποικίλοι αλγόριθμοι βασισμένοι σε τυχαίους περιπάτους για την ανίχνευση κοινοτήτων εντός ενός δικτύου.



Σχήμα 5.7. Γραφική αναπαράσταση τυχαίου περιπάτου

5.4.1 Αλγόριθμος Walktrap

Ο Walktrap είναι ένας αλγόριθμος εύρεσης κοινοτήτων όπου πρακτικά προσπαθεί να εντοπίσει συνεκτικές ομάδες κόμβων για να τις κατηγοριοποιήσει ως κοινότητες. Ο αλγόριθμος παρουσιάστηκε από τους Pons and Latapy (2005) βασιζόμενος στην ιδέα ότι οι ένας τυχαίος περιπατητής τείνει να παγιδευτεί εντός μιας πυκνά συνδεδεμένης ομάδας σε ένα δίκτυο. Ο αλγόριθμος προσομοιώνει τυχαίους περιπάτους εντός ενός δικτύου όπου έπειτα εξετάζει τα μοτίβα που δημιουργήθηκαν. Βάσει της αξιολόγησης της ομοιότητας μεταξύ των κόμβων που έγινε κατά την προσομοίωση του τυχαίου περίπατου, ο Walktrap διαμερίζει το δίκτυο σε κοινότητες.

Ο Walktrap μπορεί να χρησιμοποιηθεί και διαχειριστεί ποικίλες δομές γράφου, όπου βέβαια ανάλογα με τη δομή του, μπορεί να μην έχουμε τα αναμενόμενα αποτελέσματα. Έστω ότι θέλουμε να μελετήσουμε ένα μη κατευθυνόμενο και χωρίς βάρη δίκτυο $G = (V, E)$. Όπως προαναφέραμε, σε κάθε βήμα το άτομο διαλέγει τυχαία και ισοπίθανα σε ποιόν από τους γειτονικούς κόμβους θα μεταφερθεί. Η ακολουθία των κόμβων που έχει επισκεφτεί είναι μια Μαρκοβιανή αλυσίδα (Venkatesaramani and Vorobeychik, 2018), όπου οι καταστάσεις αντιπροσωπεύονται από τις κορυφές του γράφου ενώ οι μεταβολές αντιπροσωπεύονται από τις ακμές. Η πιθανότητα μετακίνησης από ένα κόμβο i σε ένα κόμβο j (για $\{i, j\}\epsilon V$) ισούται με:

$$P_{ij} = \frac{A_{ij}}{d(i)}$$

όπου με A συμβολίζεται ο πίνακας γειτνίασης του γράφου και με $d(i)$ συμβολίζουμε το πλήθος των γειτόνων του κόμβου i. Με αυτό τον τρόπο μπορούμε να κατασκευάσουμε τον πίνακα P ο οποίος επανυπολογίζεται σε κάθε βήμα t του αλγορίθμου και εμπεριέχει την πιθανότητα μετακίνησης από ένα κόμβο i σε ένα κόμβο j. Έτσι θα συμβολίζουμε τη πιθανότητα μετακίνησης κατά το βήμα t με P_{ij}^t.

Οι συγγραφείς (Pons and Latapy, 2005) απέδειξαν πως

α. Η πιθανότητα ενός περιπατητή, να βρίσκεται στον κόμβο j μετά από άπειρα πλήθους βήματα, εξαρτάται μόνο από το βαθμό του κόμβου j και όχι από του αρχικού i. Πιο συγκεκριμένα

$$\lim_{t\to\infty} P_{ij}^t = \frac{d(j)}{\sum_{k\epsilon V} d(k)}, \qquad \forall i\ .$$

β. Η πιθανότητα μετάβασης από τον κόμβο i στον κόμβο j και αντίστοιχα από τον j στον i μέσω του τυχαίου περιπάτου σταθερού μήκους t, εξαρτάται από το πλήθος των γειτόνων των δύο κόμβων ($d(i)$ και $d(j)$). Πιο συγκεκριμένα. Ισχύει ότι

$$d(i)P_{ij}^t = d(j)P_{ji}^t, \qquad \forall i, \forall j.$$

Όπως προαναφέραμε ο αλγόριθμος κατατάσσει τους κόμβους στις διάφορες κοινότητες βάσει της ομοιότητάς τους. Για να υπολογίσουν την ομοιότητα των κόμβων οι συγγραφείς εισήγαγαν μια νέα μετρική r η οποία υπολογίζει την απόσταση μεταξύ των κόμβων. Η απόσταση μεταξύ κόμβων της ίδια κοινότητας πρέπει να είναι μικρή, ενώ αντίθετα αν δύο κόμβοι ανήκουν σε διαφορετικές κοινότητες οι απόσταση που έχουν μεταξύ τους πρέπει να είναι αρκετά μεγάλη. Η αποστάσεις υπολογίζονται από τις πληροφορίες που μας δίνονται από τη προσομοίωση τυχαίου περιπάτου, μήκους t, που εκτελεί ο αλγόριθμος όπως και από την πιθανότητα μετάβασης P_{ij}^t από ένα κόμβο i σε ένα κόμβο j. Το πλήθος των βημάτων t του τυχαίου περιπάτου θα πρέπει να είναι επαρκές για να παραχθούν αποτελέσματα ικανοποιητικά για τη δομή του δικτύου. Βέβαια το t δεν θα πρέπει να είναι πολύ μεγάλο για να αποφύγουμε την επίδραση που μπορεί να ασκήσει, ότι δηλαδή η πιθανότητα θα μετάβασης θα οφείλεται μόνο στον κόμβο j (βλέπε αποτέλεσμα α). Η απόσταση (όπως και η πιθανότητα μετάβασης) μεταξύ δύο κόμβων i και j επηρεάζεται άμεσα από το πλήθος

γειτόνων του j $(d(j))$. Μια ακόμη παρατήρηση που αναφέραν οι συγγραφείς είναι ότι δύο κόμβοι της ίδιας κοινότητας (i και j) τείνουν να αντιμετωπίζουν περίπου το ίδιο οποιονδήποτε άλλο κόμβο, δηλαδή περιμένουμε να ισχύει ότι $\forall z \epsilon V, P_{iz}^t \approx P_{jz}^t$. Η απόσταση που χρησιμοποιείται δεν είναι ίδια με τις αποστάσεις που ορίσαμε στο Κεφάλαιο 2, αλλά όρισαν μια νέα μετρική

$$r_{ij} = \sqrt{\sum_k \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P_{i\cdot}^t - D^{-\frac{1}{2}} P_{j\cdot}^t \right\|$$

όπου $k \epsilon V \setminus \{i, j\}$ και με $\|\cdot\|$ συμβολίζουμε την ευκλείδεια νόρμα του $\mathbb{R}^n$. Όπως μπορούμε να δούμε η απόσταση μεταξύ κόμβων εξαρτάται άμεσα από το πλήθος των βημάτων t, οπότε την απόσταση θα μπορούσαμε να τη συμβολίσουμε ως $r_{ij}(t)$. Εφόσον αναμένουμε η απόσταση μεταξύ δύο κόμβων της ίδιας κοινότητας να είναι μικρή, μπορούμε να θεωρήσουμε ότι και η πιθανότητα μετάβασης μεταξύ δύο κόμβων της ίδιας κοινότητας θα είναι μεγάλη. Βέβαια, αν βρούμε δύο κόμβους με μεγάλη πιθανότητα μετάβασης μεταξύ τους, δεν πρέπει να θεωρήσουμε ότι οι κόμβοι αυτοί ανήκουν στην ίδια κοινότητα. Η πιθανότητα μετάβασης μιας κοινότητας C σε έναν κόμβο j σε t βήματα συμβολίζεται και υπολογίζεται ως εξής

$$P_{Cj}^t = \frac{1}{|C|} \sum_{i \epsilon C} P_{ij}^t$$

το οποίο μας ορίζει το διάνυσμα $P_{C\cdot}^t$ το οποίο μας δίνει την δυνατότητα γενίκευσης των αποστάσεων μεταξύ δύο κοινοτήτων. Έστω δύο κοινότητες του δικτύου $C_1, C_2 \subset V$. Τότε η απόσταση $r_{C_1 C_2}$ μεταξύ των δύο αυτών κοινοτήτων, ορίζεται ως εξής

$$r_{C_1 C_2} = \sqrt{\sum_k \frac{(P_{C_1 k}^t - P_{C_2 k}^t)^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P_{C_1\cdot}^t - D^{-\frac{1}{2}} P_{C_2\cdot}^t \right\|.$$

Ο τρόπος με τον οποίο γίνεται η συσταδοποίηση είναι παρόμοια με αυτούς των ιεραρχικών μεθόδων, το οποίο θα αναλυθεί παρακάτω. Αρχικά, θα πρέπει να ορίσουμε τον τρόπο με τον οποίο επιλέγει ο αλγόριθμος ποιες συστάδες θα συγχωνεύει σε κάθε βήμα. Η κοινότητα με την οποία θα συγχωνευθεί ο υπό μελέτη κόμβος επιλέγεται από την συνάρτηση μέσω τετραγωνικών αποστάσεων, την οποία προσπαθεί συνεχώς να ελαχιστοποιήσει

$$\sigma_k = \frac{1}{n}\sum_{C\epsilon P_k}\sum_{i\epsilon C} r_{iC}^2$$

όπου με P_k συμβολίζουμε μια διαμέριση του γράφου, την οποία θα την αναλύσουμε παρακάτω. Η προσέγγιση αυτή είναι μια λαίμαργη τεχνική που προσπαθεί να λύσει το πρόβλημα μεγιστοποίησης του σ_k για κάθε k. Παρόμοια με την βελτιστοποίηση της συνάρτησης ποιότητας του modularity (βλέπε (5.3.2)) το πρόβλημα είναι και πάλι NP-Hard (Drineas et al. 2004). Δεδομένου ότι οι υπάρχοντες προσεγγιστικοί αλγόριθμοι έχουν εκθετικό χρόνο υπολογισμού ανάλογο με το πλήθος των συστάδων, δεν κρίνονται συμβατοί για τις απαιτήσεις που έχουν τα κοινωνικά δίκτυα, λόγου του τεράστιου όγκου τους. Για τον λόγω αυτόν, για κάθε δύο γειτονικές κοινότητες $\{C_1, C_2\}$, θα υπολογίζουμε τη διαφορά $\Delta_\sigma(C_1, C_2)$, του σ που θα υπήρχε αν ενώναμε τις δύο κοινότητες ($C_3 = C_1 \cup C_2$). Η συνάρτηση αυτή εξαρτάται μόνο από τους κόμβους των δύο υπό μελέτη κοινοτήτων και όχι από κάποιον άλλον κόμβο ή το βήμα k και δίνεται από τον τύπο

$$\Delta_\sigma(C_1, C_2) = \frac{1}{n}\left(\sum_{i\epsilon C_3} r_{iC_3}^2 - \sum_{i\epsilon C_1} r_{iC_1}^2 - \sum_{i\epsilon C_2} r_{iC_2}^2\right).$$

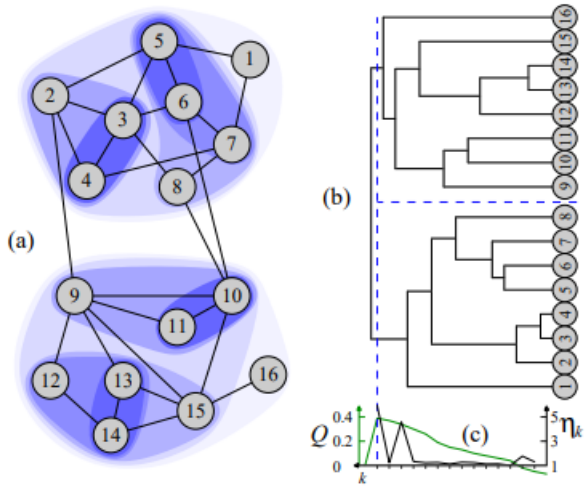
Με το τρόπο αυτό ενώνουμε τις δύο αυτές κοινότητες, για τις οποίες έχουμε την μικρότερη τιμή του Δ_σ. Ο αλγόριθμος ανανεώνει μόνο τις αποστάσεις μεταξύ των κοινοτήτων που συγχωνεύθηκαν και των γειτονικών τους, αφού όλες οι λοιπές αποστάσεις δεν πρέπει να επηρεαστούν από το γεγονός αυτό.

Ο αλγόριθμος σταματά όταν φτάσει σε μια διαμέριση P_n, στην οποία ανήκουν όλοι οι κόμβοι. Το ερώτημα που δημιουργείται είναι πως επιλέγεται η βέλτιστη διαμέριση. Οι συγγραφείς, για λύσουν αυτό το πρόβλημα χρησιμοποίησαν δύο κριτήρια στα οποία βασίζονται για την επιλογή της βέλτιστης διαμέρισης. Το πρώτο κριτήριο είναι η συνάρτηση ποιότητας Q (modularity). Ως βέλτιστη, θεωρείται η διαμέριση για τη οποία το Q παίρνει την μεγαλύτερη τιμή. Δεδομένου ότι το modularity δεν παράγει ικανοποιητικά αποτελέσματα στη περίπτωση που οι κοινότητες έχουν διαφορά μεγέθους (κάποιες κοινότητες αποτελούνται από πολύ περισσότερους κόμβους σε σχέση με κάποιες άλλες), οι συγγραφείς εισήγαγαν ένα νέο ποσοτικό κριτήριο, τη συνάρτηση η_t. Το κριτήριο αυτό βασίζεται στη συνάρτηση $\Delta\sigma$, για την οποία αναμένουν μεγάλες τιμές όταν ενώνουμε κοινότητες οι οποίες είναι ανόμοιες μεταξύ τους. Αντίθετα, αν δύο κοινότητες είναι αρκετά όμοιες ώστε να μπορούμε να θεωρήσουμε ότι πρέπει να συγχωνευθούν, τότε αναμένουμε

μικρή τιμή $\Delta\sigma$. Αυτό ήταν το κίνητρο που ώθησε τους συγγραφείς να δημιουργήσουν τη συνάρτηση

$$\eta_t = \frac{\Delta\sigma_t}{\Delta\sigma_{t-1}} = \frac{\sigma_{t+1} - \sigma_t}{\sigma_t - \sigma_{t-1}}$$

Όπως προαναφέραμε η τιμή του $\Delta\sigma$ μπορεί να μας οδηγήσει σε συμπεράσματα για το αν η συγχώνευση δύο κοινοτήτων σε κάθε βήμα ήταν ουσιαστική ή όχι. Αν το $\Delta\sigma_{t-1}$ είναι μικρό τότε θεωρούμε ότι η συγχώνευση των δύο κοινοτήτων στο βήμα $t-1$ ήταν χρήσιμη διότι οι δύο αυτές κοινότητες ήταν αρκετά όμοιες. Αν στο βήμα t η τιμή $\Delta\sigma_t$ ήταν μεγάλη τότε θεωρούμε πως οι δύο κοινότητες που συγχωνεύτηκαν δεν ήταν αρκετά όμοιες. Βάσει των παραπάνω, θεωρούμε ότι καλύτερη θα είναι η διαμέριση για την οποία το η_t παίρνει την μέγιστη τιμή.



Σχήμα 5.8. Γραφική αναπαράσταση του Walktrap (Pons and Latapy, 2005).

