Δημοκρίτειο Πανεπιστήμιο Θράκης
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Παραγωγής και Διοίκησης

Επιβλέπων: Στέφανος Κατσαβούνης, Αναπληρωτής Καθηγητής

Αλγόριθμος Ανίχνευσης Κοινοτήτων Κοινωνικών Δικτύων μέσω Δυαδικών Δένδρων με Νήματα σε Python 3.8

Διπλωματική Εργασία
Κοτσίνας Γεώργιος
Α.Μ. 91674

Ξάνθη, Ιούνιος 2021

Διπλωματική εργασία η οποία υποβλήθηκε τον Ιούνιο του 2021 για την απόκτηση του διπλώματος του Μηχανικού Παραγωγής & Διοίκησης

ΕΥΧΑΡΙΣΤΙΕΣ

Με αφορμή το πέρας των σπουδών μου με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στους ανθρώπους οι οποίοι συνέβαλαν σε αυτήν την προσπάθεια.

Αρχικά, στον επιβλέποντα καθηγητή, κύριο Στέφανο Κατσαβούνη, Αναπληρωτή Καθηγητή του Τμήματος Μηχανικών Παραγωγής και Διοίκησης του Δ.Π.Θ., για την καθοδήγηση και την οργάνωση της διπλωματικής εργασίας, καθώς επίσης για την πολύτιμη βοήθεια και την άψογη συνεργασία μας. Ακόμα και σε αυτούς τους δύσκολους καιρούς που βιώσαμε, η ανταπόκρισή του ήταν πάντα άμεση και αποτελεσματική οποιαδήποτε στιγμή. Επίσης η συνεχή υποστήριξή του ήταν καταλυτική στην ολοκλήρωση των σπουδών μου.

Τη μεγαλύτερη ευγνωμοσύνη όμως οφείλω στην οικογένειά μου που με βοήθησε με κάθε δυνατό τρόπο και μου παρείχε όλα τα υλικά και πνευματικά εφόδια προκειμένου να εξασφαλίσει την καλύτερη δυνατή διαπαιδαγώγηση μου.

Κοτσίνας Γεώργιος

Ξάνθη, Ιούνιος 2021

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ iii

ΕΙΚΟΝΕΣ, ΠΙΝΑΚΕΣ & ΕΞΙΣΩΣΕΙΣ vii

ΠΕΡΙΛΗΨΗ ix

ABSTRACT xi

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ 1

ΚΕΦΑΛΑΙΟ 2. ΚΟΙΝΩΝΙΚΑ ΔΙΚΤΥΑ, ΘΕΩΡΙΑ ΓΡΑΦΩΝ & ΑΝΙΧΝΕΥΣΗ ΚΟΙΝΟΤΗΤΩΝ 3

2.1 ΕΙΣΑΓΩΓΗ & ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΚΟΙΝΩΝΙΚΩΝ ΔΙΚΤΥΩΝ 3

2.1.1 Εισαγωγή 3

2.1.2 Χαρακτηριστικά των Κοινωνικών Δικτύων 3

2.2 ΘΕΩΡΙΑ ΓΡΑΦΩΝ 6

2.2.1 Γράφος 6

2.2.2 Χαρακτηριστικά γραφημάτων 6

2.3 ΑΝΙΧΝΕΥΣΗ ΚΟΙΝΟΤΗΤΩΝ 9

2.3.1 Βιβλιογραφική Ανασκόπηση 9

2.3.2 Κριτήρια συμμετοχής σε κοινότητα 13

ΚΕΦΑΛΑΙΟ 3 ΔΙΑΤΥΠΩΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ & ΑΛΓΟΡΙΘΜΙΚΗ ΕΠΙΛΥΣΗ 17

3.1 ΟΡΙΣΜΟΣ ΚΑΙ ΜΑΘΗΜΑΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ 17

3.1.1 Βασικά Χαρακτηριστικά 17

3.1.2 Κριτήρια συμμέτοχης σε κοινότητα 18

3.2 ΑΛΓΟΡΙΘΜΟΣ ΑΝΙΧΝΕΥΣΗΣ ΚΟΙΝΟΤΗΤΩΝ 19

3.2.1 Παραγωγή δυαδικών δέντρων με νήματα (threated binary tree generation) 20

3.2.2 Ανάλυση διαδρομής 24

ΚΕΦΑΛΑΙΟ 4 ΥΠΟΛΟΓΙΣΤΙΚΗ ΕΜΠΕΙΡΙΑ 29

4.1 ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ 29

4.1.1 Δομές δεδομένων & υλοποίηση με python 29

4.1.2 Υλοποίηση Με Αναδρομή 34

4.1.3 Υλοποίηση Χωρίς Αναδρομή 35

4.2 ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ 39

4.2.1 Δημιουργία γράφου με οριοθετημένες κοινότητες 39

4.2.2 Υπολογισμός βαρών ακμών 39

4.2.3 Κοινότητα στόχος 40

4.3 ΠΕΡΙΠΤΩΣΕΙΣ ΜΕΛΕΤΗΣ 41

4.3.1 Εκτελέσεις 41
4.3.2 Αποτελέσματα 42
4.3.3 Χρόνος εκτέλεσης 46
ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ, ΕΦΑΡΜΟΓΕΣ ΚΑΙ ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ 49
5.1 ΣΥΜΠΕΡΑΣΜΑΤΑ 49
5.1.1 Εφαρμογές στο marketing 49
5.1.2 Εφαρμογές ανίχνευσης κοινότητας 51
5.2 ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ 53
ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ 54
1. main.py 54
2. tree traversals.py 58
3. tree.py 62
4. test.py 64
ΒΙΒΛΙΟΓΡΑΦΗΚΕΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΕΣ ΑΝΑΦΟΡΕΣ 66
Βιβλιογραφία 66

ΕΙΚΟΝΕΣ, ΠΙΝΑΚΕΣ & ΕΞΙΣΩΣΕΙΣ

ΕΙΚΟΝΕΣ

Εικόνα 1 Κατευθυνόμενο, μη κατευθυνόμενο γραφημα 7
Εικόνα 2 - Συνεκτικός, μη συνεκτικός Γράφος 8
Εικόνα 3 - Διάφοροι τύποι γραφημάτων 8
Εικόνα 4 - Παράδειγμα αναπαράστασης κοινωνικού δικτύου ως Γράφος 14
Εικόνα 5 - Δυαδικά δένδρα με νήματα των κόμβων Β, Λ, Μ, Ι, Κ, Γ 22
Εικόνα 6 - Διάγραμμα ροής παραγωγής διαδίκων δένδρων με νήματα 24
Εικόνα 7 - Διάγραμμα ροής ανάλυσης διαδρομής 26
Εικόνα 8 - Δυαδικό δένδρο 30
Εικόνα 9 - Δυαδικό δένδρο με νήματα 30
Εικόνα 10 - Αντικείμενα-κόμβοι του δυαδικού δένδρου με νήματα και ριζικό κόμβο Β 32
Εικόνα 11 – Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλίσης για δίκτυο 333 κόμβων 37
Εικόνα 12- Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλίσης για δίκτυο 6.551 κόμβων 38
Εικόνα 13 - Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλίσης για δίκτυο 22.470 κόμβων 38
Εικόνα 14 - Διάγραμμα αριθμού αποτελεσμάτων ανά εκτέλεση για δίκτυο 224 κόμβων 43
Εικόνα 15 - Διάγραμμα αριθμού αποτελεσμάτων ανά εκτέλεση για δίκτυο 6.551 κόμβων 44
Εικόνα 16 - Διάγραμμα χρόνων εκτελέσεων μικρών δικτύων 46
Εικόνα 17 - Διάγραμμα χρόνων εκτελέσεων μεσαίων δικτύων 47
Εικόνα 18 - Διάγραμμα χρόνου εκτελέσεων μεγάλων δικτύων 47

ΠΙΝΑΚΕΣ

Πίνακας 1 - Αναδρομική και μη κλίση συναρτήσεων τριών δικτύων 37
Πίνακας 2 - Κατάλογος δικτιών 39
Πίνακας 3 - Δεκαψήφια δυαδική τιμή του κόμβου 123 40
Πίνακας 4 - Διαδοχικές εκτελέσεις 42
Πίνακας 5 – Χαρακτηριστικά δικτύου 224 κόμβων 43
Πίνακας 6 - Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (224 κόμβων) ... 43
Πίνακας 7 - Χαρακτηριστικά δικτύου 6.551 κόμβων 44
Πίνακας 8 - Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (6.551 κόμβων) . 44
Πίνακας 9 - Χαρακτηριστικά δικτύου 6.551 κόμβων 45
Πίνακας 10 - Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (6.551 κόμβων) 45

ΠΕΡΙΛΗΨΗ

Αναμφισβήτητα διανύουμε την εποχή μια νέας οικουμενικής ψηφιακής πολιτείας, όπου τα κοινωνικά δίκτυα μετρούν δισεκατομμύρια χρήστες παγκοσμίως. Η ανάπτυξη και η αυξανόμενη δημοτικότητα των μέσων κοινωνικής δικτύωσης επηρεάζει καθοριστικά την καταναλωτική, πολιτική και ψυχαγωγική συμπεριφορά των χρηστών, αποτελώντας έτσι αντικείμενο μελέτης της σύγχρονης επιστημονικής κοινότητας.

Η παρούσα διπλωματική εργασία πραγματεύεται τη μελέτη των κοινωνικών δικτύων μέσω της θεωρίας γραφημάτων. Αρκετές σύγχρονες εφαρμογές βασίζονται στα γραφήματα δεδομένων. Ανάμεσά τους, μια πολύ σημαντική εφαρμογή αυτού του είδους είναι η ανίχνευση κοινοτήτων. Δεδομένου ότι ο αριθμός και το μέγεθος των δικτύων που μοντελοποιούνται από γραφήματα αυξάνονται όλο και περισσότερο, πρέπει να χρησιμοποιηθούν μέθοδοι οι οποίες μειώνουν το υπολογιστικό κόστος αυτού του τεράστιου όγκου δεδομένων, με την ανάπτυξη αποτελεσματικών εφαρμογών.

Οι ιστότοποι κοινωνικής δικτύωσης επιτρέπουν στους χρήστες να κατηγοριοποιούν με μη αυτόματο τρόπο τους φίλους τους σε κοινωνικούς κύκλους, ενώ οι χρήστες, βάσει των ενδιαφερόντων τους, τοποθετούνται σε ομάδες ενδιαφέροντος.