Στο Σχήμα 5.8.a έχουμε ένα παράδειγμα κοινοτικής δομής όπως αυτή ανιχνεύθηκε από τυχαίο περίπατο μήκους $t = 3$. Όσο πιο έντονο το χρώμα της κοινότητας, τόσο πιο νωρίς δημιουργήθηκε κατά τη διάρκεια του αλγορίθμου. Το Σχήμα 5.8.b παρουσιάζει ένα δενδρόγραμμα αναπαράστασης όλων των φάσεων του αλγορίθμου. Τέλος, το Σχήμα 5.8.c περιέχει ένα γράφημα

που υποδεικνύει βάση και των δύο κριτηρίων (Q και η_κ), πως η καλύτερη διαμέριση είναι όταν έχουμε δύο κοινότητες.

Ο αλγόριθμος Walktrap ανάγει το πρόβλημα εύρεσης κοινοτήτων σε ένα πρόβλημα συσταδοποίησης. Βασισμένοι στη μέθοδο του Ward (Duch and Arenas, 2005), οι συγγραφείς παρουσιάζουν μια συσσωρευτική λύση του προβλήματος. Ο αλγόριθμος ξεκινά με τη διαμέριση $P_1 = \{\{u\}, u\epsilon V\}$ του γράφου σε n κοινότητες, οι οποίες αποτελούνται από ένα κόμβο και τον υπολογισμό των αποστάσεων μεταξύ όλων των κορυφών. Έπειτα ο αλγόριθμος επαναλαμβάνει μια διαδικασία σε κάθε βήμα t, έως ότου φτάσει στη διαμέριση $P_n = \{V\}$ όπου όλοι οι κόμβοι ανήκουν στην ίδια κοινότητα:

- Επιλέγει δύο κοινότητες βασισμένο στο κριτήριο $\Delta\sigma$
- Ενώνει τις δύο κοινότητες σε μια καινούργια $C_3 = C_1 \cup C_2$, όπου και δημιουργεί μια νέα διαμέριση $P_{t+1} = (P_t \backslash \{C_1, C_2\} \cup C_3)$
- Ανανεώνει τις αποστάσεις μεταξύ των κοινοτήτων (αυτών που ενώθηκαν και των γειτονικών τους.

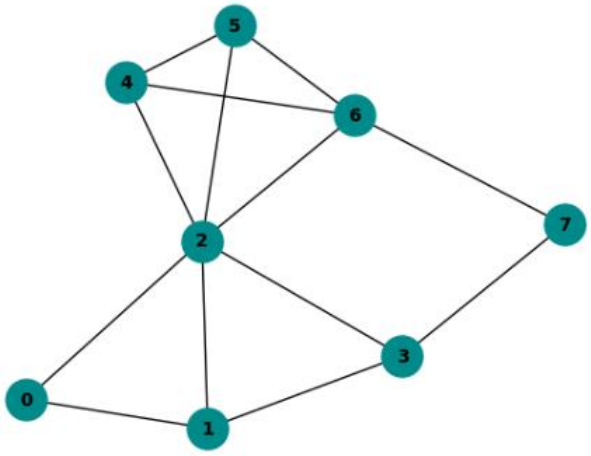
Σε κάθε βήμα ουσιαστικά, δημιουργείτε μια διαμέριση P_t η οποία μπορεί να παρουσιαστεί σε μια ιεραρχική κοινοτική δομή, όπως ένα δενδρόγραμμα. Το γεγονός ότι οι αποστάσεις μπορούν εύκολα να υπολογιστούν σε κάθε βήμα και δεν χρειάζεται να υπολογίζουμε όλες τις αποστάσεις παρα μόνο αυτές των οποίων οι γειτονικοί κόμβοι έχουν αλλάξει δομή, καταστά τον αλγόριθμο πολύ αποδοτικό. Ο χρόνος που χρειάζεται ο αλγόριθμος για να τρέξει είναι $O(mn^2)$.

5.5 Αλληλοκαλυπτόμενες κοινότητες

Ένα άτομο δεν περιορίζεται πάντα σε μια ομάδα αλλά μπορεί να ανήκει σε διάφορες ομάδες ατόμων. Για παράδειγμα, θα μπορούσε κάποιος να ανήκει σε μια αθλητική ομάδα, σε μια πολιτική ομάδα και ούτω καθεξής. Επίσης κάποιες ομάδες, θα μπορούσαν να χωριστούν σε μικρότερες υπό-ομάδες. Ως παράδειγμα θα μπορούσαμε να δώσουμε την επιστήμη η οποία χωρίζεται στα μαθηματικά, τη φυσική, τη ψυχολογία και άλλα. Ένα άλλο παράδειγμα θα μπορούσε να θεωρηθεί η ομάδα του αθλητισμού, όπου ως υπό ομάδες θα θεωρούντουσαν όλα τα αθλήματα. Οι προηγούμενες ενότητες αυτού του κεφαλαίου ανέλυσαν αλγορίθμους οι οποίοι κατατάσσουν τα άτομα σε μια μόνο ομάδα. Στη πραγματικότητα όμως, οι κόμβοι των δικτύων που μελετάμε, μπορεί να ανήκουν σε περισσότερες από μια κοινότητες. Για παράδειγμα, ένα άτομο στο Facebook μπορεί

να ανήκει σε μια αθλητική και μια πολιτική ομάδα. Κοινότητες οι οποίες εμπεριέχουν κοινά άτομα (κόμβους) εντός του δικτύου, τις ονομάσουμε αλληλοκαλυπτόμενες κοινότητες (overlapping communities). Μια από τις πιο δημοφιλής μεθόδους για εύρεση αλληλοκαλυπτόμενων κοινοτήτων είναι η μέθοδος Clique Percolation η οποία δημιουργήθηκε από τους (Palla et al. 2005) και θα αναλυθεί σε αυτή την ενότητα.

Όπως αναφέραμε στο δεύτερο κεφάλαιο μια k-κλίκα είναι ένας πλήρης (όλοι οι κόμβοι ενώνονται μεταξύ τους) υπό-γράφος ο οποίος εμπεριέχει k κόμβους. Δύο k-κλίκες ονομάζονται γειτονικές, αν έχουν $k-1$ κοινούς κόμβους, όπου ουσιαστικά μόνο ένας κόμβος διαφέρει. Κατά την μέθοδο Clique percolation, η κοινότητα του δικτύου ορίζεται ως το μέγιστο σύνολο $k-1$ γειτονικών κλικών. Ουσιαστικά ο αλγόριθμος βασίζεται στην υπόθεση ότι μία κοινότητα αποτελείται από αλλυλοκαλυπτόμενα σύνολα, πλήρως συνδεδεμένων υπογράφων και ανιχνεύει τις κοινότητες αναζητώντας κλίκες οι οποίες είναι γειτονικές μεταξύ τους. Το μέγιστο πλήθος πιθανών k-κλικών είναι $k(k-1)/2$. Βέβαια σε πραγματικά σύνολα δεδομένων θα υπάρχει θόρυβος και έτσι δεν περιμένουμε να βρούμε τόσο μεγάλο πλήθος k-κλικών. Αρχικά ανιχνεύει όλες τις κλίκες μέγιστου μεγέθους, στις οποίες ανήκουν οι κόμβοι του δικτύου. Ο αλγόριθμος που χρησιμοποιείται για να βρει τις κλίκες εντός του δικτύου, έχει προταθεί από τους Bron and Kerbosch (1973) και τρέχει σε μέγιστο χρόνο $O(3^{n/3})$ (Tomita, Tanaka and Takahashi, 2006).



Σχήμα 5.9. Παράδειγμα εύρεσης k-κλικών.

Στο παράδειγμα του Σχήματος 5.9 όλοι οι κόμβοι ανήκουν σε κλίκες 3 κόμβων, βέβαια κάποιοι κόμβοι ανήκουν σε μεγαλύτερα σύνολα (π.χ. οι κόμβοι 2,4,5 και 6 σχηματίζουν κλίκα μεγέθους 4). Έτσι ο αλγόριθμος αναγνωρίζει ως κλίκες τα παρακάτω σύνολα

a:(0,1,2), b:(1,2,3), c:(2,4,5,6), d:(3,7) και e:(6,7).

Έπειτα ο αλγόριθμος ενώνει σε κοινότητες τις κλίκες (σύνολα) οι οποίες έχουν κοινούς $k-1$ κόμβους. Έτσι, οι κοινότητες του παραπάνω παραδείγματος (Σχήμα 5.9) καθορίζονται ως εξής

$$C_1 = \{0,1,2,3\} \text{ και } C_2 = \{2,4,5,6\}$$

Σε δίκτυα με βάρη, μια k-κλίκα C_k θεωρείται ως πλήρες υπό-γράφος, όταν ο μέσος των βαρών των ακμών του (που ανήκουν στην κλίκα C_k), ξεπερνά ένα κατώτατο όριο έντασης I (Farkas et al. 2007), το οποίο υπολογίζεται από τον τύπο

$$I(C_k) = \left(\prod\nolimits_{i<j,i,j\epsilon C_k} w_{ij}\right)^{2/k(k-1)}$$

όπου με w_{ij} συμβολίζουμε το βάρος της ακμής μεταξύ των κόμβων i και j. Μπορεί να παρατηρήσει κάποιος πως η κύρια διαφορά της μεθόδου με βάρη, είναι το όριο έντασης που πρέπει να αναλύσουμε. Έστω ότι έχουμε ένα συγκεκριμένο μέγεθος k για τις k-κλίκες και βάρη για τα οποία ισχύει $w_1 \geq w_2 \geq \ldots \geq w_m$ (όπου m το πλήθος των ακμών του δικτύου). Αν $I > w_1$, τότε το όριο έντασης όλων των k-κλικών θα είναι μικρότερο από το όριο έντασης και έτσι ο αλγόριθμος δεν θα μπορεί να βρει καμιά κοινότητα. Αντιθέτως τότε αν, $I < w_1$ όλες οι k-κλίκες θα πληρούν το κριτήριο του ορίου έντασης. Ως αποτέλεσμα, θα έχουμε ένα πολύ μεγάλο σύνολο κόμβων να ανήκουν σε μια μόνο κοινότητα

ΚΕΦΑΛΑΙΟ 6

Εφαρμογή αλγορίθμων

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν τα αποτελέσματα ανάλυσης αλγόριθμων για την καταμέτρηση τριγώνων και για την εύρεση κοινοτήτων σε ποικίλα δίκτυα γράφων. Θα παρουσιάσουμε και θα αναλύσουμε διάφορα είδη συνθετικών (τεχνητών) δικτύων, στα οποία θα αξιολογήσουμε αρχικά τους αλγόριθμους. Έπειτα, θα παρουσιάσουμε αποτελέσματα για μερικά πραγματικά σύνολα δεδομένων τα οποία συλλέχθηκαν από την ιστοσελίδα του SNAP (Stanford Large Network Collection) τα οποία δίνονται ελεύθερα για πειραματική χρήση (Jure, 2014). Όλοι οι αλγόριθμοι υλοποιήθηκαν στην Python και τα αποτελέσματα που θα παρουσιαστούν προέρχονται από την υλοποίηση τους σε Jupyter notebook (Luyver et al. 2016) εφαρμογή, η οποία μας επιτρέπει να τρέξουμε κώδικες σε γλώσσα Python. Τα χαρακτηριστικά του υπολογιστή που χρησιμοποιήθηκαν για την εκτέλεση των αποτελεσμάτων είναι τα εξής:

- **Επεξεργαστής:** AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
- **Αποθηκευτικός χώρος RAM:** 16GB
- **Λειτουργικό:** Windows 11 Home 64-bit.

6.2 Συνθετικά δίκτυα γράφων

Όπως είδαμε στο προηγούμενο κεφάλαιο υπάρχουν πολλοί αλγόριθμοι για την εύρεση κοινοτήτων εντός ενός δικτύου. Ωστόσο, η εύρεση κοινοτήτων ως πρόβλημα είναι παρόμοιο με τις τεχνικές συσταδοποίησης και όχι με τεχνικές ομαδοποίησης. Αυτό σημαίνει πως τα άτομα του δικτύου δεν ανήκουν σε γνωστές εκ των προτέρων ομάδες, με αποτέλεσμα να μη μπορούμε να αξιολογήσουμε τα αποτελέσματα των αλγορίθμων με απόλυτη σιγουριά αλλά μόνο βασισμένοι σε δείκτες όπως αυτοί του modularity. Για να μπορέσουμε να αξιολογήσουμε καλύτερα τους διάφορους αλγόριθμους εύρεσης κοινοτήτων, μπορούμε να τους εφαρμόσουμε σε δίκτυα με ήδη

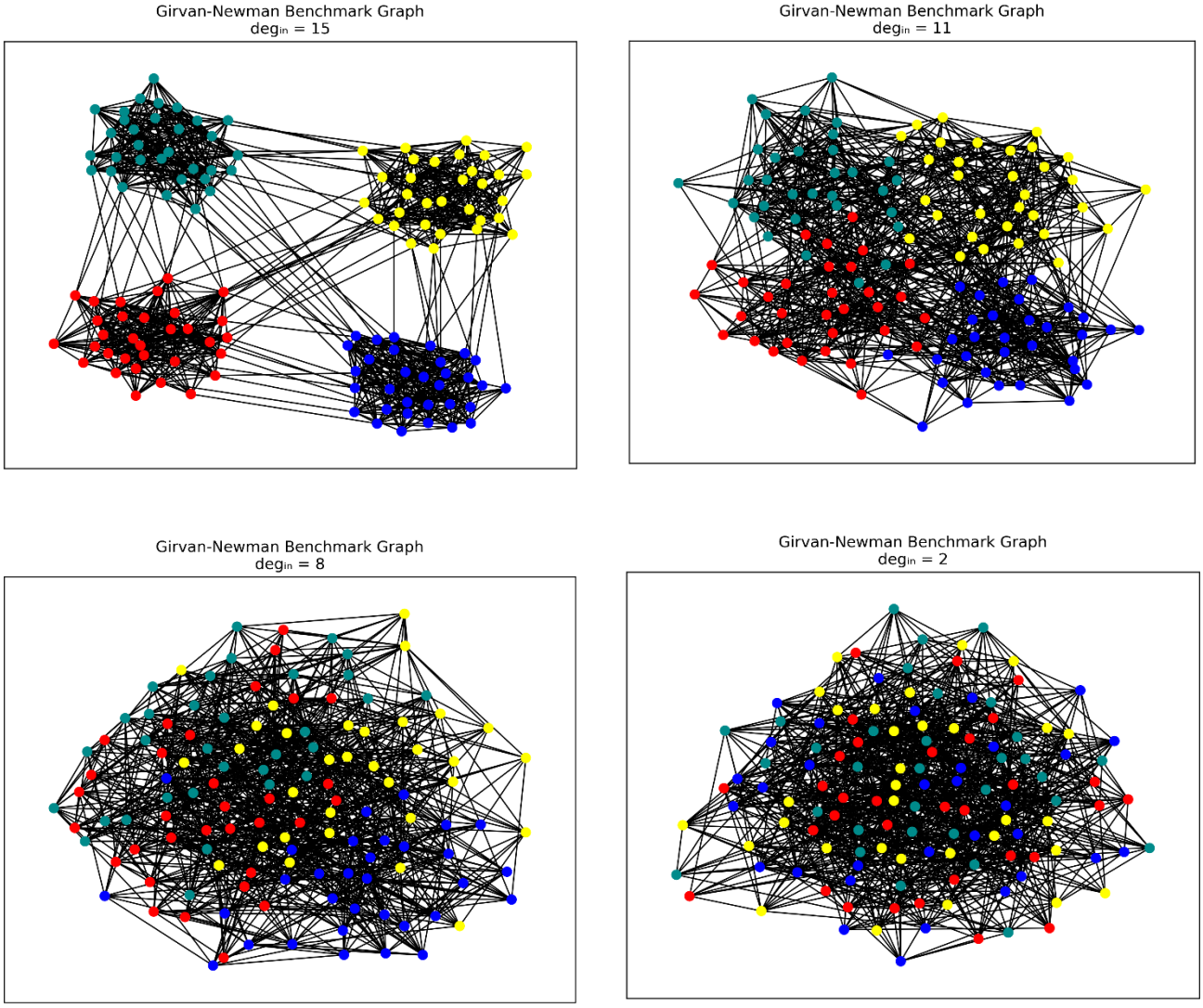
γνωστές κοινότητες και να συγκρίνουμε τα αποτελέσματα των κοινοτήτων που βρήκαμε, με τις πραγματικές κοινότητές τους.

Σε πραγματικά δεδομένα οι κοινότητες δεν είναι εξ αρχής γνωστές. Για τον λόγο αυτό έχουν δημιουργηθεί αλγόριθμοι παραγωγής συνθετικών δικτύων (benchmark networks) στων οποίων οι κοινότητες είναι προκαθορισμένες κατά την παραγωγή τους. Έτσι μπορεί κάποιος να συγκρίνει ποσοτικά τα αποτελέσματα της κάθε μεθόδου και να μπορεί ευκολότερα να αξιολογήσει τα αποτελέσματα των μεθόδων αυτών. Βέβαια, αν κάποιος από τους αλγορίθμους δεν μπορέσει να επιτύχει να θέσει στις σωστές κοινότητες όλους τους του δικτύου, δεν σημαίνει πως ο αλγόριθμος δεν είναι χρήσιμος, αλλά πιθανώς να αποτυγχάνει να επιτευχθεί σωστή ανίχνευση των κοινοτήτων σε δίκτυα με παρόμοια δομή όπως αυτή του παράγωγου μοντέλου του δικτύου αναφοράς. Αν βέβαια ο αλγόριθμος μπορεί να αποφέρει πολύ ικανοποιητικά αποτελέσματα σε πολλαπλά δίκτυα αναφοράς (με διαφορετικές κοινοτικές δομές), τότε θεωρούμε πως ο αλγόριθμος μπορεί να χρησιμοποιηθεί σε διάφορα πραγματικά μοντέλα, όπου αναμένουμε να παράγει ικανοποιητικά αποτελέσματα.

α. Δίκτυα Αναφοράς

Τα πιο γνωστά δίκτυα αναφοράς που χρησιμοποιούνται για την ανάλυση κοινοτήτων είναι τα εμφωλευμένα l-διαμέρισης (planted l-partition) δίκτυα (Condon and Karp, 2001). Μοντέλα τέτοιων δικτύων χωρίζουν τον ένα γράφο $n = g \cdot l$ κόμβων σε l κοινότητες αποτελούμενες από g σε πλήθος κόμβους. Κάθε κόμβος συνδέεται με κάποιον κόμβος εντός της ίδιας κοινότητας με πιθανότητα p_{in} ενώ με κόμβους διαφορετικής κοινότητας με πιθανότητα p_{ex}. Εάν θέσουμε το $p_{in} > p_{ex}$ τότε αναμένουμε ότι το δίκτυο θα εμπεριέχει κοινοτική δομή, εάν αν θέσουμε $p_{in} < p_{ex}$ τότε το δίκτυο δεν θα αναμένουμε να εμφανίζει κοινοτική δομή και μπορεί να θεωρηθεί ως τυχαίος γράφος, όπως θα παραγόταν από γράφους Erdös-Rényi (Erdos and Rényi, 1960). Τα πιο γνωστά δίκτυα αναφοράς τύπου l-διαμέρισης είναι αυτά των Girvan and Newman (2002). Ένα GN δίκτυο αποτελείται από 128 κόμβους και χωρίζεται σε τέσσερις κοινότητες, οι οποίες αποτελούνται από 32 κόμβους. Ο μέσος βαθμός deg του κόμβου είναι προκαθορισμένος να είναι 16 (λόγω του ότι ο βαθμός παράγεται από πιθανότητες δεν είναι πάντα ακριβώς ίσος με 16). Στο Γράφημα 6.1 μπορούμε να δούμε πως είναι τα δίκτυα, βάσει των διαφορών εσωτερικών βαθμών των κόμβων. Το κατώτερο όριο επιτυγχάνεται όταν ο εσωτερικός βαθμός είναι ίσος με τον εξωτερικό ($deg_{in} =$

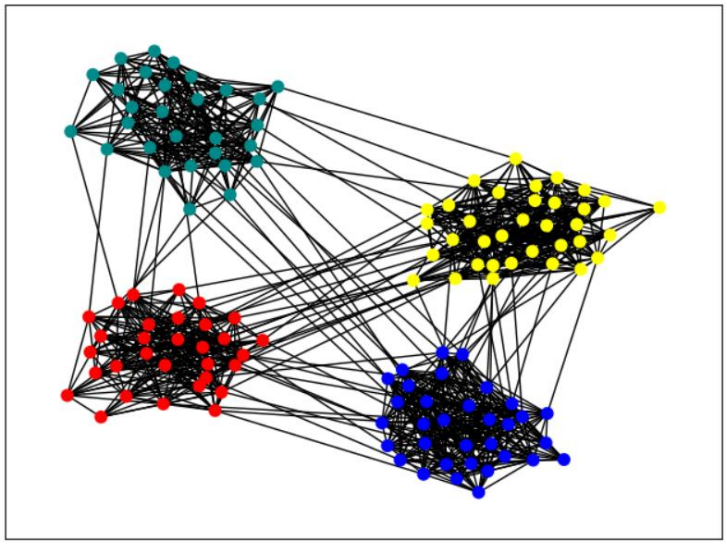
deg_{ex}), όπου και πάλι παρατηρούμε να μην υπάρχει κοινοτική δομή. Βέβαια τα δίκτυα αυτού του τύπου πέρα από το γεγονός ότι είναι μικρά σε μέγεθος έχουν δύο βασικά μειονεκτήματα. Αρχικά, όλοι οι κόμβοι του δικτύου έχουν περίπου τον ίδιο βαθμό το οποίο δεν είναι σύνηθες να συμβεί σε πραγματικά σύνολα δεδομένων. Ενώ το δεύτερο πρόβλημα που παρατηρούμε είναι ότι οι κοινότητες αποτελούνται από το ίδιο ακριβώς πλήθος κόμβων. Αυτά τα δύο χαρακτηριστικά δεν συνηθίζεται να παρατηρούνται σε πραγματικά σύνολα δεδομένων. Οι κατανομές των βαθμών των κόμβων είναι συνήθως ασύμμετρες, με κορυφές χαμηλού αριθμού γειτόνων να συνυπάρχουν με κορυφές μεγάλου αριθμού γειτόνων. Παρόμοια ανομοιομορφία παρατηρείται και στη κατανομή των μεγεθών των κοινοτήτων. Έτσι τα σύνολα αυτά, δεν είναι πάντοτε αρκετά όμοια με τα σύνολα που θέλουμε να ελέγξουμε και δεν μπορούμε να τα χρησιμοποιήσουμε ως βάση για την ανάλυση καταλληλόλητας των αλγορίθμων που θέλουμε να επιλέξουμε. Όπως μπορούμε να παρατηρήσουμε στο Σχήμα 6.1 όταν πλέον ο εσωτερικός βαθμός των κόμβων είναι κοντά στο 8, δεν είναι εύκολο να διακρίνουμε τις κοινότητες μεταξύ τους αλλά ούτε να μπορούμε να αξιολογήσουμε εάν το δίκτυο έχει όντως κοινοτική δομή. Γενικότερα παρατηρείται ότι όσο πιο μικρός είναι ο εσωτερικός βαθμός των κόμβων (μικρή πιθανότητα ένωσης κόμβων εντός της ίδιας κοινότητας) τότε το δίκτυο είναι πλέον παρόμοιο με ένα τυχαίο δίκτυο και δεν έχει κοινοτική δομή. Σε αυτή τη περίπτωση, θα μπορούσε το δίκτυο να χαρακτηριστεί ως τυχαίος γράφος και όχι ως ένας γράφος στον οποίο υπάρχει μια ξεκάθαρη εικόνα για τη δομή του. Αντίστοιχα αποτελέσματα παρατηρούνται εάν μειώσουμε ακόμη περισσότερο τη πιθανότητα ένωσης κόμβων εντός της ίδιας κοινότητας όπως φαίνεται στη τελευταία εικόνα του σχήματος όπου ο εσωτερικός βαθμός των κόμβων είναι ίσος με 2. Πρακτικά τα σχήματα όπου ο βαθμός είναι ίσος με 8 και 2 αντίστοιχα, φαίνονται να είναι αρκετά όμοια. Αντίθετα τα πρώτα δύο γραφήματα όπου έχουμε μεγαλύτερους εσωτερικούς βαθμούς κόμβων (15 και 11), άρα και υψηλότερη πιθανότητα ένωσης, παρατηρείται μια ξεκάθαρη κοινοτική δομή στο δίκτυο.



Σχήμα 6.1. Παράδειγμα δικτύων για διάφορους εσωτερικούς βαθμούς κόμβων.

Δεδομένων αυτών των χαρακτηριστικών, έχουν προταθεί διαφορά μοντέλα παραγωγής συνθετικών τα οποία δημιουργούν δίκτυα με χαρακτηριστικά, αρκετά πιο όμοια των πραγματικών συνόλων. Μια τροποποιημένη μορφή των GN δικτύων είναι τα δίκτυα αναφοράς Gaussian τυχαίων διαμερίσεων, που έχουν προταθεί από τους Brandes et. al. (2003). Στο μοντέλο αυτό, το πλήθος των κόμβων της κάθε κοινότητας ακολουθεί την κανονική κατανομή $N(\mu, \sigma^2)$, με αποτέλεσμα να μην έχουν το ίδιο μέγεθος, αν και δεν διαφέρουν πολύ μεταξύ τους. Η ανομοιογένεια του βαθμού των κοινοτήτων δημιουργεί μια ανομοιογένεια στο βαθμό των κόμβων, διότι ο αναμενόμενος

βαθμός μιας κορυφής εξαρτάται από το πλήθος των κορυφών της συστάδας. Ωστόσο, η μεταβλητότητα του βαθμού των κόμβων και του μεγέθους των κοινοτήτων δεν είναι και τόσο αισθητή. Εκτός αυτού, οι κορυφές εντός της ίδιας κοινότητας έχουν περίπου ίσο βαθμό μεταξύ τους.



Σχήμα 6.2. Παράδειγμα ενός Gaussian μοντέλου με τέσσερις συστάδες.

Όπως μπορούμε να παρατηρήσουμε στο Σχήμα 6.2, ο συνθετικός γράφος (αποτελούμενος από 128 κόμβους) που έχει παραχθεί από ένα Gaussian μοντέλο, είναι αρκετά όμοιος με αυτούς των Girvan-Newman. Αν και το πλήθος κόμβων εντός κοινοτήτων είναι διαφορετικό, δεν είναι αρκετά εμφανές όπως θα ήταν σε ένα πραγματικό δίκτυο. Στο Πίνακα 6.1 μπορεί κάποιος να δει τα ακριβή στοιχεία του γράφου που παράγει το μοντέλο.

Κοινότητες	Πλήθος κόμβων	Μέσος βαθμός
1	29	14,66
2	31	15,48
3	33	17,36
4	35	16,97

Πίνακας 6.1. Αποτελέσματα κοινοτήτων ενός Gaussian μοντέλου.

β. LFR Δίκτυα αναφοράς

Οι αλγόριθμοι που έχουν πλέον αναπτυχθεί όπως και τα συστήματα που χρησιμοποιούνται, μπορούν να αναλύσουν γράφους σε τεράστια σύνολα δεδομένων, σε ικανοποιητικό χρόνο και δεν είναι απαραίτητο να εξετάζονται μόνο σε μικρά σύνολα δεδομένων. Γενικότερα, οι αλγόριθμοι πρέπει να εξετάζονται σε σύνολα όμοια με αυτά στα οποία θέλουμε να τους αξιοποιήσουμε, έτσι ώστε να μπορούμε να αξιολογήσουμε ορθά τα αποτελέσματα που αναμένετε να έχουμε. Οι Palla et al. (2005) παρατήρησαν ότι σε πραγματικά σύνολα δεδομένων το πλήθος των κόμβων αλλά και το μέγεθος κοινοτήτων περιγράφεται καλύτερα από τη κατανομή power law (Clauset, Shalizi and Newman, 2009). Η κατανομή power law περιγράφει ικανοποιητικά σύνολα δεδομένων τα οποία είναι δεξιά ασύμμετρα, δηλαδή το μεγαλύτερο ποσοστό παρατηρήσεων έχουν πολύ μικρές τιμές. Έτσι, θα λέμε ότι μια ποσότητα x ακολουθεί την κατανομή power law όταν προέρχεται από κατανομή πυκνότητας

$$p(x) \propto x^{-\alpha} \tag{6.1}$$

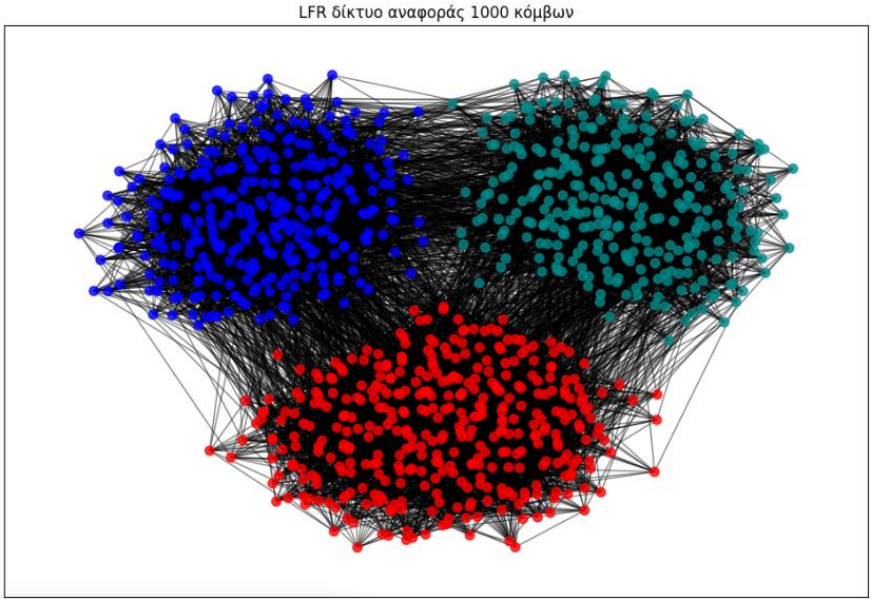
όπου το a είναι μια σταθερή παράμετρος, η οποία ονομάζεται εκθετική παράμετρος ή παράμετρος κλίμακας. Μια συνεχής power law κατανομή περιγράφεται από τη συνάρτηση πυκνότητας πιθανότητας τέτοια ώστε

$$p(x)dx = Pr(x \leq X < x + dx) = Cx^{-\alpha}dx \tag{6.2}$$

όπου το X είναι η παρατηρούμενη τιμή και το C μια σταθερά.