Αυτή η εργασία, συνδυάζει τεχνικές επεξεργασίας δεδομένων για εξοικονόμηση υπολογιστικού χρόνου με τυπικές δομές δεδομένων, όπως τα δυαδικά δένδρα με νήματα για τον εντοπισμό των κοινοτήτων με αποτελεσματικό τρόπο. Η στρατηγική αυτή εφαρμόστηκε σε σταθμισμένα δίκτυα πραγματικού κόσμου με ακανόνιστες τοπολογίες με τη βοήθεια της Python 3.8.

Τα αποτελέσματα της εφαρμογής του αλγορίθμου που αναπτύχθηκε με δεδομένα των δημοφιλέστερων κοινωνικών δικτύων καταδεικνύει τη σπουδαιότητα της εφαρμογής της θεωρίας γραφημάτων σε μεγάλου όγκου δεδομένα και την αναγκαιότητα ανάπτυξης αποτελεσματικών αλγορίθμων για την αντιμετώπιση των προκλήσεων που δημιουργεί η σύγχρονη ψηφιακή πραγματικότητα.

Λέξεις – Κλειδιά: κοινωνικό δίκτυο, θεωρία γραφημάτων, σταθμισμένα γραφήματα δικτύων, αλγόριθμος ανίχνευσης κοινότητας, δυαδικά δένδρα με νήματα

ABSTRACT

Title: Community Detection Algorithm for Social Networks using Threaded Binary Trees implemented in Python 3.8

Undoubtedly, we are entering the era of a new universal digital state, where social networks count billions of users worldwide. The development and growing popularity of social media has a decisive influence on the consumer, political and entertainment behavior of users, thus being the subject of study of the modern scientific community.

This dissertation deals with the study of social networks through graph theory. Many modern applications are based on data graphs. Among them, a very important application of this kind is the community detection. As the number and size of networks modeled by graphs increase, methods that reduce the computational cost of this vast amount of data must be used by developing efficient applications.

This work combines data processing techniques to save computing time with standard data structures, such as threaded binary trees to locate communities effectively. This strategy was implemented in weighted real-world networks with irregular topologies with the help of Python 3.8.

The results of the application of the algorithm developed with data from the most popular social networks demonstrate the importance of the application of graph theory to large volumes of data and the need to develop effective algorithms to meet the challenges posed by modern digital reality.

Keywords: social network, graph theory, weighted graph networks, community detection algorithm, Threaded binary trees

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Το φαινόμενο της διαδικτυακής κοινωνικής αλληλεπίδρασης είναι σταθερά επίκαιρο λόγω της δυναμικής του εξέλιξης, και φαίνεται να έχει ξεπεράσει την παροδική διάσταση μιας μόδας τείνοντας να παγιωθεί σαν ένα κοινωνικό φαινόμενο, το οποίο βρίσκεται ακόμα σε φάση εξέλιξης. Όπως κάθε πρωτόγνωρο κοινωνικό φαινόμενο, με προεκτάσεις οικονομικές, πολιτικές, προσωπικές, η ηλεκτρονική κοινωνική δικτύωση εγείρει ανησυχίες, απορίες, διλήμματα και για το λόγο αυτό έχει γίνει αντικείμενο έντονης παρατήρησης και συζήτησης από την Επιστημονική Κοινότητα.

Τα μέσα κοινωνικής δικτύωσης (Social Media), μέσω των οποίων επιτυγχάνεται η ηλεκτρονική κοινωνική δικτύωση, αποτελούν απόρροια του Web 2.0, το οποίο κατάφερε να αλλάξει την υφή του Διαδικτύου προσδίδοντάς του μια πιο κοινωνική διάσταση. Κάποιοι τα χαρακτηρίζουν σαν ψηφιακά καφενεία, άλλοι τα αντιμετωπίζουν σαν ένα αθώο κουτσομπολιό, ενώ για κάποιους αποτελούν μια νέα εξουσία. Τα Social Media δεν είναι τίποτα παραπάνω από τη φυσική μετεξέλιξη των παραδοσιακών μέσων μαζικής ενημέρωσης και επικοινωνίας, τα οποία υιοθετήθηκαν από την τεχνολογική πρόοδο. Ιστοσελίδες όπως Wikipedia, Facebook, YouTube, Twitter, έχουν καταφέρει να αποτελούν αναπόσπαστο μέρος της καθημερινότητας των ανθρώπων παγκοσμίως.

Τα κοινωνικά δίκτυα κατάφεραν να μετατρέψουν το «μονόλογο» των παραδοσιακών μέσων ενημέρωσης και επικοινωνίας σε έναν ευρύτερο διάλογο, προσέφεραν διαδραστικότητα και άμεση αλληλεπίδραση, ενώ παράλληλα μείωσαν τις γεωγραφικές αποστάσεις κάνοντας τον κόσμο να φαντάζει μικρότερος και φέρνοντας τους ανθρώπους πιο κοντά από ποτέ. Αποτελούν πηγές πληροφόρησης για τους χρήστες και συνδέονται με ένα ευρύ φάσμα θετικών επιρροών όπως η ενθάρρυνση της συζήτησης, των σχολίων, της ανταλλαγής και της διάχυση πληροφοριών από όλα τα ενδιαφερόμενα μέρη, η παροχή κοινωνικής και συναισθηματικής υποστήριξης, ενώ συμβάλλουν στην ενίσχυση του κοινωνικού κεφαλαίου. Έτσι τα Social Media σήμερα αποτελούν σημαντικούς καταλύτες κοινωνικής, οικονομικής, πολιτικής και πολιτιστικής αλλαγής.

Διάφορα συστήματα υψηλού ενδιαφέροντος για την επιστημονική κοινότητα μπορούν να καταγραφούν και να μελετηθούν ως δίκτυα. Παραδείγματα αποτελούν το Διαδίκτυο (παγκόσμιος ιστός) και τα κοινωνικά δίκτυα όπως το Facebook (2,85 δισεκατομμύρια χρήστες), το Instagram (1,074 δισεκατομμύρια χρήστες) ή το Twitter (363 εκατομμύρια χρήστες). Είναι σαφές ότι αυτά τα δίκτυα είναι τόσο τεράστια, ώστε η μοντελοποίηση και η μελέτη τους μέσω γραφημάτων που περιλαμβάνουν πολύ μεγάλο αριθμό κόμβων και ακμών μπορεί να είναι χρονοβόρα και δαπανηρή. Επόμενος, η επεξεργασία τέτοιων δεδομένων αποτελεί μια μεγάλη πρόκληση και ένα θέμα μεγάλου ενδιαφέροντος για τη σύγχρονη βιβλιογραφία.

Το θέμα που πραγματεύεται η παρούσα εργασία αφορά την αναγνώριση της ταυτότητας των χρηστών μιας κοινότητας ενός κοινωνικού δικτύου που παρουσιάζεται και αναγράφεται ως κοινοτική ανίχνευση. Η έρευνα για την ανίχνευση της κοινότητας έχει μεγάλο ενδιαφέρον και αποτελεί ισχυρό κίνητρο για τη διερεύνησή του επειδή οι παραδοσιακοί αλγόριθμοι ανίχνευσης κοινότητας αποτυγχάνουν να κλιμακωθούν στον αυξανόμενο αριθμό χρηστών και στο πλήθος και την πολυπλοκότητα των σχέσεων τους. Απαιτείται πλέον αποτελεσματική μελέτη για την ανάλυση και ακόμη και την πρόβλεψη της συμπεριφοράς του χρήστη. Αυτός ο τύπος ανάλυσης έχει μεγάλη σημασία για οργανισμούς και εταιρείες,

υπεύθυνες για τον προγραμματισμό των πολιτικών μάρκετινγκ ή διαφήμισης ή για τα πολιτικά κόμματα που παρακολουθούν τις απόψεις των χρηστών των κοινωνικών δικτύων

Με την έκρηξη των κοινωνικών δικτύων, το πρόβλημα της κοινοτικής ανίχνευσης έχει γίνει αρκετά δύσκολο. Το αυξανόμενο μέγεθος των κοινωνικών δικτύων περιορίζει την ικανότητα γρήγορης επεξεργασίας των αντίστοιχων γραφημάτων. Οι υπολογισμοί που εκτελούνται θα πρέπει να οργανώνονται πολύ προσεκτικά λόγω των τεράστιων όγκων δεδομένων που υποβάλλονται σε επεξεργασία από τα κοινοτικά σχήματα ανίχνευσης. Σ' αυτήν την εργασία παρουσιάζεται μια στρατηγική κοινοτικής ανίχνευσης για κοινωνικά δίκτυα αναπαριστόμενα ως γράφοι. Όσο αναφορά την ορολογία της διπλωματικής εργασίας ο όρος γράφος ή γράφημα αναφέρεται στο ίδιο περιεχόμενο και χρησιμοποιείται για την απεικόνιση, μελέτη των κοινωνικών δικτύων.

Η παρούσα διπλωματική εργασία αποτελείται από πέντε κεφάλαια.

1. Το πρώτο κεφάλαιο αποτελεί την εισαγωγή πάνω στο αντικείμενο που πραγματεύεται η εργασία.

2. Το δεύτερο κεφάλαιο αφορά τα κοινωνικά δίκτυα και τα βασικά χαρακτηριστικά της θεωρίας γραφημάτων. Το κεφάλαιο ολοκληρώνεται με τον προσδιορισμό των εννοιών που χαρακτηρίζουν την περιοχή της ανίχνευσης κοινοτήτων και τη βιβλιογραφική ανασκόπηση του αντικειμένου.

3. Το τρίτο κεφάλαιο παρουσιάζει τη μαθηματική μοντελοποίηση του προβλήματος, περιλαμβάνοντας τον καθορισμό των παραμέτρων, των μεταβλητών και των κριτηρίων που χρησιμοποιούνται στην αλγοριθμική διατύπωση της μεθοδολογίας που εφαρμόζει η διπλωματική εργασία.

4. Το τέταρτο κεφάλαιο περιγράφει τις δομές δεδομένων που υλοποιήθηκαν στην Python 3.8. Στη συνέχεια περιγράφεται η κλασική διαδικασία αναδρομικής κλήσης συναρτήσεων για τη δημιουργία και προσπέλαση των δυαδικών δένδρων. Ακολουθεί η κύρια υλοποίηση που βελτιώνει σημαντικά τον υπολογιστικό χρόνο. Οι επόμενες δύο ενότητες του κεφαλαίου αφιερώνονται στις εκτελέσεις του κώδικα για πραγματικά δεδομένα και στην παρουσίαση της υπολογιστικής εμπειρίας που αποκτήθηκε, παράλληλα με την παράθεση των αποτελεσμάτων των δύο προαναφερθέντων υλοποιήσεων.

5. Το πέμπτο κεφάλαιο αποτυπώνει τα σχετικά συμπεράσματα και τις εφαρμογές της προτεινόμενης μεθοδολογίας ανίχνευσης κοινοτήτων σε γενικότερο πλαίσιο.

ΚΕΦΑΛΑΙΟ 2. ΚΟΙΝΩΝΙΚΑ ΔΙΚΤΥΑ, ΘΕΩΡΙΑ ΓΡΑΦΩΝ & ΑΝΙΧΝΕΥΣΗ ΚΟΙΝΟΤΗΤΩΝ

2.1 ΕΙΣΑΓΩΓΗ & ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΚΟΙΝΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

2.1.1 Εισαγωγή

Τα δίκτυα έχουν εδραιωθεί ως απαραίτητα στοιχεία των ανθρωπίνων δραστηριοτήτων, αφού η καθημερινότητα των ατόμων είναι συνυφασμένη με τη χρήση δικτύου. Η επιστημονική έρευνα έχει ξεκινήσει εδώ και αρκετές δεκαετίες, όμως τις τελευταίες δεκαετίες έχει καταστεί απαραίτητη, και έχει ως στόχο την εξέταση κοινωνικών, πολιτισμικών και φυσικών φαινομένων και κυρίως του Διαδικτύου. Εκατομμύρια άνθρωποι χρησιμοποιούν το Διαδίκτυο ως ένα μέσο επικοινωνίας σε καθημερινή βάση και αυτός είναι ο βασικότερος λόγος για τον οποίο το Διαδίκτυο ασκεί τόσο μεγάλη επιρροή στις κοινωνικές σχέσεις.

Ως κοινωνικά δίκτυα ορίζονται τα δίκτυα που σχηματίζονται από κάθε είδους κοινωνικές σχέσεις μεταξύ των ανθρώπων. Ο πιο συνηθισμένος τρόπος αναπαράστασης και ανάλυσης των κοινωνικών δικτύων είναι με τη μορφή γράφων. Τα κοινωνικά δίκτυα αποτελούνται από ομάδες χρηστών. Οι χρήστες εισέρχονται σε ένα δίκτυο και κατατάσσονται σε μια ομάδα δημιουργώντας ένα προφίλ, δημοσιεύοντας πληροφορίες και αλληλοεπιδρώντας με χρήστες που έχουν κοινά ενδιαφέροντα. Με αυτόν τον τρόπο δημιουργούνται οι ομάδες αποτελούμενες από χρήστες με ίδια ενδιαφέροντα. Αυτές οι ομάδες αναφέρονται στην παρούσα εργασία ως κοινότητες. Γενικά μπορεί κανείς να ορίσει ως κοινότητα μια συλλογή από κόμβους και ακμές σε κάποιο δίκτυο οι οποίοι είναι συνδεδεμένα με τέτοιο τρόπο ώστε οι εσωτερικές συνδέσεις της κοινότητας να είναι ισχυρότερες από ότι οι εξωτερικές. Παρόλα αυτά δεν υπάρχει διεθνώς αποδεκτός ορισμός της κοινότητας. Οι κόμβοι μιας κοινότητας που αντιπροσωπεύουν τους χρήστες του δικτύου θεωρούνται παρεμφερείς μεταξύ τους και διαφορετικοί μεταξύ των υπόλοιπων κόμβων του δικτύου. Οι ακμές αντικατοπτρίζουν την ομοιότητα μεταξύ των χρηστών μιας κοινότητας ή μεταξύ των χρηστών διαφορετικών κοινοτήτων. Με διαφορετικό τρόπο, ως κοινότητα θα μπορούσε να οριστεί ένα υποδίκτυο υψηλά συσχετισμένων χρηστών. Διαφορετικά υποδίκτυα είναι συνδεδεμένα μεταξύ τους, καθώς χρήστες διαφορετικών κοινοτήτων έχουν τη δυνατότητα να συνδέονται μεταξύ τους. Με αυτή τη λογική δύο ή περισσότερες κοινότητες μπορεί να αλληλεπικαλύπτονται εν μέρει (overlapping) έχοντας ένα ή περισσότερα κοινά μέλη. Σε περιπτώσεις όπου τα κοινά μέλη είναι πάρα πολλά με ισχυρή συνδεσιμότητα οι δύο κοινότητες μπορούν να θεωρηθούν ως μία.

2.1.2 Χαρακτηριστικά των Κοινωνικών Δικτύων

Ο όρος μέσα κοινωνικής δικτύωσης (ή αλλιώς social media) αναφέρεται στα μέσα αλληλεπίδρασης και επικοινωνίας ομάδων ανθρώπων μέσω διαδικτυακών κοινοτήτων. Τα μέσα κοινωνικής δικτύωσης ορίζονται ως βασισμένες στο Διαδίκτυο (διαδικτυακές) υπηρεσίες και εμφανίζονται σε διάφορες μορφές όπως π.χ. το Facebook, το Twitter, κ.α. Τα Μέσα κοινωνικής δικτύωσης αποτελούν την κοινωνική

διάδραση μεταξύ ανθρώπων που δημιουργούν, μοιράζονται ή ανταλλάσσουν πληροφορίες και ιδέες μέσα σε εικονικές κοινότητες και δίκτυα. Θεωρείται ότι αποτελούν κυρίαρχο μέρος της καθημερινότητας των σύγχρονων ανθρώπων καθώς τους επιτρέπουν:

1. Να δημιουργήσουν ένα δημόσιο ή ημι-δημόσιο προφίλ μέσα σε ένα οριοθετημένο σύστημα
2. Να επικοινωνήσουν με μια λίστα από άλλους χρήστες με τους οποίους μοιράζονται μια μορφή σύνδεσης και
3. Να δουν και να διανείμουν τη δική τους λίστα των συνδέσεων και αυτών που φτιάχτηκαν από άλλους μέσα στο σύστημα (danah m. boyd, 2007).

Ορισμένα από τα χαρακτηριστικά των κοινωνικών δικτύων είναι ότι απλοποιούν, βελτιώνουν την ταχύτητα και το εύρος της διάδοσης των πληροφοριών (Shannon, 1948), καθώς γίνεται εφικτή η επικοινωνία από σημείο προς σημείο, αλλά και από σημείο προς πολυσημείο δηλαδή παρατηρείται μεγάλη εμβέλεια και μαζικότητα. Ακόμη, υπάρχει πρόσβαση σε αυτά από πληθώρα ηλεκτρονικών συσκευών, π.χ. από κινητά (έξυπνα τηλέφωνα), ταμπλέτες, ηλεκτρονικούς υπολογιστές.

Επιπρόσθετα χαρακτηριστικά

- Υποστηρίζουν ποικιλία των μορφών περιεχομένου, όπως κείμενο, βίντεο, φωτογραφίες, ήχο, κλπ. Πολλά από αυτά κάνουν χρήση περισσοτέρων της μίας από αυτές τις επιλογές ως προς το περιεχόμενο
- Επιτρέπουν αλληλεπιδράσεις χρησιμοποιώντας μία ή περισσότερες πλατφόρμες μέσω διαμοιρασμού και ηλεκτρονικού ταχυδρομείου
- Χαρακτηρίζονται από διαφορετικά επίπεδα εμπλοκής του χρήστη οι οποίοι μπορούν να δημιουργήσουν, να σχολιάσουν ή να παρακολουθούν σε δίκτυα κοινωνικής δικτύωσης
- Απλοποιούν, βελτιώνουν την ταχύτητα και το εύρος της διάδοσης των πληροφοριών
- Προσφέρουν ενός- προς-ένα, ενός-προς-πολλούς και πολλών προς-πολλούς επικοινωνία
- Επιτρέπουν την επικοινωνία αυτή να πραγματοποιείται είτε σε πραγματικό χρόνο ή με ασύγχρονη μορφή
- Είναι ανεξάρτητα της συσκευής: Ο χρήστης μπορεί να χρησιμοποιήσει για τη διείσδυση σε μέσα κοινωνικής δικτύωσης έναν υπολογιστή, ή κινητές συσκευές (ιδιαίτερα ταμπλέτες και έξυπνα τηλέφωνα)
- Επεκτείνουν την εμπλοκή με τρεις τρόπους: με τη δημιουργία σε πραγματικό χρόνο διαδικτυακών εκδηλώσεων, με την επέκταση σε απευθείας σύνδεση αλληλεπιδράσεων δια ζώσης εκδηλώσεων, και τελευταία με την υποστήριξη ζωντανών εκδηλώσεων

Τα μέσα κοινωνικής δικτύωσης, είναι ικανά να συντελέσουν στην προβολή αφενός της κοινής γνώμης, καθώς καθιστούν τους χρήστες δέκτες αλλά και εκδότες περιεχομένου μέσω της ανατροφοδότησης, ενώ παρέχουν κοινωνική και συναισθηματική υποστήριξη (Evans, 2008). Επιπλέον, τα Social Media αποτελούν εξαιρετικό σύμμαχο στην προώθηση συγκεκριμένων ιδεών, αξιών, ακόμη και προϊόντων (Μάρκετινγκ), ώστε επιχειρήσεις, πολιτικοί, ομάδες συμφερόντων κ.α. «μάχονται» για την αλλοίωσή τους, στο σκληρό πλαίσιο του ανταγωνισμού.

Προς επίρρωση των παραπάνω, δεν εκλείπουν φαινόμενα νεολογισμών, επεξηγηματικά: η αθέμιτη χρήση των μέσων για συναισθηματική κακοποίηση, που επιτυγχάνεται με δημοσίευση δυσμενών σχολίων που στοχεύουν στην πρόκληση θυμού ή λύπης. Επιπλέον, συχνό φαινόμενο αποτελεί η μετάδοση επιλεγμένων πληροφοριών με σκοπό τη χειραγώγηση της κοινής γνώμης και ο βομβαρδισμός διαφημιστικών μηνυμάτων, με στοχοθεσία τον αποπροσανατολισμό .