Ένα δίκτυο αναφοράς, όπως προαναφέραμε, πρέπει να έχει μια ανομοιομορφία στην κατανομή των βαθμών των κόμβων αλλά και στο πλήθος των κόμβων ανά κοινότητα, ώστε να είναι όμοιο σε πραγματικά σύνολα δεδομένων για να μπορούμε να έχουμε συγκρίσιμα αποτελέσματα. Δεδομένου ότι τα δίκτυα Girvan-Newman και τα Gaussian δεν μας προσφέρουν ρεαλιστικά σύνολα, όπου η δομή τους θα μπορούσε να συναντηθεί σε πραγματικά σύνολα δεδομένων, οι Lancichinetti, Fortuno and Radicchi (2008) δημιούργησαν ένα καινούργιο μοντέλο για τη δημιουργία δικτύων αναφοράς, τα LFR δίκτυα. Για τη δημιουργία των LFR δικτύων χρησιμοποιήθηκε η βιβλιοθήκη της NetworkX (Hagberg, Swart and Schult, 2008). Τα LFR δίκτυα που δημιουργούνται από τη βιβλιοθήκη είναι μη κατευθυνόμενα (undirected) και μη σταθμισμένα (unweighted). Αν θέλει κάποιος να προσθέσει βάρη στα δίκτυα, θα πρέπει να παράξει τα δεδομένα των βαρών σε μεταγενέστερο στάδιο (Lancichinetti and Fortunato 2009). Τα χαρακτηριστικά που πρέπει να συμπληρωθούν για να παραχθεί το δίκτυο είναι

1) N: συνολικό πλήθος κόμβων
2) t_1: παράμετρος της κατανομής power law
3) t_2: παράμετρος της κατανομής power law
4) m_u (mixing parameter): το ποσοστό ακμών που συνδέουν έναν κόμβο με κόμβους άλλων κοινοτήτων.
5) $\bar{k}$: μέσος βαθμός κόμβων
6) k_{min}: ελάχιστος βαθμός που μπορεί να έχει ένας κόμβος
7) k_{max}: μέγιστος βαθμός που μπορεί να έχει ένας κόμβος
8) C_{min}: ελάχιστο μέγεθος κοινότητας
9) C_{max}: μέγιστο μέγεθος κοινότητας
10) ε: Ανοχή κατά τη σύγκριση της μέσης τιμής
11) max_iters: Μέγιστος αριθμός επαναλήψεων για την προσπάθεια δημιουργίας των κατάλληλων δομικών προδιαγραφών που επιλέγει ο χρήστης (π.χ. μέγεθος κοινοτήτων).
12) seed: δείκτης τυχαίου αριθμού για τη παραγωγή συνεχώς του ίδιου δικτύου. Ο δείκτης αυτός είναι αρκετά χρήσιμος αφού αν επιλέξουν δύο διαφορετικοί χρήστες να παραχθεί ένα δίκτυο με κάποια συγκεκριμένα χαρακτηριστικά και θέσουν τον ίδιο αριθμό του δείκτη seed τότε θα παράγουν ακριβώς τον ίδιο γράφο.



Σχήμα 6.3. Παράδειγμα ενός LFR δικτύου αναφοράς.

Αρχικά, κάθε κόμβος παίρνει έναν βαθμό, ο οποίος παράγεται βάσει της κατανομής power law $k_i \backsim k^{t_1}$ (με i συμβολίζεται ο δείκτης του κόμβου). Τα άκρα της κατανομής k_{min} και k_{max} επιλέγονται έτσι ώστε ο μέσος βαθμός να είναι ίσος με $\bar{k}$, τα οποία μπορούν να προκαθοριστούν από τον χρήστη. Έπειτα, ο αλγόριθμος υπολογίζει το πραγματικό μέσο που έχει παράξει το δίκτυο $\overline{k_t}$ και αν $\overline{k_t} - \bar{k} > \varepsilon$ τότε ο αλγόριθμος επαναυπολογίζει τους βαθμούς έως ότου $\overline{k_t} - \bar{k} < \varepsilon$. Βέβαια, πρέπει να επισημανθεί ότι, αν ο επαναυπολογισμός των βαθμών γίνει περισσότερες από max_iters φορές, τότε ο αλγόριθμος αποτυγχάνει. Κάθε κόμβος μοιράζεται $1 - m_u$ του ποσοστού των συνδέσεων του με κόμβους της ίδια κοινότητας, ενώ το υπόλοιπο ποσοστό m_u, με κόμβους άλλων κοινοτήτων. Βέβαια επειδή ο αλγόριθμος επιλέγει τυχαία τους κόμβους που θα ενωθούν εντός και εκτός συστάδας, είναι συχνό να υπάρχουν κόμβοι που έχουν ακμή προς τον εαυτό τους. Σε αυτές της περιπτώσεις είναι στο χέρι του ερευνητή να αφαιρέσει τέτοιου είδους ακμές. Δεδομένου του γεγονότος ότι σε κοινωνικά δίκτυα όπως αυτά που θα μελετήσουμε δεν θα υπάρχουν κόμβοι οι οποίοι θα ενώνονται με τον εαυτό τους, μετά την παραγωγή συνθετικών δικτύων θα αφαιρέσουμε οποιαδήποτε τέτοιου είδους ακμή προκύψει. Εφόσον έχουν πλέον υπολογιστεί οι βαθμοί όλων των κόμβων εντός του δικτύου, ο αλγόριθμος υπολογίζει το πλήθος των κόμβων εντός κάθε κοινότητας βάσει της κατανομής power law, $n_c \backsim n^{t_2}$ (όπου με n_c συμβολίζεται το μέγεθος της εκάστοτε κοινότητας. Όπως και στο βαθμό των κόμβων, ο αλγόριθμος μπορεί να κάνει έως και max_iters επαναλήψεις για να υπολογίσει το μέγεθος των κοινοτήτων ώστε να ταιριάζουν στις απαιτήσεις που έχει θέσει ο χρήστης (μέγιστο/ελάχιστο πλήθος για κάθε κοινότητα). Έτσι, είμαστε σίγουροι για το δομικό σχέδιο που έχουμε προκαθορίσει. Οι κόμβοι που δημιουργεί ο αλγόριθμος είναι αρχικά εκτός των κοινοτήτων και κάθε κόμβος επαναληπτικά ανατίθεται τυχαία σε κάποια κοινότητα ώστε να πληροί τα προκαθορισμένα κριτήρια.

6.3 Αποτελέσματα της μελέτης συνθετικών δικτύων

Σε αυτή την ενότητα, θα παρουσιάσουμε διάφορα συνθετικά δίκτυα τα οποία παράχθηκαν για να συγκρίνουμε τα αποτελέσματα των αλγορίθμων που παρουσιάσαμε στα προηγούμενα κεφάλαια. Αρχικά, θα παρουσιάσουμε τον δείκτη NMI (normalized mutual information) τον οποίο

αξιοποιήσαμε για την αξιολόγηση εύρεσης κοινοτήτων σε LFR δίκτυα αναφοράς. Έπειτα, θα συγκρίνουμε τους αλγόριθμους εύρεσης τριγώνων για τους οποίους παράχθηκαν δίκτυα τύπου LFR όπως και τυχαία δίκτυα *Erdös-Rényi.*

6.3.1 Αξιολόγηση των ομαδοποιήσεων

Για να συγκρίνουμε τα αποτελέσματα κάθε αλγορίθμου, χρειαζόμαστε ένα ποσοτικό κριτήριο για να υπολογίσουμε πόσο κοντά είναι οι κοινότητες που εξάγουν τα δίκτυα με τις πραγματικές κοινότητες των δικτύων αναφοράς. Όπως είδαμε και στο τρίτο κεφάλαιο υπάρχουν διάφοροι δείκτες που μπορούν να συμβάλουν στην αξιολόγηση των ομαδοποιήσεων, με πιο γνωστό και πολυχρησιμοποιημένο δείκτη τον Normalized Mutual Information (NMI). Ο δείκτης NMI βασίζεται στην ομοιότητα mutual information (MI) όπου και την οποία μετατρέπει σε κλίμακα $[0,1]$. Όσο μεγαλύτερη είναι η τιμή του δείκτη τόσο πιο κοντά θεωρούμε πως βρίσκονται τα αποτελέσματα του αλγόριθμου στις 'πραγματικές' κοινότητες οι οποίες έχουν οριστεί. Ο δείκτης αυτός διαχειρίζεται τις διαμερίσεις ως ονομαστικές (nominal) τιμές. Ο δείκτης NMI δίνεται από τον τύπο

$$NMI(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij} log\left(\frac{N_{ij}N}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{C_A} N_{i.}\, log\left(\frac{N_{i.}}{N}\right) + \sum_{j=1}^{C_B} N_{j.}\, log\left(\frac{N_{.j}}{N}\right)} \tag{6.3}$$

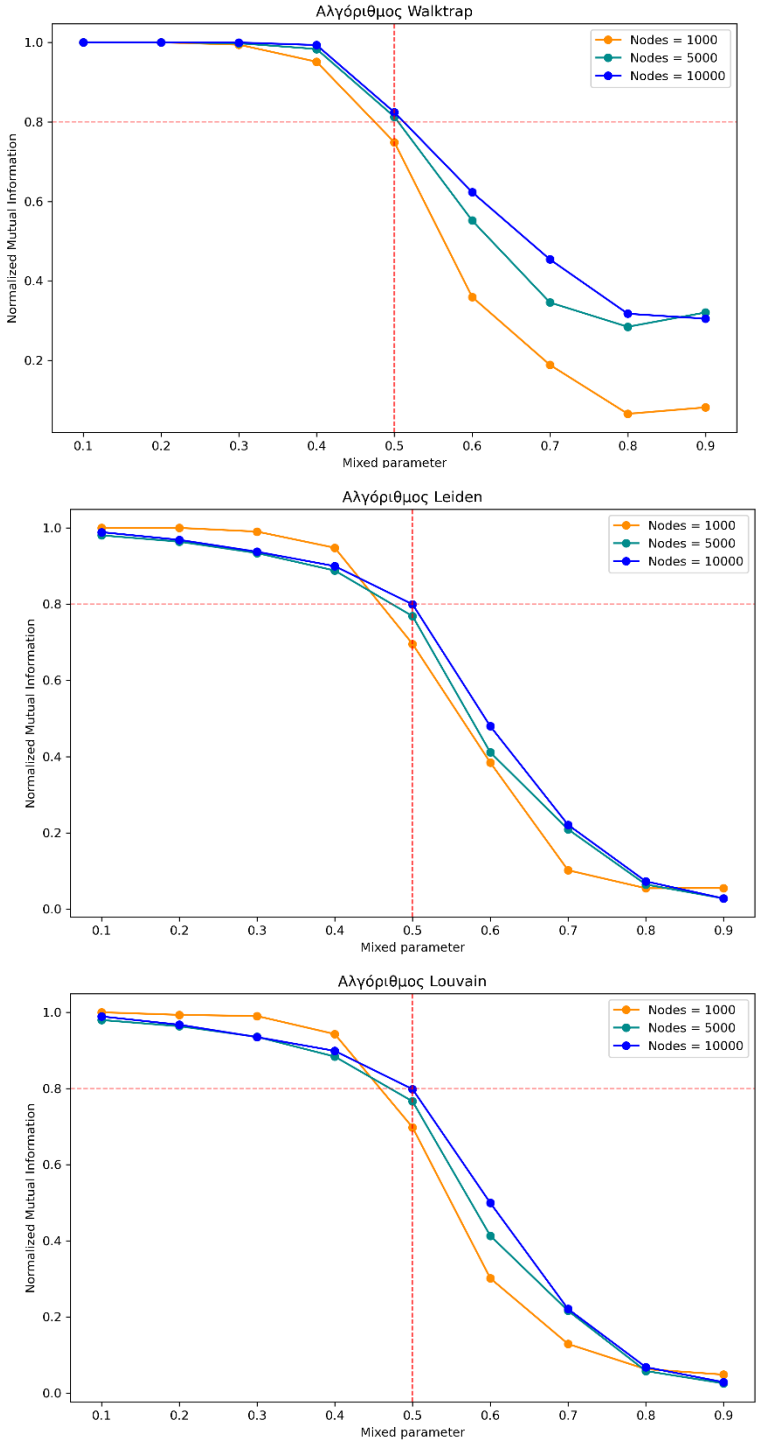
όπου το C_A συμβολίζει το πλήθος κοινοτήτων όπως παράγονται από το δίκτυο αναφοράς (ground truth), ενώ το C_B συμβολίζει το πλήθος κοινοτήτων όπως παράγονται από τον αλγόριθμο εύρεσης κοινοτήτων που αναλύουμε. Το N συμβολίζει το συνολικό πλήθος κόμβων του δικτύου, με N_{ij} συμβολίζεται το πλήθος των κόμβων που ανήκει στις κοινότητες i και j των κοινωνιών A και B αντίστοιχα, ενώ με $N_{i.}$ συμβολίζουμε το σύνολο των ατόμων της κοινότητας i της κοινωνίας A. Βέβαια όπως έχει αποδειχθεί, ο δείκτης NMI μπορεί να μεροληπτεί, ειδικότερα σε περιπτώσεις όπου ο αλγόριθμος παράγει περισσότερες κοινότητες από όσες υπάρχουν στη πραγματικότητα (Mahmoudi and Jemielniak, 2024).

6.3.2 Αποτελέσματα εύρεσης κοινοτήτων σε συνθετικά δίκτυα

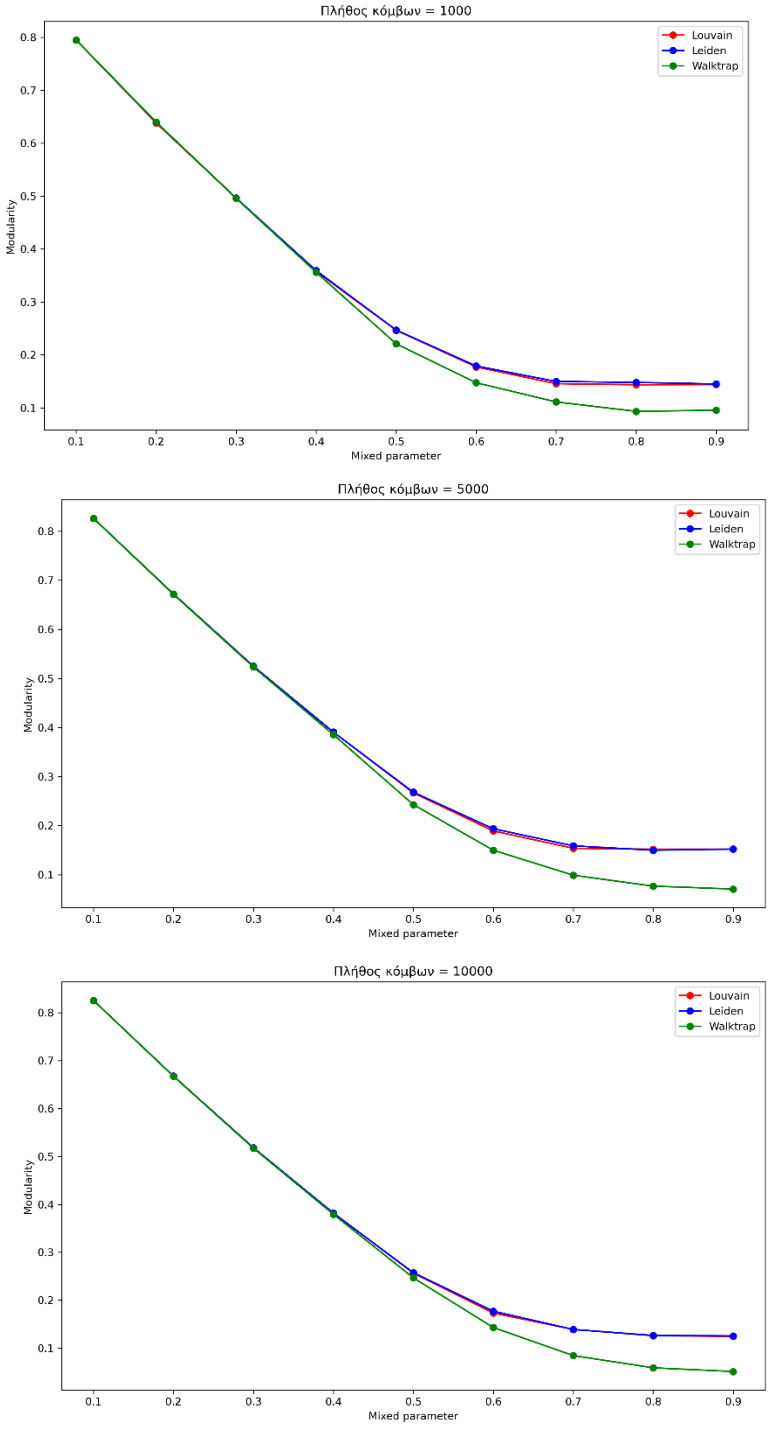
Για την αξιολόγηση των συνθετικών δικτύων δημιουργήσαμε σύνολα δεδομένων με 1000, 5000 και 10000 κόμβους. Τα δίκτυα αυτά παράχθηκαν για τιμές παραμέτρου μίξης 0.1-0.9 με βήμα 0.1. Στον Πίνακα 6.2 δίνονται οι παράμετροι που χρησιμοποιήθηκαν για τη δημιουργία αυτών των δικτύων