Αυτό που κάνει τα διαδικτυακά κοινωνικά δίκτυα να ξεχωρίζουν από τις υπόλοιπες διαδικτυακές υπηρεσίες είναι:

(1) Τα εξελιγμένα εργαλεία που επιτρέπουν στους χρήστες να διαμοιράζονται ψηφιακά αρχεία (π.χ. κείμενο, εικόνες και άλλα) και

(2) τα εξελιγμένα εργαλεία για την επικοινωνία και την κοινωνικοποίηση των χρηστών (RominaCachia, 2007). Οι ίδιοι συγγραφείς μάλιστα ομαδοποίησαν τα διαδικτυακά κοινωνικά δίκτυα ανάλογα με την αλληλεπίδραση και την κοινωνικοποίηση που προσφέρει η κάθε ιστοσελίδα.

Από μία εναλλακτική κατηγοριοποίηση προκύπτουν οι παρακάτω επτά τύποι κοινωνικών δικτύων:

- Κοινωνικές ειδήσεις και προτάσεις (π.χ digg.com)
- Ιστοσελίδες κοινωνικών σελιδοδεικτών (π.χ delicious.com)
- Υπηρεσίες μικροϊστολογίων (twitter)
- Συστήματα ιστολογίων (π.χ.blogger.com)
- Μέσα κοινωνικής δικτύωσης (π.χ. facebook, linkedin)
- Μέσα κοινωνικού διαμοιρασμού (π.χ. youtube, flickr)
- Wiki (π.χ. mediawiki.org)

Τα μέσα κοινωνικής δικτύωσης, συμβάλλουν επαναστατικά στην ενημέρωση για τα τρέχοντα ζητήματα, την απρόσκοπτη έκφραση απόψεων και ιδεών, ενθαρρύνοντας τη συζήτηση, την έννοια του πλουραλισμού και τη διαλογικότητα. Ακόμη, λειτουργούν επικουρικά στη σύναψη κοινωνικών δεσμών, κυρίως φιλικών. Ως αποτέλεσμα της Θεωρίας Χρήσεων και Ηθικών Ικανοποιήσεων (U and G Theory), που αναλύει τους λόγους επιλογής των μέσων από τους χρήστες, οι κινητήριες δυνάμεις είναι η ενημέρωση, η ψυχαγωγία, η κοινωνική αλληλεπίδραση και η δημιουργία προσωπικής ταυτότητας (mcQuail, 1994). Το πόρισμα καταδεικνύει την άποψη ότι τα μέσα συμβάλλουν στην ενίσχυση της επικοινωνίας μεταξύ ατόμων σε οικουμενικό (universal) επίπεδο, ενώ μέχρι πρότινος αυτό αποτελούσε προνόμιο λίγων, που ανήκαν σε συγκεκριμένη κοινωνική τάξη (danah m. boyd, 2007).

Συμπερασματικά, τα μέσα κοινωνικής δικτύωσης αναπτύσσονται με ταχύτατους ρυθμούς, ενώ κατέχουν τη δύναμη της σχηματοποίησης και της διαστρέβλωσης της κοινής γνώμης. Επομένως, μπορούν να αλλάξουν τον τρόπο σκέψης και προσέγγισης ενός θέματος και να επιφέρουν ευρύτερες κοινωνικές ανακατατάξεις, που ποτέ δεν είχαμε φανταστεί (Joinson, 2008).

2.2 ΘΕΩΡΙΑ ΓΡΑΦΩΝ

Το πρόβλημα που πραγματεύεται η παρούσα εργασία εντάσσεται στην κατηγορία των γραφημάτων. Σ' αυτήν την ενότητα αναφέρονται τα βασικά στοιχεία της θεωρίας γράφων (ή γραφημάτων). Η θεωρία γραφημάτων είναι ένα γνωστικό πεδίο των διακριτών μαθηματικών, με αρκετές εφαρμογές στην πληροφορική. Ένας σχετικά πλήρης ορισμός ορίζει πως η θεωρία γράφων είναι η μελέτη των γράφων (γραφημάτων) και των σχέσεων τους. Οι μαθηματικοί υπολογισμοί επί των γράφων υλοποιούνται με συγκεκριμένους αλγόριθμους. Με γραφήματα μπορούν να μοντελοποιηθούν πολλές διαφορετικές φυσικές ή τεχνολογικές δομές, όπως π.χ. τα δίκτυα ύδρευσης, τα οδικά δίκτυα, τα κοινωνικά δίκτυα, τα δίκτυα υπολογιστών κτλ. (J. A. Bondy, 1976)

2.2.1 Γράφος

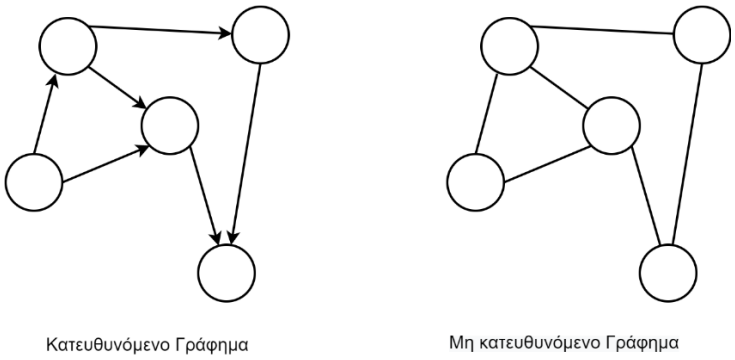
Ο Γράφος στον απλούστερο ορισμό του είναι η οπτική αναπαράσταση των σχέσεων που αναπτύσσουν ορισμένες ποσότητες, σχεδιασμένες σε σχέση με ένα σύνολο αξόνων. Ένας άλλος ορισμός που κινείται στο ίδιο εννοιολογικό πλαίσιο της οπτικής αναπαράστασης αναγνωρίζει τον Γράφο ως απεικόνιση αποτελούμενη από ένα σύνολο σημείων (κορυφών ή κόμβων) που συνδέονται με γραμμές (ακμές).

1. Οι κορυφές αποτελούν μια οντότητα και αποδίδουν κάποια αξία σε αυτήν. Κατ' αυτόν τον τρόπο οι κορυφές μπορούν να συμβολίζουν ανθρώπους, κτίρια, πόλεις, φυτά κτλ. Στην επιστήμη των δικτύων οι κορυφές συχνά αποκαλούνται κόμβοι, έτσι ο όρος κόμβος χρησιμοποιείται τυπικά στην παρούσα εργασία.
2. Οι ακμές μπορούν να οριστούν ως οι συσχετίσεις μεταξύ των κόμβων του δικτύου, αυτή η σύνδεση είναι εμφανής και απεικονίζεται στα γραφήματα με γραμμές. Οι ακμές μπορούν να είναι διακριτές, απεικονίζοντας για παράδειγμα τους δρόμους που ενώνουν τις πόλεις σε ένα οδικό δίκτυο ή ακαθόριστες όπως η φιλία μεταξύ δύο χρηστών ενός κοινωνικού δικτύου. Οι ακμές αποκαλούνται συχνά και συνδέσεις ή συσχετίσεις, ορολογία που χρησιμοποιείται στα επόμενα κεφάλαια.

2.2.2 Χαρακτηριστικά Γραφημάτων

Κατεύθυνση

Αν οι ακμές προσανατολίζονται οριζόμενες από διατεταγμένα ζεύγη κόμβων, τότε ο Γράφος αποκαλείται κατευθυνόμενος. Αλλιώς, αν οι ακμές δεν προσανατολίζονται, οριζόμενες απλώς από διμελή σύνολα και όχι διατεταγμένα ζεύγη, τότε αποκαλείται μη κατευθυνόμενος (Εικόνα 1)



Εικόνα 1. Κατευθυνόμενο, μη κατευθυνόμενο γραφημα

Η μαθηματική αποτύπωση των προηγούμενων γράφων έχει ως εξής:

Γράφος G είναι ένα διατεταγμένο ζεύγος $G = (V(G), E(G))$ όπου:

- $V(G) = \{v_1, v_2 \dots v_5\}$ το σύνολο των κορυφών,
- $E(G) = \{e_1, e_2 \dots e_6\}$ το σύνολο των ακμών.

Στην περίπτωση του μη κατευθυνόμενου γράφου, κάθε ακμή είναι ένα διμελές σύνολο αποτελούμενο από δύο κορυφές, οι οποίες αποκαλούνται τερματικές κορυφές (κόμβοι) και δεν είναι απαραίτητα διαφορετικές μεταξύ τους. Στους κατευθυνόμενους Γράφους κάθε ακμή είναι ένα διατεταγμένο ζεύγος αποτελούμενο από δύο κορυφές, οι οποίες αποκαλούνται τερματικές κορυφές (κόμβοι) και δεν είναι απαραίτητα διαφορετικές μεταξύ τους. Η διαφορά ανάμεσα σε έναν μη κατευθυνόμενο και έναν κατευθυνόμενο γράφο είναι ότι στην πρώτη περίπτωση έχουμε διμελές σύνολο, ενώ στη δεύτερη διατεταγμένο ζεύγος (Wilson, 1996).

Βαθμός κορυφής

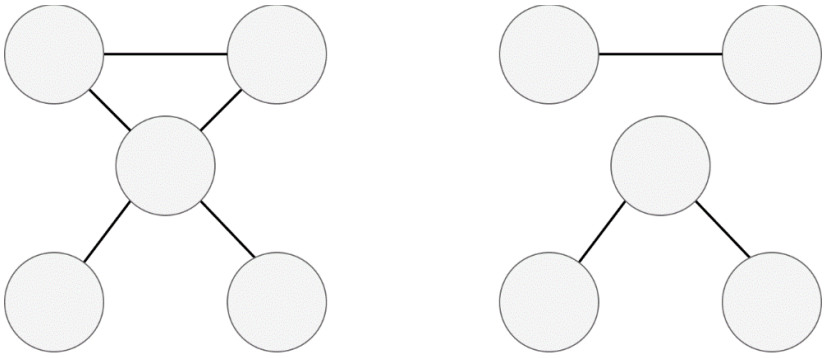
Βαθμός ($\deg_{v_i}$) μιας κορυφης v_i ενός γράφου $G = (V(G), E(G))$ για $v_i \in V(G) = \{v_1, v_2 \dots v_5\}$, είναι ο αριθμός που εκφράζει το πλήθος των ακμών που καταλήγουν σ' αυτήν την κορυφή.

Διαδρομή

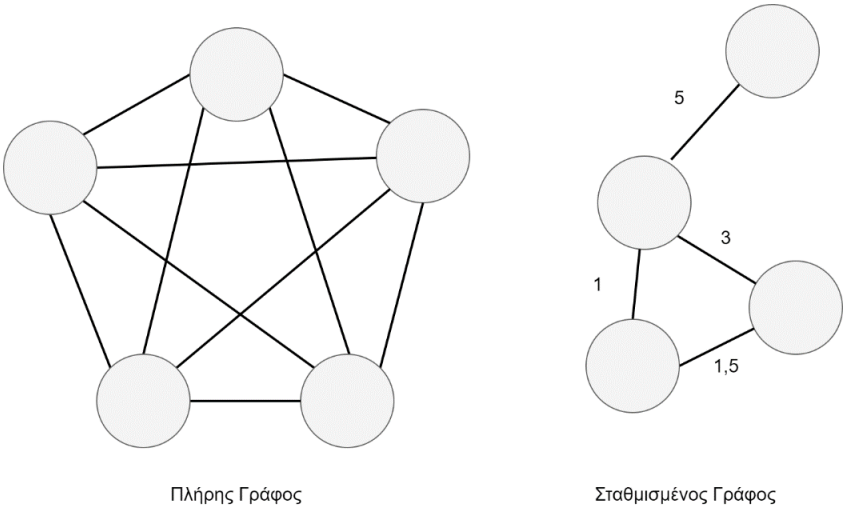
Η διαδρομή σε ένα γράφημα είναι μια πεπερασμένη ή άπειρη ακολουθία ακμών που ενώνει μια ακολουθία κορυφών (James Evans, 2019). Σε μια κατευθυνόμενη διαδρομή όλες οι ακμές κατευθύνονται προς μία κατεύθυνση. Στην ελληνική βιβλιογραφία η διαδρομή αναφέρεται και ως μονοπάτι από την αγγλική λέξη path.

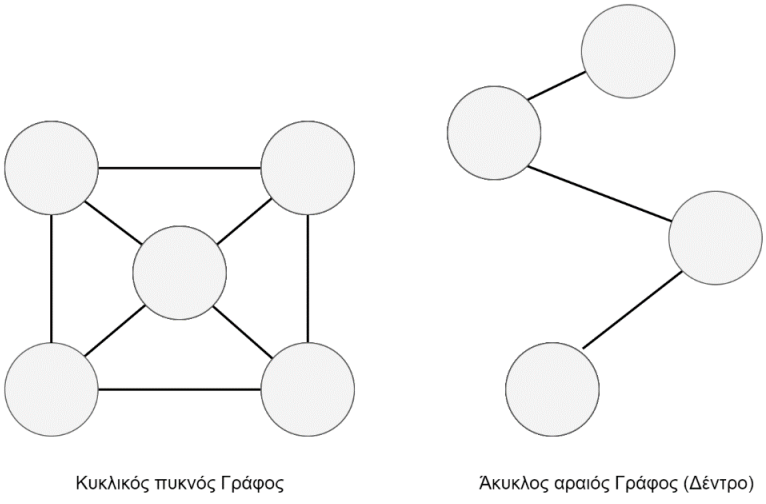
Συνεκτικότητα

Συνεκτικό (connected) ονομάζεται εκείνο το γράφημα $G = (V, E)$, όπου για κάθε ζεύγος κορυφών $u, v \in V$, υπάρχει μονοπάτι που συνδέει του κόμβους u, v (Εικόνα 2).



Εικόνα 2. Συνεκτικός, μη συνεκτικός Γράφος





Εικόνα 3. Διάφοροι τύποι γραφημάτων

Κατηγορίες γραφημάτων (Εικόνα 3)

- Σταθμισμένο γράφημα, είναι ένα ειδικού τύπου γράφημα στο όποιο κάθε ακμή έχει ένα αριθμητικό βάρος,
- Πλήρες γράφημα, ένα πλήρες γράφημα έχει το χαρακτηριστικό ότι κάθε ζεύγος κορυφών έχει μια ακμή που τις συνδέει,
- Ένα γράφημα καλείται αραιό όταν έχει λίγες ακμές, ενώ σε αντίθετη περίπτωση καλείται πυκνό.
- Ένας κύκλος σε ένα γράφημα είναι ένα μη κενό μονοπάτι στο οποίο οι μόνες επαναλαμβανόμενες κορυφές είναι οι πρώτες και οι τελευταίες κορυφές. Ένα γράφημα χωρίς κύκλους ονομάζεται ακυκλικό ή άκυκλο γράφημα. Ένα συνεκτικό γράφημα χωρίς κύκλους ονομάζεται δένδρο.

2.3 ΑΝΙΧΝΕΥΣΗ ΚΟΙΝΟΤΗΤΩΝ

2.3.1 Βιβλιογραφική Ανασκόπηση

Η παρούσα εργασία αντιμετωπίζει το πρόβλημα της ανίχνευσης κοινότητας και αλληλοεπικάλυψης (community detection and overlapping) σε κοινωνικά δίκτυα. Ο εντοπισμός κοινότητας (community detection) έχει πολλές εφαρμογές καθώς οι κοινότητες αποτελούν μια πιο ρεαλιστική προσέγγιση των σύγχρονων δικτύων. Για παράδειγμα οι επιστήμονες μπορούν να ανήκουν σε μία ή περισσότερες κοινότητες με βάση τα ερευνητικά τους ενδιαφέροντα, οι ιατρικές κοινότητες οργανώνονται σε μικρότερες με βάση τις ειδικότητες και οι αθλητές ανήκουν σε διαφορετικές κοινότητες ανάλογα με το άθλημα με το οποίο απασχολούνται. Οι κοινότητες μπορούν είτε να αποσυνδεθούν είτε να επικαλυφθούν (disjoined ή overlapped). Η πλειοψηφία των πιο πρόσφατων μελετών επικεντρώνεται στο πρόβλημα των αλληλεπικαλυπτόμενων κοινοτήτων (overlapped communities).

Η μελέτη των κοινοτικών δομών σχετίζεται γενικά με το πρόβλημα διαμέλισης του δικτύου. Διαμέλιση δικτύου ορίζεται ως ο διαχωρισμός ενός δικτύου σε ένα σύνολο ομάδων περίπου ίσων μεγεθών με ελάχιστο αριθμό ακμών. Ωστόσο, η συγκεκριμένη διαμέλιση των δικτύων δεν είναι η ιδανική μέθοδος για την ανάλυση και μελέτη αυτών καθώς στα πραγματικά δίκτυα, οι κοινότητες που σχηματίζονται πρώτον σπάνια έχουν περίπου το ίδιο μέγεθος και δεύτερον η διαμέλιση των δικτύων δεν λαμβάνει υπόψη τις ομοιότητες μεταξύ των κόμβων (χρηστών) που λαμβάνουν χώρα σε ένα κοινωνικό δίκτυο.

Οι μέθοδοι ανίχνευσης κοινότητας έχουν ως στόχο να ομαδοποιήσουν τους κόμβους του δικτύου με βάση τις σχέσεις που διατηρούν μεταξύ τους προκειμένου να σχηματιστούν συνδεδεμένα υποδίκτυα-υπογράφοι από ολόκληρο το δίκτυο-γράφημα. Κατ’ αυτόν τον τρόπο ο εντοπισμός κοινότητας ανήκει στην κατηγορία προβλημάτων και μελέτης των γραφημάτων.

Η ανίχνευση κοινότητας μπορεί να κατηγοριοποιηθεί σε τρείς βασικές προσεγγίσεις:

1. στην προσέγγιση από πάνω προς τα κάτω, η οποία ξεκινά από το γράφημα που αντιπροσωπεύει ολόκληρο το δίκτυο και προσπαθεί να το χωρίσει σε κοινότητες,

2. την προσέγγιση από κάτω προς τα πάνω που χρησιμοποιεί τοπικές δομές και προσπαθεί να τις επεκτείνει για να σχηματίσει κοινότητες και

3. την προσέγγιση που βασίζεται στις δομές δεδομένων η οποία προϋποθέτει τη μετατροπή ολόκληρου του δικτύου σε δομή δεδομένων (συνήθως σε μορφή δένδρου) η οποία στη συνέχεια υποβάλλεται σε επεξεργασία για τον εντοπισμό κοινότητας.

2.3.1.1 Προσέγγιση από πάνω προς τα κάτω

Η προσέγγιση από πάνω προς τα κάτω βασίζεται στην ιδέα του διαμοιρασμού των κόμβων ενός δικτύου σε μικρότερες ομάδες προκειμένου να αναγνωρίζονται ως κοινότητες. Όταν οι σύνδεσμοι που συνδέονται με έναν κόμβο βρίσκονται σε περισσότερες από μία ομάδα αυτός ο κόμβος θεωρείται ότι επικαλύπτεται. Ο Prat-Perez (Prat-Pérez, 2012) προτείνει τη σταθμισμένη κοινοτική ομαδοποίηση $Weighted\ Community\ Clustering\ (WCC)$, η οποία υπολογίζει το επίπεδο συνοχής ενός συνόλου κόμβων S. Η ιδέα πίσω από αυτό το έργο είναι ότι οι ισχυρές κοινότητες είναι εκείνες με σημαντικό αριθμό τριγώνων τα οποία κατανέμονται ομοιόμορφα σε όλους τους κόμβους. Επομένως, για να οριστεί μια κοινότητα, το WCC μετρά την αναλογία των τριγώνων που σχηματίζει ένας κόμβος x με τους κόμβους ενός συνόλου S, σε σχέση με τον αριθμό των τριγώνων που σχηματίζει ο x σε ολόκληρο το γράφημα. Επίσης, υπολογίζει τον αριθμό των κόμβων που σχηματίζουν τουλάχιστον ένα τρίγωνο με τον x, σε σχέση με την ένωση αυτού του συνόλου και του συνόλου S. Μια κοινότητα που περιλαμβάνει τον x ορίζεται ως ισχυρή όταν το σύνολο S περιλαμβάνει το μεγαλύτερο δυνατό αριθμό κορυφών που σχηματίζουν τρίγωνα με τον x και το μικρότερο δυνατό αριθμό κόμβων έτσι ώστε ο x να μην σχηματίζει τρίγωνα με το υπόλοιπο δίκτυο. Ιδανικά ο x θεωρείται ισχυρό μέλος της κοινότητας που ορίζεται από το σύνολο S, όταν σχηματίζει τρίγωνα με όλους τους κόμβους του συνόλου S (σε ζεύγη) και όταν ο αριθμός των κόμβων του υπόλοιπου δικτύου που σχηματίζουν τρίγωνα με τον x είναι 0. Τα πειραματικά αποτελέσματα έχουν δείξει ότι η WCC παράγει πράγματι υψηλής πυκνότητας καλά καθορισμένες κοινότητες. Μια παρόμοια προσέγγιση που βασίζεται σε τρίγωνα (Tung, 2012)

Οι Chen et al. (D. Chen, 2010) υπολογίζουν πρώτα την ισχύ κάθε κόμβου και στη συνέχεια εντοπίζουν μια αρχική κοινότητα από τον κόμβο με τη μεγαλύτερη ισχύ (το άθροισμα όλων των βαρών που συνδέονται με αυτόν τον κόμβο). Έπειτα επιλέγεται ο ισχυρότερος κόμβος και ο βαθμός του (ένα μέτρο που βασίζεται στους συντελεστές των συνδέσμων) και συγκρίνεται με ένα όριο. Ο κόμβος ανήκει σε μια κοινότητα εάν ο βαθμός του είναι μικρότερος από αυτό το όριο. Προφανώς, οι σύνδεσμοι του κόμβου μπορεί να είναι συνδεδεμένοι με ορισμένες κοινότητες, στις οποίες μπορεί να ανήκει ο ίδιος ο κόμβος, ανάλογα με τον βαθμό και την τιμή ορίου του.