Πλήθος κόμβων	1000	5000	10000
t_1	3	3	3
t_2	1.5	1.5	1.5
k_{min}	20	20	30
k_{max}	50	50	60

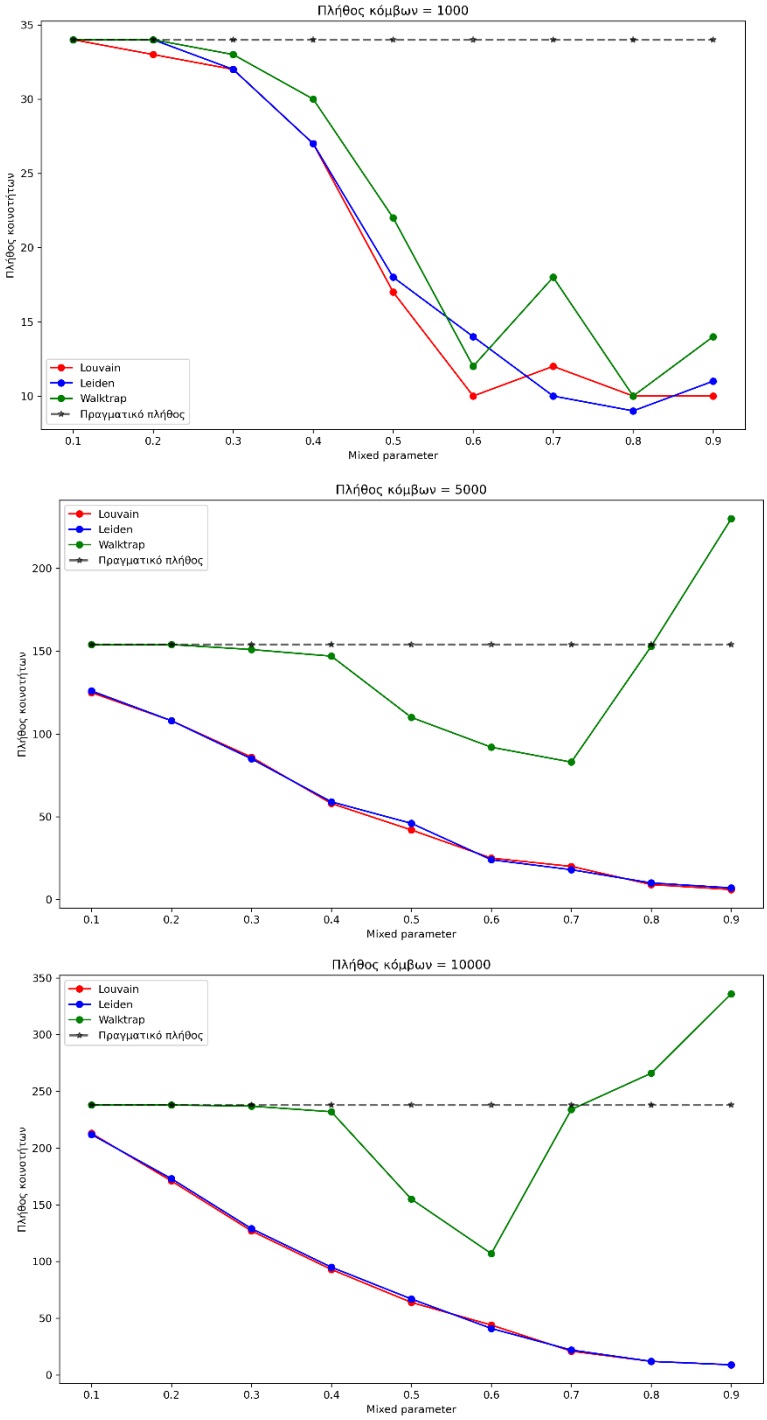
Πίνακας 6.2. Παράμετροι για τη δημιουργία συνθετικών δικτύων για τις διάφορες τιμές παραμέτρου μίξης.



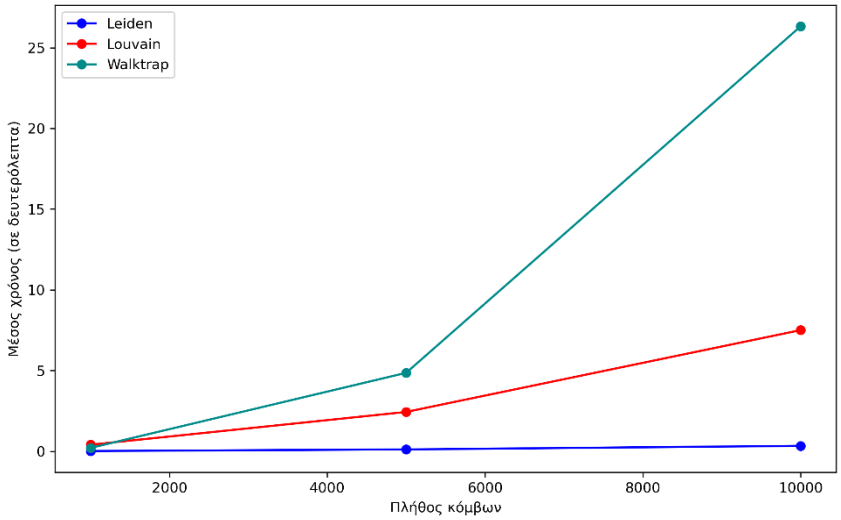
Σχήμα 6.4. Τιμές NMI των αλγορίθμων για διάφορες τιμές παραμέτρου μίξης.



Σχήμα 6.5. Αποτελέσματα modularity για τον κάθε αλγόριθμο.



Σχήμα 6.6. Σύγκριση πλήθους κοινοτήτων των αλγορίθμων.



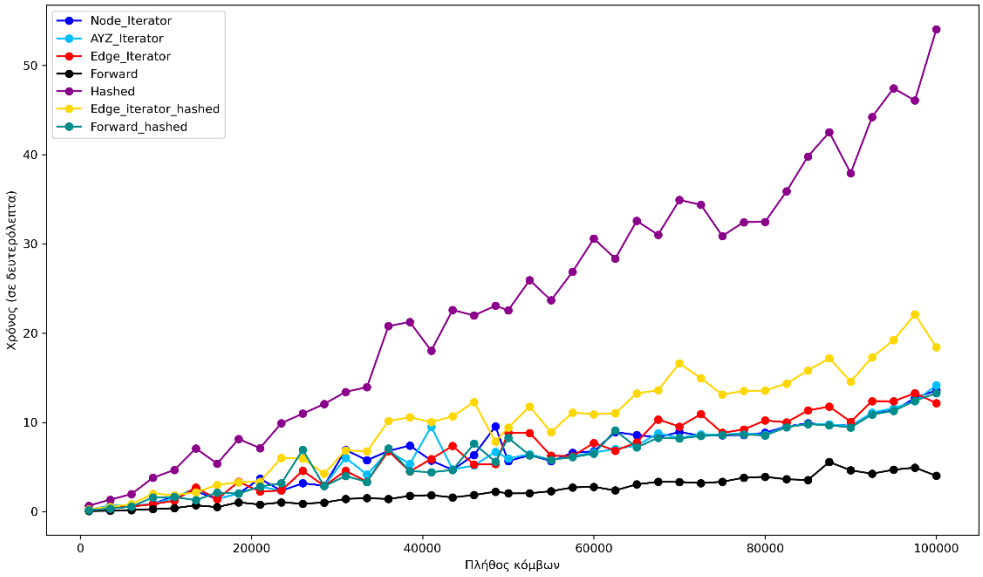
Σχήμα 6.7. Χρόνοι για την εύρεση κοινοτήτων που χρειάστηκε κάθε αλγόριθμος.

Στο Σχήμα 6.4 μπορούμε να παρατηρήσουμε ότι οι υψηλότερες τιμές NMI δίνονται από τον αλγόριθμο τυχαίου περιπάτου Walktrap, για τιμές παραμέτρου μίξης μεγαλύτερες του 0.2. Όπως προαναφέραμε βέβαια ο δείκτης NMI μπορεί να μεροληπτεί σε περιπτώσεις όπου ο αλγόριθμος βρίσκει πλήθος ομάδων μεγαλύτερο από αυτό που έχει παραχθεί από το συνθετικό δίκτυο. Επίσης, για μεγάλες τιμές τις παραμέτρου μίξης (Σχήμα 6.5), ο αλγόριθμος Walktrap δίνει αρκετά χαμηλότερες τιμές modularity σε σύγκριση με τους Louvain και Leiden. Αυτό μας υποδεικνύει ότι, αν σε ένα δίκτυο οι κοινότητες δεν είναι αρκετά δεμένες μεταξύ τους, δηλαδή οι κόμβοι των κοινοτήτων έχουν αντιστοίχως πολλούς γείτονες εκτός της κοινότητάς τους, ο αλγόριθμος Walktrap δεν δίνει τόσο ικανοποιητικά αποτελέσματα. Βέβαια, όπως μπορούμε να παρατηρήσουμε στο Σχήμα 6.6, οι αλγόριθμοι Louvain και Leiden δεν βρίσκουν πλήθος ομάδων αρκετά όμοιο με αυτόν που παρήγαγαν τα LFR δίκτυα όμως το modularity που έχουν αυτές οι ομαδοποιήσεις είναι υψηλότερο, με αποτέλεσμα να μπορούμε να θεωρήσουμε τις διαμερίσεις λίγο καλύτερες από αυτές του Walktrap. Ένα μεγάλο πλεονέκτημα που έχουν οι αλγόριθμοι Louvain και Leiden είναι ο χρόνος που χρειάζονται για την εύρεση των ομάδων αφού φαίνεται να είναι ταχύτεροι από τον Walktrap, ειδικά όσο μεγαλώνει το πλήθος των κόμβων.

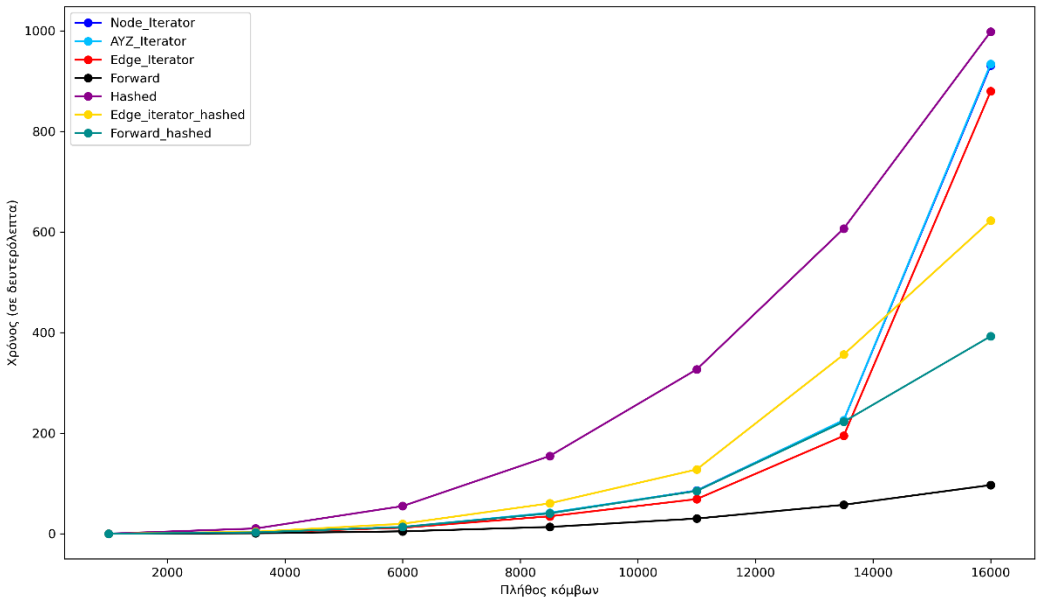
6.3.3 Αποτελέσματα σχετικά με την απαρίθμηση τριγώνων σε συνθετικούς γράφους

Για την αξιολόγηση των αλγορίθμων εύρεσης τριγώνων δημιουργήσαμε μη κατευθυνόμενα και χωρίς βάρη LFR δίκτυα αναφοράς, με μέσο βαθμό $\bar{k} = 35$ (μέσο πλήθος γειτόνων). Αντίστοιχα, δημιουργήσαμε τυχαία δίκτυα Erdös-Rényi $G_{n,p}$ για $p = 0.05$, έτσι ώστε να μπορέσουμε να αξιολογήσουμε τη συμπεριφορά και την αποτελεσματικότητα των αλγορίθμων σε δίκτυα τα οποία δεν έχουν κοινοτική δομή. Ο αλγόριθμος Forward φαίνεται να είναι ο πιο αποτελεσματικός αλγόριθμος για την εύρεση τριγώνων (βλέπε Σχήμα 6.8 και 6.9), αφού χρειάζεται το λιγότερο χρόνο για να υπολογίσει τον αριθμό τριγώνων αλλά έχει και την πιο σταθερή αύξηση χρόνου όσο αυξάνεται το πλήθος των κόμβων.

Τα τυχαία δίκτυα τείνουν να μην έχουν κόμβους με υψηλό αριθμό γειτόνων, αλλά όλοι οι κόμβοι έχουν περίπου τον ίδιο αριθμό γειτόνων. Αντίθετα, τα LFR δίκτυα τα οποία δημιουργούνται με τρόπο ώστε να έχουν κοινοτική δομή, εμπεριέχουν κόμβους με μεγάλο αριθμό γειτόνων αλλά και κόμβους με μικρό αριθμό γειτόνων. Ο αλγόριθμος edge-iterator παρατηρούμε ότι επηρεάζεται περισσότερο από το μέσο πλήθος γειτόνων του δικτύου, αφού για τους τυχαίους γράφους φαίνεται να μην έχει τόσο σταθερά αποτελέσματα, ιδίως όταν αυξάνεται το πλήθος των κόμβων και επακολούθως το μέσο πλήθος κόμβων.



Σχήμα 6.8. Χρόνοι για την εύρεση τριγώνων για δίκτυα αναφοράς LFR.



Σχήμα 6.9. Χρόνοι για την εύρεση τριγώνων για δίκτυα Erdös-Rényi.