Μια παρόμοια προσέγγιση βρίσκεται στο (Z. Lu, 2015), όπου οι συγγραφείς παρουσιάζουν μια στρατηγική που μπορεί να εντοπίσει αλληλεπικαλυπτόμενες και μη αλληλεπικαλυπτόμενες κοινότητες και προσθέτει έναν κόμβο σε μια κοινότητα σε κάθε αναπτυσσόμενο βήμα. Συγκεκριμένα, κάθε επεκτεινόμενο βήμα υπολογίζει το βαθμό των κόμβων μιας κοινότητας C και στη συνέχεια, επιλέγεται αυτός με τον υψηλότερο βαθμό. Αυτός ο κόμβος συνδέεται προσωρινά στο C, σχηματίζοντας C'. Στη συνέχεια, εάν η αγωγιμότητα του C' είναι χαμηλότερη σε σύγκριση με την αγωγιμότητα του C, ο επιλεγμένος κόμβος συνδέεται τελικά με την κοινότητα C.

Μια πρόσφατα εισαγόμενη στρατηγική από πάνω προς τα κάτω που ονομάζεται picaso εισήχθη πρόσφατα από τους Qiao et al. (Qiao et al, 2018) Το προτεινόμενο σχήμα εντοπίζει κοινότητες που

χρησιμοποιούν ένα αρθρωτής μορφής μοντέλο, $mountain\ model$, το οποίο χωρίζει το δίκτυο σε ομάδες αλυσίδων (από πάνω προς τα κάτω) και τις ταξινομεί ανάλογα με τα βάρη των ακμών. Με βάση τα χαρακτηριστικά της κοινότητας, ορισμένες άκρες «χαμηλώνουν» και άλλες «υψώνονται» (εξ ου και η ορολογία mountain). Νέες κοινότητες σχηματίζονται από τα «βουνά» που παράγονται. Περιλαμβάνεται επίσης μια φάση ενημέρωσης-λειτουργικότητας.

Γενικά, οι προσεγγίσεις από πάνω προς τα κάτω είναι μια ενδιαφέρουσα και καλά υιοθετημένη ιδέα για την πραγματοποίηση εντοπισμού κοινότητας. Το πλεονέκτημά τους είναι ότι μπορούν εύκολα να εντοπίσουν αλληλεπικαλυπτόμενες κοινότητες, αλλά μερικές φορές αυτή η αλληλεπικάλυψη είναι πολύ υψηλή, εάν ένας κόμβος συνδέεται με μεγάλο αριθμό συνδέσμων που διανέμονται σε πολλές διαφορετικές κοινότητες. Σε ένα τέτοιο σενάριο, ενδέχεται να προκύψουν καθυστερήσεις επεξεργασίας, οπότε οι αλγόριθμοι θα πρέπει να είναι προσεκτικοί όσον αφορά τις συνδέσεις που θεωρούν.

2.3.1.2 Προσέγγιση από κάτω προς τα πάνω

Το δεύτερο είδος προσεγγίσεων ξεκινά από τοπικές δομές και επεκτείνεται στο συνολικό δίκτυο. Κατά τη διάρκεια αυτής της διαδικασίας, διαμορφώνονται διάφορες κοινότητες. Ένας αριθμός διαφορετικών ιδεών χρησιμοποιείται για την εφαρμογή μιας προσέγγισης εντοπισμού κοινότητας από κάτω προς τα πάνω.

Η $βελτιστοποίηση$ είναι μια προσέγγιση από κάτω προς τα πάνω που χαρακτηρίζει την ποιότητα ενός διασυνδεδεμένου τμήματος του δικτύου. Η κοινότητα θεωρείται ως υπογράφος που προσδιορίζεται από τη μεγιστοποίηση της φυσικής κατάστασης των κόμβων. Αυτό το μέτρο βασίζεται στους συνολικούς εσωτερικούς και εξωτερικούς βαθμούς των κόμβων μιας ομάδας (ή λειτουργικής μονάδας). Ο στόχος αυτού του προβλήματος βελτιστοποίησης είναι να βρεθεί ένας υπογράφος που ξεκινά από έναν καθορισμένο κόμβο έτσι ώστε, η συμπερίληψη ενός νέου κόμβου ή η εξάλειψη ενός κόμβου από τον υπογράφο να μειώσει την τιμή φυσικής κατάστασης. Οι Nascimento και Pitsoulis (Pitsoulis, 2013) παρουσίασαν μια επιθετική διαδικασία τυχαίας προσαρμοστικής αναζήτησης, Greedy Randomized Adaptive Search Process (GRASP) με επανασύνδεση διαδρομής, για την επίλυση του προβλήματος μεγιστοποίησης της αρθρωτότητας (clustering) σε σταθμισμένα γραφήματα. Χρησιμοποιήθηκε μια κλάση {0, 1} για τον χαρακτηρισμό της οικογένειας συμπλεγμάτων στο δίκτυο. Η κατασκευή συμπλέγματος είναι το πρώτο στάδιο αυτού του αλγορίθμου και παρέχει μια καλή αρχική λύση για μια τοπική αναζήτηση. Στο δεύτερο στάδιο, η εξαντλητική αναζήτηση πραγματοποιείται στη γειτονιά μιας δεδομένης λύσης, προκειμένου να επιτευχθεί ένα τοπικό βέλτιστο. Τέλος, χρησιμοποιείται επανασύνδεση, και πραγματοποιείται αναζήτηση για να εξερευνηθεί ο χώρος των λύσεων που ορίζεται από δύο καλής ποιότητας λύσεις, για να βρεθεί μια καλύτερη λύση. Ο Džamić et al. (D. Džamić, 2019) πρότεινε μια μέθοδο που ονομάζεται Ascent-Descent VNDS, η οποία συνδυάζει την τοπική αναζήτηση με συστηματικές αλλαγές δομών γειτονιάς για να ξεφύγει από τις τοπικές βέλτιστες παγίδες. Αυτή η μέθοδος χρησιμοποιείται για την ανίχνευση κοινοτήτων με μεγιστοποίηση αρθρωτότητας.

Μια άλλη ιδέα για την εφαρμογή μιας προσέγγισης από κάτω προς τα πάνω είναι το Clique Percolation. Αυτή η μέθοδος προϋποθέτει ότι μια κοινότητα αποτελείται από πλήρως συνδεδεμένους υπογράφους. Σύνολα τέτοιων υπογράφων ενδέχεται να αλληλεπικαλύπτονται. Η ανίχνευση της κοινότητας βασίζεται στην αναζήτηση και τον εντοπισμό γειτονικών ομάδων. Αρχικά, βρίσκει όλες τις κλίκες στο δίκτυο, οι οποίες στη συνέχεια απεικονίζονται στο γράφημα από μια κορυφή. Εάν δύο κλίκες μοιράζονται έναν προκαθορισμένο αριθμό μελών, τότε συνδέονται οι αντίστοιχες κορυφές τους. Έτσι, οι συνδεδεμένες

κορυφές στο γράφημα αντιπροσωπεύουν κοινότητες δικτύου. Μια ενδιαφέρουσα τεχνική διήθησης κλίκας (clique) εισήχθη από τους Farkas et al. (I. Farkas, 2007). Σε αυτόν τον αλγόριθμο, εισήχθη ένα προκαθορισμένο όριο έντασης και μόνον οι κλίκες με ένταση υψηλότερη από το κατώτατο όριο αποτελούν μια κοινότητα. Στο (J. M. Kumpula, 2008), η διαδικασία ανίχνευσης της κοινότητας χωρίζεται σε φάσεις: Στην πρώτη φάση, οι κλίκες μελών $k-1$ ανιχνεύονται ελέγχοντας όλες τις κλίκες μελών $k-2$, που βρίσκονται στη γειτονιά δύο κορυφών. Στη συνέχεια, ελέγχονται τα συνδεδεμένα στοιχεία των $k-1$, για να σχηματιστεί μια κοινότητα. Σε μια πιο πρόσφατη προσέγγιση, ο Zhang et al. (X. Zhang, 2017) αντιμετωπίζει το πρόβλημα της διαστατικότητας από το οποίο υποφέρουν οι τεχνικές διήθησης κλίκας. Πρότεινε το ευρετήριο Salton, για να χαρακτηριστούν ομοιότητες κόμβων και οι αδύναμες κλίκες που εντοπίστηκαν να συγχωνευτούν σε μεγαλύτερες κοινότητες, όποτε ήταν δυνατόν.

Μια νέα ιδέα εισήχθη από τους Xiao et al. (W. Zhi-Xiao, 2016), που βασίζεται στην εγγύτητα των κόμβων μέσα σε μια δομή δικτύου. Αυτή η ανάλυση θέσης κόμβου βασίζεται στην εκτίμηση της μάζας των κόμβων και εντοπίζει τη θέση τους στο δίκτυο. Η μέθοδος Parallel Louvain with Refinement (PLMR) βασίζεται στην αρχική μέθοδο Parallel Louvain (PLM) που εισήχθη από τους Blondel et al. (V. D. Blondel, 2008). Το PLM είναι μια προσέγγιση από κάτω προς τα πάνω, η οποία χρησιμοποιεί τη λειτουργικότητα (modularity) ως αντικειμενική συνάρτηση. Σε κάθε βήμα, οι κόμβοι μεταφέρονται σε γειτονικές κοινότητες με τέτοιο τρόπο ώστε η αρθρωτότητα (clustering) να μεγιστοποιείται τοπικά. Η διαδικασία σταματά όταν όλες οι κοινότητες βρίσκονται σταθερές. Η βελτίωση προστέθηκε από τους Staudt και Meyerhenke (Meyerhenke, 2016) και είναι μια επιπλέον φάση μετακίνησης μετά από κάθε παράταση. Αυτή η κίνηση είναι απαραίτητη για την επανεκτίμηση των αναθέσεων κόμβων, προκειμένου να προσαρμοστεί στις αλλαγές που έχουν πραγματοποιηθεί.

Τέλος, μια άλλη κοινή ιδέα για την εφαρμογή μιας προσέγγισης από κάτω προς τα πάνω είναι η διάδοση ετικετών (label propagation). Η διάδοση ετικετών είναι μια τεχνική που εκχωρεί ετικέτες σε σημεία δεδομένων που δεν είχαν επισημανθεί προηγουμένως. Αρχικά, μερικοί κόμβοι έχουν μόνο ετικέτες και καθώς προχωρά ο αλγόριθμος, οι κόμβοι υιοθετούν τις ετικέτες που έχουν οι περισσότεροι από τους γείτονές τους μέχρι στιγμής (U. N. Raghavan, 2007).

Ο Gregory [25] πρότεινε επέκταση της τεχνικής διάδοσης ετικετών των Raghavan et al. (U. N. Raghavan, 2007) ονομάστηκε Community Overlap Propagation Algorithm (COPRA), προκειμένου να είναι σε θέση να εντοπίσει αλληλεπικαλυπτόμενες κοινότητες. Συγκεκριμένα, τα βήματα ετικέτας και διάδοσης επεκτείνονται ώστε να περιλαμβάνουν πληροφορίες για περισσότερες από μία κοινότητες, έτσι κάθε κορυφή μπορεί τώρα να ανήκει σε περισσότερες κοινότητες. Οι κορυφές δικτύου που αντιπροσωπεύουν κλίκες έχουν ετικέτες που διαδίδονται μεταξύ γειτονικών κορυφών, έτσι ώστε τα μέλη μιας κοινότητας να κάνουν εμφανή τη συμμετοχή τους στην κοινότητα.

Γενικά, οι προσεγγίσεις από κάτω προς τα πάνω έχουν το πλεονέκτημα ότι, σε πολλές περιπτώσεις, η πολυπλοκότητά τους είναι γραμμική (για βελτιστοποίηση και τεχνικές διάδοσης ετικετών). Ωστόσο, οι στρατηγικές που ανήκουν σε αυτές τις υποκατηγορίες συχνά αποτυγχάνουν να εντοπίσουν πολύ μικρές κοινότητες, ακόμη και σε περιπτώσεις που είναι καλά καθορισμένες. Αυτό συμβαίνει γενικά επειδή οι αρχικές τοπικές δομές δεν συλλαμβάνουν εξ αρχής αυτές τις μικρές κοινότητες και η μέθοδος επέκτασης που χρησιμοποιείται δεν ενσωματώνει κόμβους-μέλη μιας κοινότητας. Οι στρατηγικές διείσδυσης κλίκας (clique) υπόκεινται σε υψηλό υπολογιστικό κόστος, καθώς πρέπει να ελέγχουν συνεχώς κάθε ζεύγος κλίκας που εντοπίζεται, για να προσδιορίσουν εάν μπορούν να ανήκουν σε μια μεγαλύτερη κλίκα.

2.3.1.3 Προσέγγιση δομών δεδομένων

Η προσέγγιση που βασίζεται στις δομές δεδομένων βασίζεται στην ιδέα του σχηματισμού ενός δικτύου για κάποιο τύπο δομής δεδομένων (συνήθως σε μορφή δένδρου), το οποίο στη συνέχεια αναλύεται με τρόπο εντοπισμού κοινοτήτων. Εδώ, περιγράφονται εν συντομία μερικά από τα πιο αντιπροσωπευτικά παραδείγματα στρατηγικών αυτού του τύπου.

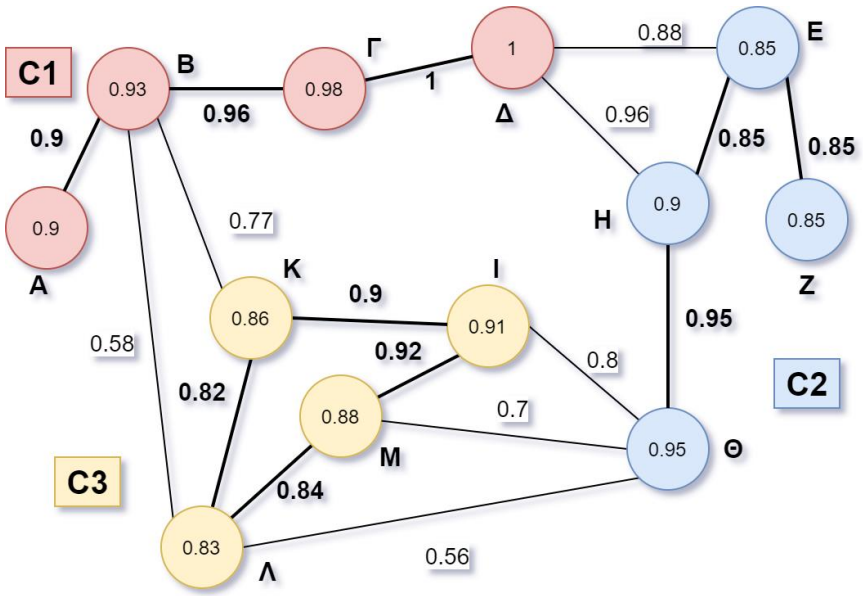
Ο Ahn et al. [26] χρησιμοποίησε το μετρητή του ευρετηρίου Jaccard για να υπολογίσει την ομοιότητα για κάθε δεδομένο ζεύγος συνδέσμων που συνδέονται με έναν κόμβο. Με βάση τις ομοιότητες, χτίζεται ένα δενδροδιάγραμα συνδέσμων, το οποίο στη συνέχεια κόβεται σε κάποιο σημείο για να παραχθούν οι κοινότητες. Ο Overlapping Community Algorithm (OCA) (Padrol-Sureda, 2010) βασίζεται στην ιδέα της χαρτογράφησης κάθε κόμβου σε ένα πολυδιάστατο διάνυσμα. Στη συνέχεια, κάθε υποσύνολο κόμβου ορίζεται ως το άθροισμα των μεμονωμένων διανυσμάτων σε αυτό το σύνολο.

Οι αλγόριθμοι συσσωμάτωσης δημιουργούν ιεραρχίες δένδρων ξεκινώντας από μικρές συστάδες και επεκτείνονται σε μεγαλύτερες. Ο Clauset et al. (A. Clauset, 2004) παρουσίασε έναν αλγόριθμο που ξεκινά από μεμονωμένους κόμβους και δημιουργεί τη δομή δενδρογράμματος που περιγράφει τη δομή της κοινότητας, διατηρώντας παράλληλα τις αλλαγές στην αρθρωτότητα (modularity). Μια ελαφρώς διαφορετική προσέγγιση, το $matrix\ blocking$ (Saad, 2012) δημιουργεί ένα δένδρο ιεραρχίας όπου κάθε κόμβος του δικτύου αποτελεί μια στήλη στον πίνακα (matrix). Στη συνέχεια, η μερική ομαδοποίηση υπολογίζεται στο γράφημα, όπου κάθε κόμβος δεν θεωρείται απαραίτητα μέλος κοινότητας. Τέλος, μια άλλη ενδιαφέρουσα προσέγγιση δομής δεδομένων παρουσιάζεται στο (J. Huang, 2013). Το σχήμα ονομάζεται skeleton-based clustering (gSkeletonClu) και η ιδέα του είναι να προβάλλει ένα μη κατευθυνόμενο δίκτυο στο μέγιστο δένδρο που εκτείνεται. Στη συνέχεια, εντοπίζονται οι βελτιστοποιημένες συστάδες στο δένδρο.

Η παρούσα εργασία ανήκει στην κατηγορία της προσέγγισης μέσω δομών δεδομένων. Γενικά, η ιδέα της μετατροπής ενός δικτύου σε μια δομή δένδρου είναι αρκετά ενδιαφέρουσα, παρόλα αυτά θα πρέπει να εφαρμοστεί με προσοχή καθώς μπορεί να είναι πολύ δαπανηρή από θέμα υπολογισμού, ειδικά κατά την επεξεργασία δικτύων με μεγάλο αριθμό κόμβων και ακμών. Ο προσεκτικός παραλληλισμός των απαραίτητων διεργασιών είναι η λύση γι' αυτό το πρόβλημα κάτι το οποίο αποτελεί αντικείμενο μελλοντικής έρευνας.

2.3.2 Κριτήρια Συμμετοχής σε Κοινότητα

Έστω ένα μικρό δίκτυο ακανόνιστης τοπολογίας με τρεις αρχικοποιημένες κοινότητες $C1, C2, C3$ όπως φαίνεται στην Εικόνα 4. Επίσης, υποθέτουμε ότι οι σύνδεσμοι μεταξύ των κόμβων δείχνουν ότι υπάρχει συσχέτιση μεταξύ τους, με την έννοια ότι έχουν παρόμοιες απόψεις σε θέματα ή κοινά ενδιαφέροντα. Πραγματικά σύνολα δεδομένων μπορούν να αντληθούν από την αναφορά (Sifaleras, 2018).



Εικόνα 4. Παράδειγμα αναπαράστασης κοινωνικού δικτύου ως Γράφος

Η Εικόνα 4 αποκαλύπτει ορισμένες πτυχές οι οποίες πρέπει να αναφερθούν:

1. Ο χρήστης Β σχετίζεται με τον Κ που ανήκει στο C3. Έτσι, υπάρχει πιθανότητα ο Β να ανήκει επίσης στο C3.