6.4 Αποτελέσματα πραγματικών δικτύων

Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα των αλγορίθμων σε δίκτυα πραγματικών συνόλων δεδομένων. Όλα τα δίκτυα που χρησιμοποιήθηκαν προήλθαν από τη βάση δεδομένων του SNAP (Jure, 2014). Τα σύνολα δεδομένων χωρίζονται σε σύνολα δύο κατηγορίες. Η πρώτη κατηγορία αφορά σύνολα για τα οποία έχουμε τους κόμβους χωρισμένους σε πραγματικές κοινότητες, ενώ τα υπόλοιπα σύνολα δεν έχουν αυτή τη πληροφορία. Παρακάτω παρουσιάζονται τα δίκτυα που χρησιμοποιήθηκαν όπως και ο πίνακας με τα χαρακτηριστικά του κάθε δικτύου (Πίνακας 6.3).

- **email-EU-core:** Σο συγκεκριμένο δίκτυο δημιουργήθηκε από δεδομένα ηλεκτρονικού ταχυδρομείου ενός ευρωπαϊκού ινστιτούτου. Οι κόμβοι του δικτύου είναι αποκλειστικά άτομα του ινστιτούτου, αφού έχει αφαιρεθεί οποιοδήποτε εισερχόμενο ή εξερχόμενο μήνυμα, από ή προς άτομα που δεν ανήκουν στο ινστιτούτο. Κάθε άτομο του ινστιτούτου ανήκει σε ένα από τα 42 τμήματα του ερευνητικού κέντρου τα οποία έχουν οριστεί ως οι πραγματικές κοινότητες (ground truth).

- **DBLP:** Το σύνολο δεδομένων της DBLP περιέχει ένα πλήρη κατάλογο ερευνητικών άρθρων στην επιστήμη τον υπολογιστών. Το σύνολο δημιουργήθηκε από ερευνητές οι οποίοι είναι συνδεδεμένοι μεταξύ τους στη περίπτωση που έχουν δημοσιεύσει μαζί κάποιο άρθρο. Ο τόπος δημοσίευσης (π.χ. ένα συνέδριο) αποτελεί μια από τις πραγματικές κοινότητες (ground truth) και ένας ερευνητής ανήκει σε μια από τις κοινότητες, στη περίπτωση όπου έχει δημοσιεύσει έστω και μια φορά σε αυτή.

- **GitHub:** Το GitHub είναι ένα ηλεκτρονικό αποθετήριο, το οποίο επιτρέπει σε προγραμματιστές να αποθηκεύουν και να επιβλέπουν τους κώδικες τους. Το σύνολο δεδομένων συλλέχθηκε από το δημόσιο API του GitHub το 2019. Οι κόμβοι είναι προγραμματιστές οι οποίοι έχουν δώσει ένα αστέρι σε τουλάχιστον 10 αποθετήρια κώδικα και οι ακμές είναι οι αμοιβαίες σχέσεις (ακολουθεί ένας προγραμματιστής έναν άλλον) μεταξύ τους. Τα χαρακτηριστικά των κόμβων εξάγονται βάσει της τοποθεσίας, των αποθετηρίων που έχουν ξεκινήσει, τον εργοδότη και τη διεύθυνση ηλεκτρονικού ταχυδρομείου. Στο συγκεκριμένο σύνολο δεν έχουμε εκ των προτέρων κάποια ομαδοποίηση μεταξύ των κόμβων.

- **ego-Facebook:** Το σύνολο αυτό αποτελείται από φίλους του Facebook. Τα δεδομένα συλλέχθηκαν από τους συμμετέχοντας έρευνας η οποία διεξάχθηκε με τη χρήση της εφαρμογής του Facebook. Τα ego-networks αποτελούνται από έναν κεντρικό κόμβο ("ego") και τους κόμβους στους οποίους συνδέεται "alters", όπως και τους δεσμούς μεταξύ των alters. Τα δεδομένα του Facebook έχουν γίνει ανώνυμα αντικαθιστώντας τα εσωτερικά ID του Facebook για κάθε χρήστη με μια νέα τιμή. Επίσης, ενώ έχουν παρασχεθεί διανύσματα χαρακτηριστικών από αυτό το σύνολο δεδομένων, η ερμηνεία αυτών των χαρακτηριστικών έχει συγκαλυφθεί. Για παράδειγμα, όπου το αρχικό σύνολο δεδομένων μπορεί να περιείχε ένα χαρακτηριστικό (πχ. πολιτική = Δημοκρατικό Κόμμα), τα νέα δεδομένα θα το αποκρύπτουν (πχ. πολιτική = ανώνυμο χαρακτηριστικό 1). Έτσι, χρησιμοποιώντας τα ανώνυμα δεδομένα, είναι δυνατόν να προσδιοριστεί αν

δύο χρήστες υποστηρίζουν τα ίδια πολιτικά κόμματα, αλλά όχι αυτό που υποστηρίζει ο καθένας μεμονωμένα.

Σύνολο	Email	DBLP	GitHub	Facebook
Πλήθος κόμβων	1005	317080	37700	4039
Πλήθος ακμών	25571	1049866	289003	88234
Μέσος συντελεστής συσταδοποίησης	0.3994	0.6324	0.1675	0.6055
Πλήθος τριγώνων	105461	2224385	523810	1612010
Διάμετρος	7	21	11	8
ground-truth κοινότητες	Ναι	Ναι	Όχι	Όχι

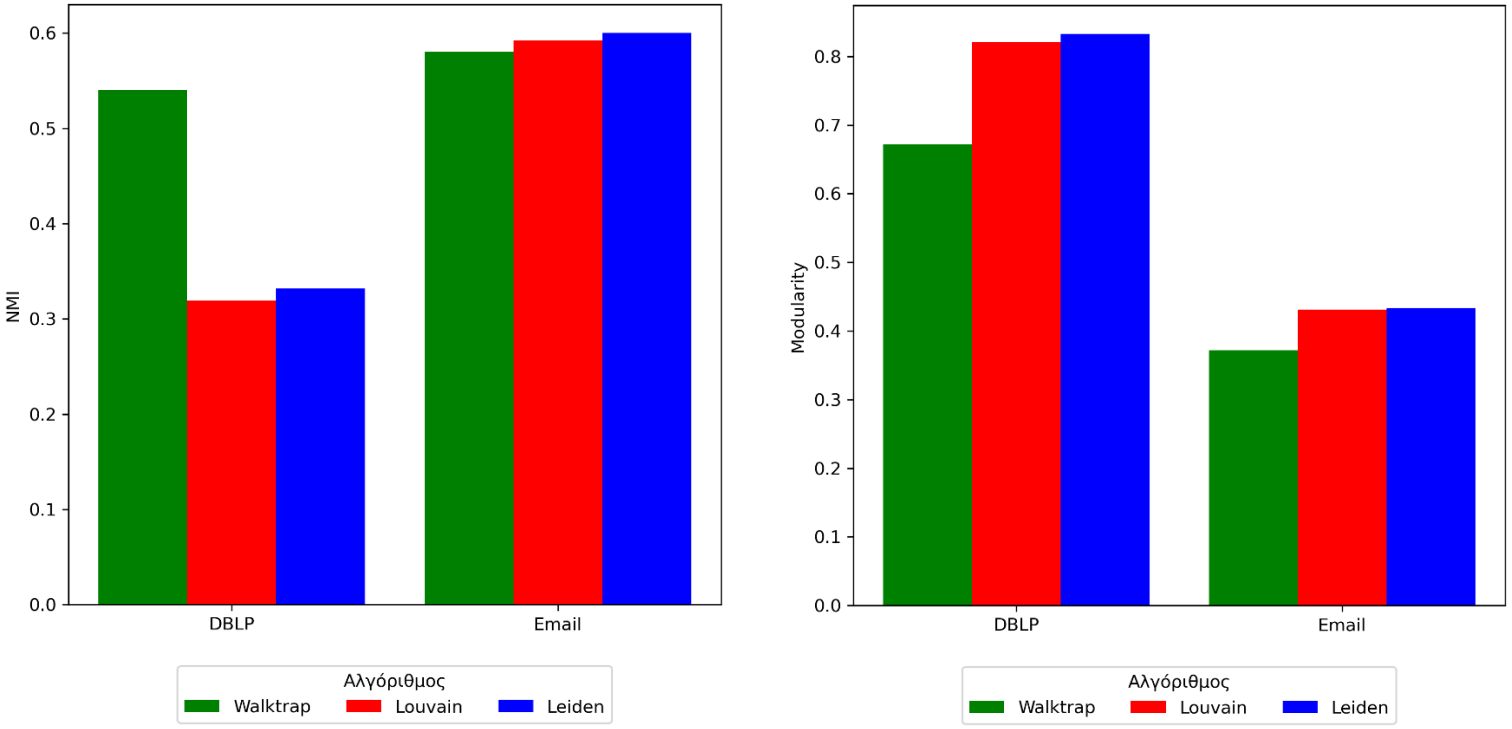
Πίνακας 6.3. Χαρακτηριστικά πραγματικών συνόλων δεδομένων.

6.4.1 Αξιολόγηση ομαδοποιήσεων σε σύνολα πραγματικών δεδομένων

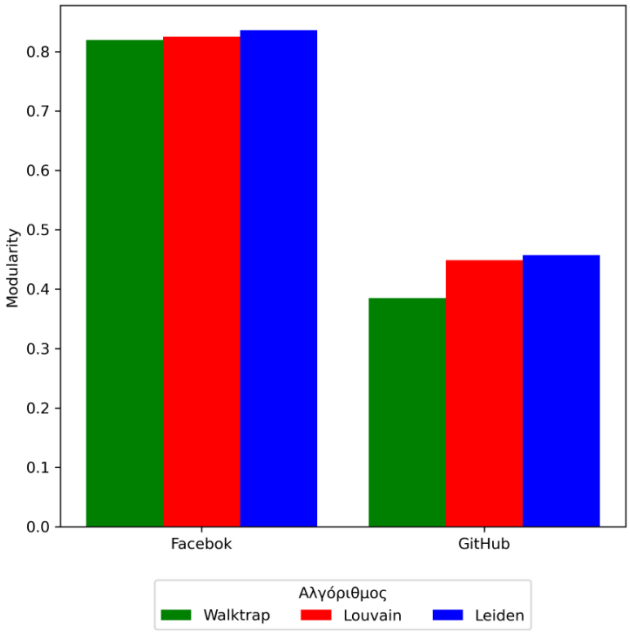
Για την αξιολόγηση των αποτελεσμάτων των συνόλων θα συγκρίνουμε το σύνολο των κοινοτήτων που δημιουργεί ο κάθε αλγόριθμος, τον χρόνο που χρειάστηκε για να υπολογίσει τις κοινότητες καθώς και αν μας δίνει κοινότητες οι οποίες είναι αποσυνδεμένες μεταξύ τους ή κοινότητες αποτελούμενες από μόνο έναν κόμβο. Επίσης για τα σύνολα των οποίων έχουμε τους κόμβους εντός προκαθορισμένων κοινοτήτων (Email και DBLP) θα μελετήσουμε τα αποτελέσματα του δείκτη NMI για να δούμε κατά πόσο οι αλγόριθμοι μας έδωσαν κοινότητες όμοιες με αυτές που έχουν σχηματισθεί κατά τη συλλογή. Από τα σύνολα έχουν αφαιρεθεί τα loops (ακμή που συνδέει έναν κόμβο με τον εαυτό του) όπως και οι κόμβοι οι οποίοι είναι απομονωμένοι από το υπόλοιπο δίκτυο (isolated). Ο καθαρισμός έγινε με τη βοήθεια της βιβλιοθήκης του NetworkX, στα συγκεκριμένα, αρχικά αφαιρέθηκαν όλα τα loops των κόμβων ενώ έπειτα αφαιρέθηκαν όλοι οι αποκομμένοι από το δίκτυο κόμβοι. Η σειρά των βημάτων, είναι καθοριστικής σημασίας καθώς η βιβλιοθήκη δεν θεωρεί ως απομονωμένους τους κόμβους που έχουν ως γείτονα μόνο τον εαυτό τους.

Αρχικά, για το σύνολο δεδομένων DBLP παρατηρούμε ότι ο αλγόριθμος Walktrap δίνει καλύτερα αποτελέσματα ως προς τον δείκτη NMI (βλέπε Σχήμα 6.10). Αυτό βέβαια συμβαίνει διότι οι άλλοι δύο αλγόριθμοι βρίσκουν πολύ λιγότερες κοινότητες από τις πραγματικές, με αποτέλεσμα να μειώνεται αρκετά ο δείκτης NMI, αφού συγκρίνει κοινότητες αρκετά

ανομοιόμορφες μεταξύ τους. Αυτό έχει ως αποτέλεσμα, ο αλγόριθμος Walktrap ο οποίος βρίσκει πλήθος κοινοτήτων πλησιέστερα με αυτά του πραγματικού να παρουσιάζει μεγαλύτερες τιμές του δείκτη. Αντίθετα, στο σύνολο δεδομένων Email παρατηρούμε ότι ο δείκτης NMI (βλέπε Σχήμα 6.10) είναι αρκετά όμοιος για όλους τους αλγορίθμους, παρόλο που ο αλγόριθμος Walktrap δημιουργεί πλήθος κοινοτήτων μεγαλύτερο από αυτά των υπολοίπων αλγορίθμων. Για το σύνολο του DBLP, παρατηρούμε δυσανάλογα αποτελέσματα μεταξύ των δεικτών, αφού οι δύο αλγόριθμοι (Louvain και Leiden) έχουν αρκετά μεγαλύτερες τιμές modularity σε σύγκριση με αυτές του Walktrap. Για το σύνολο δεδομένων των Email παρατηρούμε ότι ο αλγόριθμος Leiden μας δίνει καλύτερα αποτελέσματα με πολύ μικρές διαφορές αφού και οι δύο δείκτες (modularity και NMI) είναι υψηλότερες.



Σχήμα 6.10. Αποτελέσματα NMI και Modularity των συνόλων Email και DBLP.



Σχήμα 6.11. Αποτελέσματα Modularity για τα σύνολα Facebook και GitHub.

Για τα σύνολα δεδομένων των οποίων δεν δίνονται κοινότητες (Facebook και GitHub) παρατηρούμε αρκετά παρόμοια αποτελέσματα, όπου ο αλγόριθμος Leiden μας δίνει ελάχιστα καλύτερες τιμές modularity από τους άλλους αλγορίθμους (βλέπε Σχήμα 6.11). Ο αλγόριθμος Leiden σε όλα τα σύνολα δεδομένων, συμπεριλαμβανομένων των συνθετικών δικτύων παρατηρούμε πως δίνει ελάχιστα καλύτερα αποτελέσματα, ως προς την δομή των παραγόμενων κοινοτήτων, αφού το modularity των κοινοτήτων που βρίσκει είναι υψηλότερο (βλέπε Σχήμα 6.5, 6.10 και 6.11). Γενικότερα ο αλγόριθμος φαίνεται να μη μας παραδίδει πάντα πλήθος κοινοτήτων όπως το σύνολο των προκαθορισμένων (π.χ. Σχήμα 6.6) αλλά παράγει κοινότητες που είναι καλά ορισμένες. Μια βασική διαφορά που έχει από τους άλλους αλγόριθμους είναι πως έχει τη μεγαλύτερη ταχύτητα υπολογισμού (βλέπε Σχήμα 6.7 και 6.12). Παρότι ο Leiden έχει ένα έξτρα βήμα για τον υπολογισμό των κοινοτήτων, σε σύγκρισή με τον Louvain, παρατηρούμε πως είναι αρκετά ταχύτερος και δίνει καλύτερα αποτελέσματα.

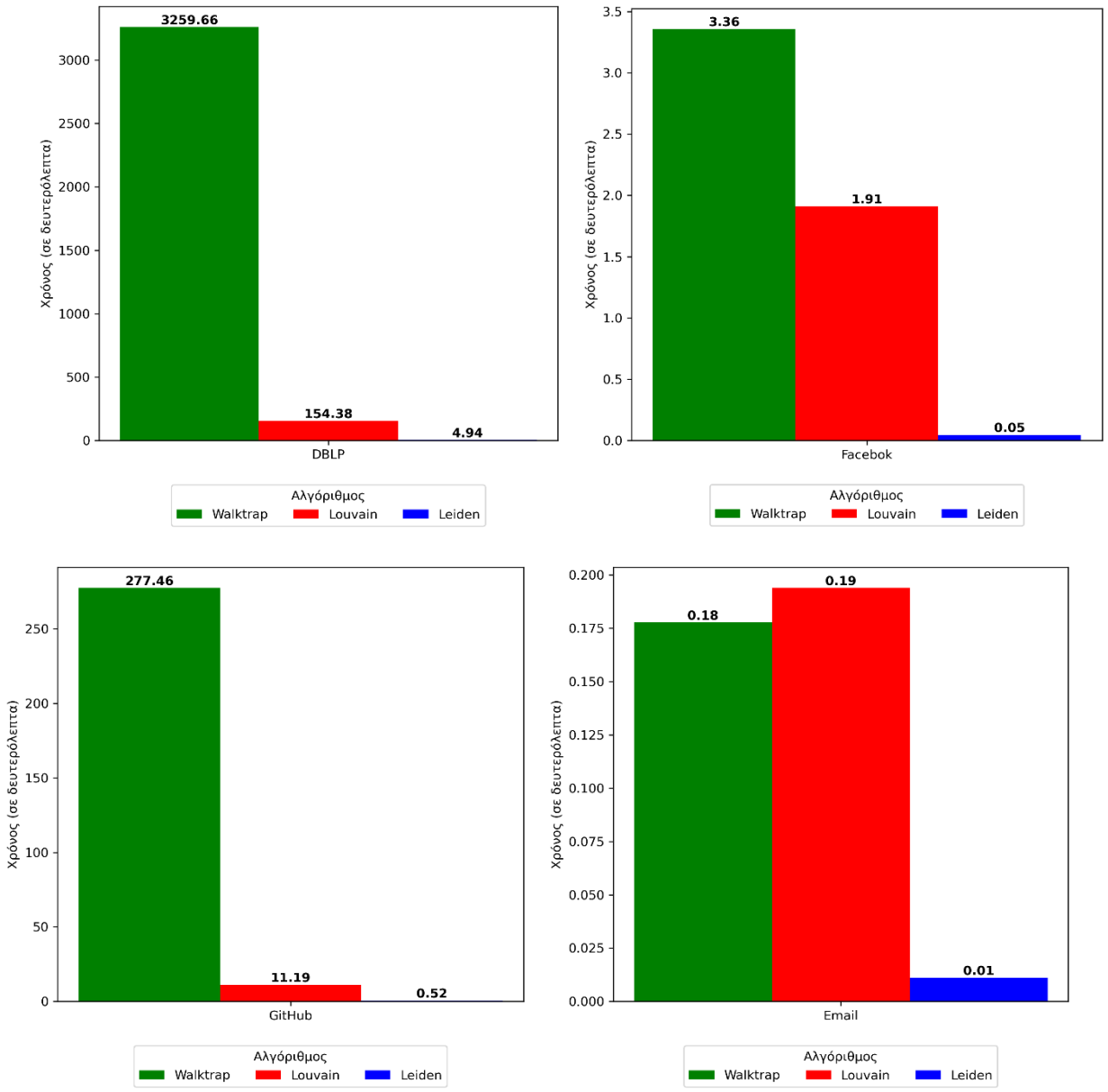
Για να αξιολογήσουμε αν οι αλγόριθμοι δημιουργούν αποσυνδεμένες κοινότητες (κοινότητες που ένας κόμβος δεν έχει μονοπάτι που τον οδηγεί σε κόμβο της ίδιας κοινότητας χωρίς να περάσει από κόμβο κάποιας άλλης κοινότητας) τρέξαμε τους αλγορίθμους 10 φορές έκαστο. Παρατηρούμε ότι ο αλγόριθμος Louvain δημιουργεί αποσυνδεμένες μεταξύ τους κοινότητες (βλέπε Πίνακα 6.4).

Σε μικρά δίκτυα (Email, Facebook) ο αλγόριθμος δεν δημιούργησε αποσυνδεμένες κοινότητες, αλλά όσο αυξάνεται το πλήθος των κόμβων βλέπουμε πως το ποσοστό αυτό αυξάνεται (Ο πίνακας αποτελεσμάτων όλων των επαναλήψεων βρίσκεται στο Παράρτημα 1). Οι αλγόριθμοι Leiden και Walktrap φαίνεται να μην δημιουργούν αποσυνδεμένες κοινότητες, ενώ κατά τον αλγόριθμο Louvain δεν δημιουργήθηκαν κοινότητες, όπου ο αποσυνδεμένος κόμβος ήταν εντελώς απομονωμένος από την υπόλοιπη κοινότητα.

Δίκτυο	Ποσοστό
DBLP	3,4%
Email	0,0%
Facebook	0,0%
GitHub	6,4%

Πίνακας 6.4. Ποσοστά αποσυνδεμένων κοινοτήτων αλγορίθμου Louvain για δέκα επαναλήψεις.

Όλοι οι αλγόριθμοι δοκιμάστηκαν για τα δίκτυα που συλλέχθηκαν από τη βιβλιοθήκη του SNAP, χωρίς να αφαιρεθούν τα loops ή οι αποσυνδεμένοι από το υπόλοιπο δίκτυο κόμβοι. Τα αποτελέσματα που παρουσίασαν είναι αρκετά όμοια με αυτά που παρουσιάζονται στην ενότητα αυτή. Όλοι οι αλγόριθμοι, δημιούργησαν ξεχωριστές ομάδες για τους αποσυνδεμένους κόμβους. Επιπλέον, ο χρόνος που χρειάστηκαν για να υπολογίσουν τις κοινότητες είναι αρκετά παρόμοιος με αυτά όπου οι αποσυνδεμένοι κόμβοι έχουν αφαιρεθεί.



Σχήμα 6.12. Απαιτούμενος χρόνος για τον υπολογισμό των κοινοτήτων των πραγματικών δεδομένων.

Έχοντας αναλύσει τα αποτελέσματα των πραγματικών δεδομένων σε αυτή την ενότητα αλλά και τα αποτελέσματα που μας δίνουν οι αλγόριθμοι για τα συνθετικά σύνολα δεδομένων (LFR) μπορούμε να αντιληφθούμε πως οι αλγόριθμοι λειτουργούν σε ποικίλες περιπτώσεις. Αρχικά πρέπει να αναφέρουμε πως τα συνθετικά δεδομένα έχουν διαμορφωμένα σύνολα, βάση προκαθορισμένων απαιτήσεων στα οποία θέλουμε να δουλέψουμε. Για παράδειγμα, στα συνθετικά

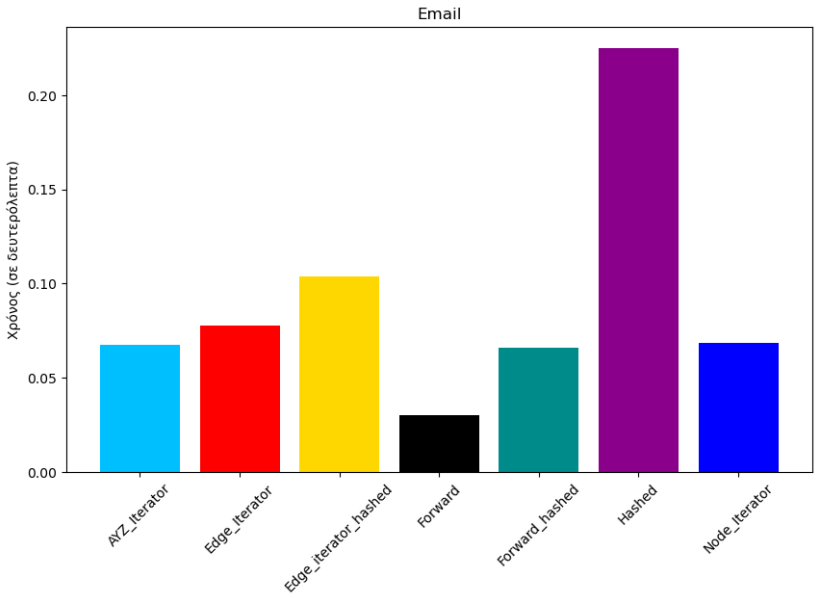
σύνολα έχουμε πάντα τις πραγματικές κοινότητες κάτι που δε συμβαίνει πάντα στο πραγματικό κόσμο (βλέπε σύνολα Facebook και GitHub), τονίζοντας τη σημασία της χρήσης άλλων δεικτών όπως του modularity. Μια ακόμη βασική διαφορά είναι ότι στα συνθετικά δεδομένα μπορούμε να επιλέξουμε το πόσο 'καλά συνδεδεμένες' είναι οι κοινότητες μεταξύ τους (χρησιμοποιώντας τη παράμετρο μίξης (mu). Συνολικά μπορούμε να δούμε πως ο αλγόριθμος Leiden είναι ταχύτερος, με την διαφορά να αυξάνεται όσο μεγαλύτερο είναι το σύνολο. Παρατηρούμε ότι ο αλγόριθμος Louvain είναι ταχύτερος σε σύγκρισή με αυτόν του Walktrap, αφού σε μικρά σύνολα δεδομένων βλέπουμε ελάχιστες διαφορές (π.χ. σύνολο Email, όπου ο Walktrap είναι ταχύτερος). Ο αλγόριθμος Walktrap φαίνεται να δίνει γενικότερα υψηλότερες τιμές NMI ειδικά όταν υπάρχει ξεκάθαρη ιεραρχική δομή (π.χ. χαμηλή τιμή παραμέτρου μίξης). Το πλήθος των κοινοτήτων που δημιουργεί ο αλγόριθμος Walktrap, φαίνεται να είναι πιο κοντά σε αυτό των πραγματικών αλλά δημιουργεί αρκετές μεμονωμένες κοινότητες. Τέλος πρέπει να αναφερθεί ότι οι αλγόριθμοι Leiden και Louvain δίνουν αρκετά υψηλές τιμές modularity, ειδικά στα πραγματικά σύνολα (βλέπε Σχήμα 6.10 και Σχήμα 6.11), το οποίο μας υποδεικνύει ότι δημιουργούν κοινότητες που είναι 'καλά συνδεδεμένες' μεταξύ τους.

6.4.2 Αξιολόγηση αλγορίθμων καταμέτρησης τριγώνων σε σύνολα πραγματικών δεδομένων

Οι αλγόριθμοι καταμέτρησης τριγώνων αξιολογήθηκαν για τα τέσσερα πραγματικά σύνολα δεδομένων που παρουσιάστηκαν παραπάνω (Email, DBLP, GitHub και Facebook). Αρχικά πρέπει να αναφερθεί πως οι αλγόριθμοι εύρεσης τριγώνων, επαναλήφθηκαν 100 φορές για το κάθε σύνολο και τα αποτελέσματα που παρουσιάζονται στα παρακάτω σχήματα (Σχήματα 6.13-6.16) αναπαριστούν τον μέσο χρόνο που χρειάστηκε ο κάθε αλγόριθμος. Στον Πίνακα 6.5 παρουσιάζονται τα στατιστικά αποτελέσματα των χρόνων εκτέλεσης, όπου μπορούμε να δούμε πως ο αλγόριθμος Forward είναι πιο σταθερός. Επίσης, αλγόριθμος Forward συνεχώς φαίνεται να είναι ο πιο αποδοτικός σε θέμα χρόνου, αφού για όλα τα δίκτυα έχει τον ταχύτερο χρόνο εκτέλεσης (βλέπε Σχήμα 6.13-6.16). Αντίθετα η μέθοδος Hashed αν και αναμέναμε να έχει γρήγορο χρόνο εκτέλεσης δεδομένου ότι αποθηκεύει τα δίκτυα σε μορφή που διευκολύνει τον υπολογιστή να τα διαβάσει. Βέβαια, η μετατροπή των συνόλων χρειάζεται χρόνο, για να εκτελεστεί, το οποίο κάνει

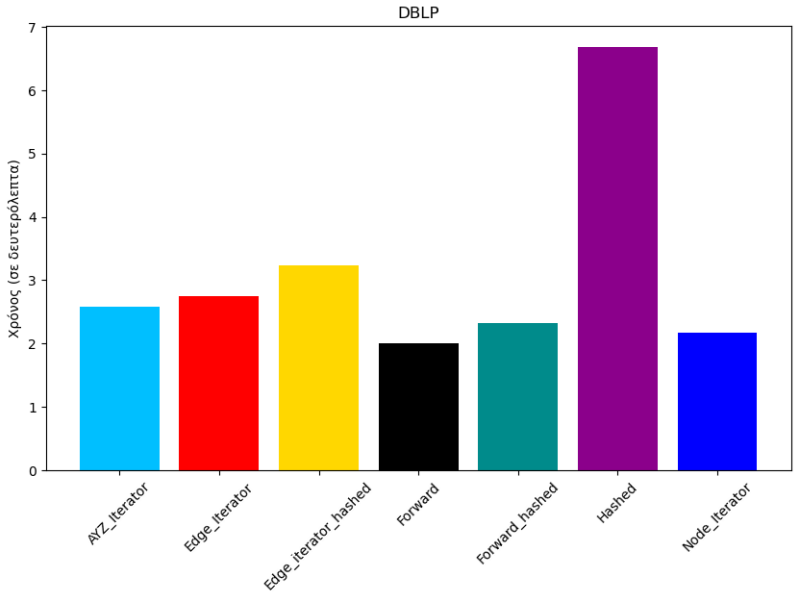
τους αλγόριθμους να μην είναι τόσο αποδοτικοί. Για παράδειγμα, η μετατροπή του συνόλου της DBLP χρειάζεται περίπου 1.6 δευτερόλεπτα και ο αλγόριθμος χρειάζεται 5.2 δευτερόλεπτα αντί για 6.8 που χρειαζόταν προηγουμένως. Αυτό σημαίνει πως και πάλι είναι πιο αργός από ότι οι λοιποί αλγόριθμοι. Το ίδιο συμβαίνει και στους υπόλοιπους αλγορίθμους Hashed μορφής, που σημαίνει ότι η μετατροπή του συνόλου δεν διευκόλυνε τους αλγορίθμους. Για μικρότερα σύνολα όπως του Facebook (βλέπε Σχήμα 6.16), οι υπόλοιποι αλγόριθμοι βλέπουμε πως δεν έχουν σημαντικές διαφορές μεταξύ τους και μπορούμε να πούμε ότι είναι το ίδιο αποτελεσματικοί. Βέβαια, ο Node-Iterator είναι λίγο πιο γρήγορος και βλέπουμε πως έχει καλύτερα αποτελέσματα ακόμη και από τον Forward στο σύνολο του DBLP (βλέπε Σχήμα 6.14).

αλγόριθμος	**δευτερόλεπτα**
Node_Iterator	0,0686
AYZ_Iterator	0,0673
Edge_Iterator	0,0776
Forward	0,0302
Hashed	0,2249
Edge_iterator_hashed	0,1037
Forward_hashed	0,0661



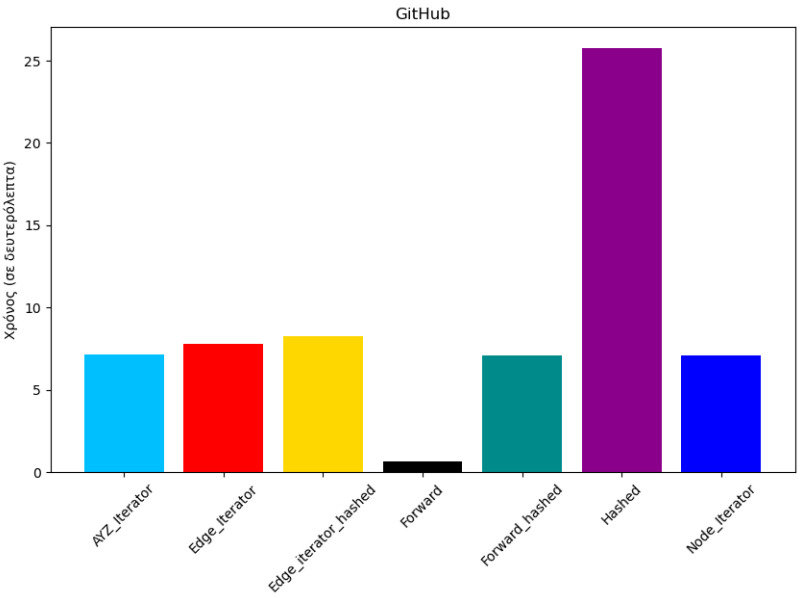
Σχήμα 6.13. Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του email.

αλγόριθμος	δευτερόλεπτα
Node_Iterator	2,1762
AYZ_Iterator	2,5783
Edge_Iterator	2,7499
Forward	2,0111
Hashed	6,6800
Edge_iterator_hashed	3,2295
Forward_hashed	2,3242



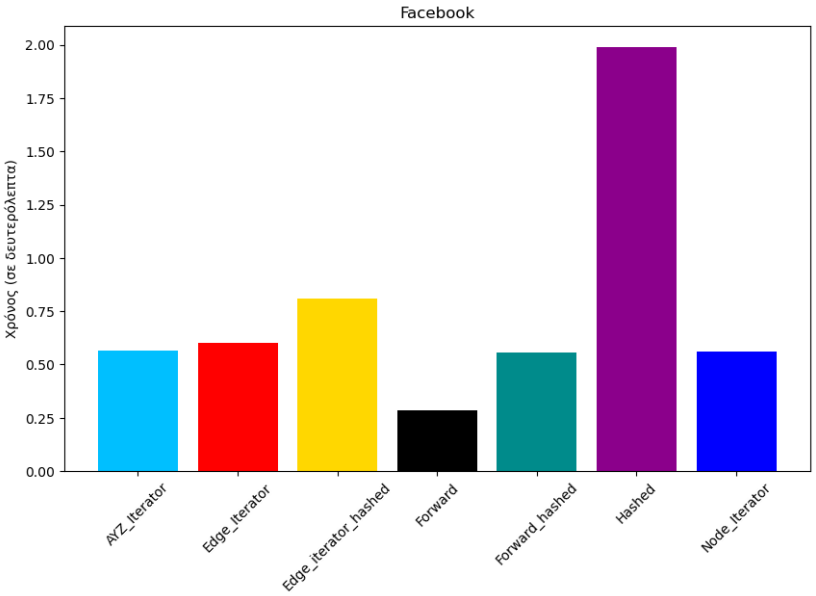
Σχήμα 6.14. Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του DBLP.

αλγόριθμος	δευτερόλεπτα
Node_Iterator	7,0665
AYZ_Iterator	7,1271
Edge_Iterator	7,7854
Forward	0,6677
Hashed	25,7440
Edge_iterator_hashed	8,2291
Forward_hashed	7,0687



Σχήμα 6.15. Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του GitHub.

αλγόριθμος	δευτερόλεπτα
Node_Iterator	0,5613
AYZ_Iterator	0,5666
Edge_Iterator	0,6025
Forward	0,2870
Hashed	1,9889
Edge_iterator_hashed	0,8072
Forward_hashed	0,5580



Σχήμα 6.16. Μέσος χρόνος εκτέλεσης αλγορίθμων για την εύρεση τριγώνων στο σύνολο δεδομένων του Facebook.

Σύνολο δεδομένων	Αλγόριθμος	Μέσος χρόνος	Ελάχιστος χρόνος	Μέγιστος χρόνος	Τυπική απόκλιση	Διακύμανση
DBLP	Node_Iterator	2,1762	2,0724	3,3989	0,267	0,0715
DBLP	AYZ_Iterator	2,5783	2,4693	3,8734	0,275	0,0755
DBLP	Edge_Iterator	2,7499	2,6136	4,4998	0,335	0,1122
DBLP	Forward	2,0111	1,8330	3,1874	0,242	0,0583
DBLP	Hashed	6,6800	6,3669	10,1112	0,645	0,4166
DBLP	Edge_iterator_hashed	3,2295	3,0960	5,0485	0,358	0,1282
DBLP	Forward_hashed	2,3242	2,2221	3,5279	0,219	0,0482
Email	Node_Iterator	0,0686	0,0614	0,0797	0,007	0,0001
Email	AYZ_Iterator	0,0673	0,0612	0,0812	0,007	0,0001
Email	Edge_Iterator	0,0776	0,0625	0,1096	0,005	0,0000
Email	Forward	0,0302	0,0156	0,0424	0,004	0,0000
Email	Hashed	0,2249	0,2170	0,3227	0,015	0,0002
Email	Edge_iterator_hashed	0,1037	0,0897	0,1107	0,008	0,0001
Email	Forward_hashed	0,0661	0,0609	0,0795	0,006	0,0000
Facebook	Node_Iterator	0,5613	0,5400	0,8305	0,033	0,0011
Facebook	AYZ_Iterator	0,5666	0,5463	0,8816	0,035	0,0012
Facebook	Edge_Iterator	0,6025	0,5780	1,0800	0,050	0,0025
Facebook	Forward	0,2870	0,2684	0,4383	0,017	0,0003
Facebook	Hashed	1,9889	1,9263	2,8244	0,111	0,0123
Facebook	Edge_iterator_hashed	0,8072	0,7895	1,1736	0,039	0,0015
Facebook	Forward_hashed	0,5580	0,5357	0,8303	0,030	0,0009
GitHub	Node_Iterator	7,0665	6,9655	8,6576	0,238	0,0565
GitHub	AYZ_Iterator	7,1271	6,9688	8,0874	0,194	0,0375
GitHub	Edge_Iterator	7,7854	7,6187	10,6429	0,350	0,1222
GitHub	Forward	0,6677	0,5946	0,8782	0,073	0,0053
GitHub	Hashed	25,7440	25,2717	36,8295	1,189	1,4141
GitHub	Edge_iterator_hashed	8,2291	8,1003	9,6109	0,215	0,0462
GitHub	Forward_hashed	7,0687	6,9633	7,9877	0,171	0,0294

Πίνακας 6.5. Στατιστικά χρόνων εκτέλεσης των αλγορίθμων καταμέτρησης τριγώνων για τα πραγματικά σύνολα δεδομένων.

Βιβλιογραφία

Alon, N., Yuster, R., & Zwick, U. (1994). Finding and counting given length cycles. In Algorithms—ESA'94: Second Annual European Symposium Utrecht, The Netherlands, September 26–28, 1994 Proceedings 2 Springer Berlin Heidelberg, 354-364.

Bader, D. A. (2023). Fast Triangle Counting. In 2023 IEEE High Performance Extreme Computing Conference (HPEC) IEEE, 1-6.

Bae, S. H., Halperin, D., West, J. D., Rosvall, M., & Howe, B. (2017). Scalable and efficient flow-based community detection for large-scale graph analysis. ACM Transactions on Knowledge Discovery from Data (TKDD), 11(3), 1-30.

Becchetti, L., Boldi, P., Castillo, C., & Gionis, A. (2008). Efficient semi-streaming algorithms for local triangle counting in massive graphs. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 16-24.

Berry, J. W., Hendrickson, B., LaViolette, R. A., & Phillips, C. A. (2011). Tolerating the community detection resolution limit with edge weighting. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 83(5), 056119.

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10), P10008.

Boldi P, & Sebastiano Vigna (2014). Axioms for centrality. Internet Mathematics 10.3-4.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2006). Maximizing modularity is hard. arXiv preprint physics/0608255.

Brandes, U., Gaertler, M., & Wagner, D. (2003). Experiments on graph clustering algorithms. In European symposium on algorithms . Berlin, Heidelberg: Springer Berlin Heidelberg, 568-579.

Bron, C., & Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM, 16(9), 575-577.

Chiba, N., & Nishizeki, T. (1985). Arboricity and subgraph listing algorithms. SIAM Journal on computing, 14(1), 210-223. 12.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. Physical review E, 70(6), 066111.

Clauset, A., Shalizi, C. R., & Newman, M. E. (2009). Power-law distributions in empirical data. SIAM review, 51(4), 661-703.

Condon, A., & Karp, R. M. (2001). Algorithms for graph partitioning on the planted partition model. Random Structures & Algorithms, 18(2), 116-140.

Coppersmith, D., & Winograd, S. (1987). Matrix multiplication via arithmetic progressions. In Proceedings of the nineteenth annual ACM symposium on Theory of computing (pp. 1-6).

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (2004). Clustering large graphs via the singular value decomposition. Machine learning, 56, 9-33.

Duch, J., & Arenas, A. (2005). Community detection in complex networks using extremal optimization. Physical review E, 72(2), 027104.

Eurostat 6% of young people in the EU uses the internet daily https://ec.europa.eu/eurostat/web/products-eurostat-news/w/ddn-20230714-1

Erdos, P., & Rényi, A. (1960). On the evolution of random graphs. Publ. math. inst. hung. acad. sci, 5(1), 17-60.

Farkas, I., Ábel, D., Palla, G., & Vicsek, T. (2007). Weighted network modules. New Journal of Physics, 9(6), 180.

Fortunato, S. (2010). Community detection in graphs. Physics reports, 486(3-5), 75-174.

Fortunato, S., & Barthelemy, M. (2007). Resolution limit in community detection. Proceedings of the national academy of sciences, 104(1), 36-41.

Fortunato, S., Latora, V., & Marchiori, M. (2004). Method to find community structures based on information centrality. Physical review E, 70(5), 056104.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the national academy of sciences, 99(12), 7821-7826.

Hagberg, A., Swart, P. J., & Schult, D. A. (2008). Exploring network structure, dynamics, and function using NetworkX (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

Javaid A. (2013) Understanding Dijkstra Algorithm. SSRN Electronic Journal

Jeh, G., & Widom, J. (2002). Simrank: a measure of structural-context similarity. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 538-543.

Jure, L. (2014). SNAP Datasets: Stanford large network dataset collection. Retrieved December 2021 http://snap. stanford. edu/data.

Lancichinetti, A., & Fortunato, S. (2009). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 80(1), 016118, 3-4.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 78(4), 046110.

Latora, V., & Marchiori, M. A measure of centrality based on the network efficiency (2004).

Luyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & Jupyter Development Team. (2016). Jupyter notebooks – A publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), Positioning and power in academic publishing: Players, agents and agendas. IOS Press, 87–90 https://doi.org/10.3233/978-1-61499-649-1-87

Mahmoudi, A., & Jemielniak, D. (2024). Proof of biased behavior of Normalized Mutual Information. Scientific Reports, 14(1), 9021

Newman, M. E. (2004). Fast algorithm for detecting community structure in networks. Physical review E, 69(6), 066133.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. Physical review E, 69(2), 026113.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. nature, 435(7043), 814-818.

Pokharel, M. (2020). Computational Complexity Theory (P, NP, NP-Complete and NP-Hard Problems). Tribhuvan University, Nepal.

Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. In Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20 . Springer Berlin Heidelberg, 284-293.

Rajaraman, A. (2011). Mining of massive datasets.

Rochat, Y. (2009) Closeness centrality extended to unconnected graphs: the harmonic centrality index. Applications of Social Network Analysis, ASNA.

Schank, T. (2007). Algorithmic aspects of triangle-based network analysis (Doctoral dissertation). Fakultät für Informatik, Universität Fridericiana zu Karlsruhe (TH), Karlsruhe, Germany.

Schank, T., & Wagner, D. (2005). Approximating clustering coefficient and transitivity. Journal of Graph Algorithms and Applications, 9(2), 265-275.

Schank, T., & Wagner, D. (2005). Finding, counting and listing all triangles in large graphs, an experimental study. In International workshop on experimental and efficient algorithms . Berlin, Heidelberg: Springer Berlin Heidelberg, 606-609.

Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical computer science, 363(1), 28-42.

Traag, V. A., Van Dooren, P., & Nesterov, Y. (2011). Narrow scope for resolution-limit-free community detection. Physical Review E, 84(1), 016114

Traag, V. A., Waltman, L., & Van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. Scientific reports, 9(1), 5233.

Venkatesaramani, R., & Vorobeychik, Y. (2018). Community detection by information flow simulation. arXiv preprint arXiv:1805.04920.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. nature, 393(6684), 440-442.

Παράρτημα

Όπως αναφέραμε στο πέμπτο κεφάλαιο ο αλγόριθμοι εύρεσης κοινοτήτων και κυρίως ο αλγόριθμος Louvain μπορεί να δημιουργήσουν κοινότητες οι οποίες δεν είναι καλά συνδεμένες μεταξύ τους. Ως μια μη καλά συνδεμένη κοινότητα ορίζεται μια κοινότητα στην οποία ένας κόμβος δεν μπορεί να 'μεταφερθεί' σε κόμβο της ίδιας κοινότητας, όπου στο μονοπάτι που θα χρησιμοποιήσει, δεν υπάρχει κόμβος μια άλλης κοινότητας. Για τον λόγο αυτό 'τρέξαμε' τους αλγορίθμους εύρεσης κοινοτήτων για τα σύνολα πραγματικών δεδομένων ώστε να βρούμε το ποσοστό αποσυνδεμένων κοινοτήτων που δημιουργούν οι αλγόριθμοι (βλέπε πίνακα 6.4). Βάσει των αποτελεσμάτων παρατηρήθηκε πως οι αλγόριθμοι Leiden και Walktrap δεν παρουσιάζουν αποσυνδεμένες κοινότητες. Παρακάτω παρουσιάζονται τα αποτελέσματα των δέκα επαναλήψεων για τον αλγόριθμο Louvain.

Δίκτυο/ Επανάληψη		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DBLP	Πλήθος κοινοτήτων	211	184	199	221	212	238	206	219	213	216
	Αποσυνδεμένες	8	11	10	5	8	5	8	3	5	7
Email	Πλήθος κοινοτήτων	7	8	8	7	8	7	8	8	8	7
	Αποσυνδεμένες	0	0	0	0	0	0	0	0	0	0
Facebook	Πλήθος κοινοτήτων	16	16	16	16	16	16	15	16	15	16
	Αποσυνδεμένες	0	0	0	0	0	0	0	0	0	0
GitHub	Πλήθος κοινοτήτων	46	40	27	21	45	29	34	36	42	45
	Αποσυνδεμένες	1	4	2	1	4	3	2	2	3	1

Πίνακας 10 επαναλήψεων αλγορίθμου Louvain σε πραγματικά δεδομένα