2. Ο χρήστης Β συνδέεται έμμεσα με την κοινότητα C2 μέσω του κόμβου Λ ή μέσω του Γ, Δ. Έτσι, υπάρχει η πιθανότητα ο Β να ανήκει επίσης στην κοινότητα C2. Σ' αυτήν την περίπτωση, οι κόμβοι Λ ή Γ, Δ μπορεί επίσης να ανήκουν στην κοινότητα C3.

3. Συνεχίζοντας από το 2, εάν υπάρχει υψηλό ποσοστό αλληλοεπικάλυψης μεταξύ δύο κοινοτήτων, υπάρχει πιθανότητα οι δύο κοινότητες να είναι μία.

4. Δεν είναι απαραίτητο όλοι οι χρήστες ενός δικτύου να είναι συσχετισμένοι (π.χ. δυο χρήστες του Facebook δεν έχουν άμεση σχέση αλλά μπορεί να έχουν κοινούς φίλους). Στο Εικόνα 4, ο Β δεν είναι άμεσα συνδεδεμένος με τον Θ, αλλά έχει άμεση συσχέτιση με τον Λ. Με τη σειρά του ο Λ είναι συνδεδεμένος με τον Θ. Με αυτόν τον τρόπο (συνεχίζοντας με το παράδειγμα του Facebook) ο Β μπορεί να μην γνωρίζει την ύπαρξη της κοινότητας C2 έως ότου επισκεφθεί τη σελίδα κάποιου χρήστη που έχει σχέση με αυτήν την κοινότητα (όπως ο Λ). Με άλλα λόγια, η διαδρομή που σχετίζεται με τους Β και Θ μπορεί να είναι αρκετά ισχυρή ώστε οι δύο κόμβοι να είναι μέλη της ίδιας κοινότητας.

Έχουν προταθεί διάφορα κριτήρια για τον εντοπισμό της συμμετοχής στην κοινότητα. Πριν εισαγάγουμε τα κριτήρια που χρησιμοποιούνται στη παρούσα εργασία, παρουσιάζονται εν συντομία μερικά από τα αντιπροσωπευτικά κριτήρια που βρίσκονται στα άρθρα που παρουσιάζονται στην ενότητα Βιβλιογραφικής ανασκόπησης, στις διάφορες προσεγγίσεις ανίχνευσης κοινότητας. Περισσότερα κριτήρια μπορούν να βρεθούν στα αναφερόμενα άρθρα.

- Ανάλυση θέσης κόμβου, *Node Locacion Analysis* (W. Zhi-Xiao, 2016): Η μάζα κόμβων αξιολογείται και η σχέση μεταξύ κόμβων καθορίζεται με βάση τη θέση τους στη δομή.

- Λειτουργία αγωγιμότητας, $Conductance\ Function$ (Z. Lu, 2015): Τα βάρη των ακμών ενός δικτύου διαιρούμενο με το βάρος όλων των ακμών του δικτύου. Όταν η αγωγιμότητα είναι χαμηλή, περισσότεροι κόμβοι βρίσκονται σε μια κοινότητα, η οποία θεωρείται ισχυρότερη

- $Salton\ index$ (X. Zhang, 2017): Βασίζεται στην ενότητα των γειτονικών συνόλων δύο κόμβων. Όσο μεγαλύτερη είναι η μεταβλητή, τόσο περισσότερους κοινούς γείτονες έχουν οι δύο κόμβοι, οπότε είναι πιο πιθανό ότι είναι μέλη της ίδιας κοινότητας. Το ευρετήριο Jaccard και Sørensen αναφέρεται επίσης σε αυτήν την εργασία, οι κόμβοι αναφέρονται ως πιθανοί υποψήφιοι.

- Συμμετοχικός συντελεστής $Belonging\ Coefficient$ (Gregory, 2010): Δείχνει τη δύναμη της συμμετοχής ενός κόμβου σε μια κοινότητα. Κάθε βήμα διάδοσης ορίζει την ετικέτα κόμβων στην ένωση των ετικετών των γειτόνων της, αθροίζει τους συντελεστές που ανήκουν στις κοινότητες έναντι όλων των γειτόνων και ομαλοποιεί το αποτέλεσμα.

- $Modularity$ (Newman, 2006): Ο αριθμός των ακμών που εμπίπτουν σε ομάδες μείον τον αναμενόμενο αριθμό σε ένα ισοδύναμο δίκτυο με ακμές τοποθετημένες τυχαία.

Το Κεφάλαιο 3 παρουσιάζει λεπτομερώς τη μαθηματική μοντελοποίηση, τα κριτήρια και στη συνέχεια την υλοποίηση του σχήματος με το οποίο ασχολείται η παρούσα διπλωματική εργασία.

ΚΕΦΑΛΑΙΟ 3 ΔΙΑΤΥΠΩΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ & ΑΛΓΟΡΙΘΜΙΚΗ ΕΠΙΛΥΣΗ

3.1 ΟΡΙΣΜΟΣ ΚΑΙ ΜΑΘΗΜΑΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ

3.1.1 Βασικά Χαρακτηριστικά

Έστω $G = (V, E)$ ένα σταθμισμένο μη κατευθυνόμενο γράφημα, όπου V, E είναι τα σύνολα των κόμβων και των ακμών αντίστοιχα. Οι κόμβοι αντιπροσωπεύουν τους χρήστες και οι ακμές τη σχέση μεταξύ δύο χρηστών (π.χ. φιλία στο Facebook). Η ομοιότητα $w_{i,j}$ μεταξύ δυο χρηστών i, j είναι το βάρος της ακμής που συνδέει αυτούς τους κόμβους. Αυτή η τιμή βρίσκεται στο διάστημα [0…1]. Όπως περιγράφεται στο Κεφάλαιο 5 - Υπολογιστική Εμπειρία, αυτή η τιμή υπολογίζεται (Εξίσωση (4.2)) με βάση τα δεδομένα που συλλέγονται για τα διάφορα δίκτυα από την αναφορά (Krevl, 2014). Για παράδειγμα η τιμή του βάρους 0,78 δείχνει ότι οι απόψεις δυο χρηστών μπορούν να θεωρηθούν συγκλίνουσες σε ποσοστό 78%.

Επιπρόσθετα, οι κόμβοι του δικτύου σταθμίζονται και αυτοί: το βάρος ενός κόμβου υποδηλώνει το βαθμό συνδεσιμότητας δικτύου, $Network\ Connectivity\ Degree\ (NCD)$, δηλαδή το πόσο καλά οι προτιμήσεις και οι προβολές ενός χρήστη προσαρμόζονται σε μια κοινότητα. Ο βαθμός συνδεσιμότητας δικτύου κυμαίνεται επίσης μεταξύ [0…1] (Stavros Souravlas, 2019). Στην αναπαράσταση του δικτύου της Εικόνας 4 τα γράμματα είναι τα ονόματα των κόμβων, οι τιμές των ακμών είναι οι ομοιότητες (βάρη) και οι τιμές των κόμβων υποδεικνύουν τους βαθμούς συνδεσιμότητας δικτύου των χρηστών. Ο βαθμός συνδεσιμότητας δικτύου ενός χρήστη i, NCD_i υπολογίζεται ως εξής:

Εξίσωση 3.1 - Βαθμός συνδεσιμότητας δικτύου, Network Connectivity Degree (NCD)

$$NCD_i = \frac{\sum w_{i,j}}{n_e} \tag{3.1}$$

Όπου $w_{i,j}$ είναι το βάρος οποιασδήποτε ακμής που συνδέει τον χρήστη i με οποιοδήποτε χρήστη j που βρίσκεται στην ίδια κοινότητα και n_e είναι ο αριθμός τέτοιων ακμών. Για παράδειγμα, στην κοινότητα $C1$, ο χρήστης Β έχει δύο εσωτερικούς συνδέσμους με τους χρήστες Α, Γ και δύο εξωτερικούς συνδέσμους με τους χρήστες Λ και Κ. Για τον υπολογισμό του NCD_B, λαμβάνουμε υπόψη μόνο τους εσωτερικούς συνδέσμους (ακμές) της κοινότητας $C1$ και έχουμε $NCD_B = \frac{(0{,}9+0{,}96)}{2} = 0{,}93$.

Επομένως, η μέση συνδεσιμότητα κοινότητας $Average\ Community\ Connectivity\ (ACC)$ για μια κοινότητα C ορίζεται ως ο μέσος όρος των NCD όλων των κόμβων S στην κοινότητα C, δηλαδή:

Εξίσωση 3.2 - Μέση συνδεσιμότητα κοινότητας, Average Community Connectivity (ACC)

$$ACC_C = \frac{\sum_{i=1}^{n} NCD_i}{N} \tag{3.2}$$

Αυτό το μέτρο δείχνει πόσο έντονα σχετίζονται τα μέλη μιας κοινότητας μεταξύ τους.

3.1.2 Κριτήρια Συμμέτοχης σε Κοινότητα

Θα χρησιμοποιηθεί ένα διπλό κριτήριο για να προσδιορίσουμε την ιδιότητα μέλους ενός χρήστη για μία κοινότητα.

1ο Κριτήριο

Ένα χρήστης μπορεί να θεωρηθεί μέλος μιας κοινότητας είτε άμεσα, μέσω μια σχέσης με έναν η περισσότερους χρήστες της κοινότητας, είτε έμμεσα, μέσω μιας σχέσης με κάποιον χρήστη που σχετίζεται με ένα μέλος της κοινότητας. Για να προσδιορίσουμε τη σχέση ενός χρήστη με μια κοινότητα, πρέπει να εντοπίσουμε την ύπαρξη ισχυρότερης διαδρομής (stronger path), από πλευράς βαρών σε σχέση με τη διαδρομή που ήδη καθορίζει τη συμμετοχή του σε μια κοινότητα. Παρουσιάζεται λοιπόν η έννοια της ισχύος διαδρομής $Path\ Strength\ (PS)$ ως το άθροισμα των βαρών μιας διαδρομής που συνδέει δύο μη συνδεδεμένους (άμεσα) χρήστες i, j διαιρεμένο με το πλήθος των αναπηδήσεων σ' αυτήν τη διαδρομή.

Εξίσωση 3.3 - Ισχύς διαδρομής, Path strength (PS)

$$PS_{i,j} = \frac{\sum_{i=1}^{k} w_{i,r_{i-1}} + w_{r_i,j}}{k} \tag{3.3}$$

Όπου r_i ένας από τους κόμβους k που βρίσκεται ανάμεσα από τους κόμβους i, j. Έτσι όπως φαίνεται, ένα κριτήριο για τον προσδιορισμό της συμμετοχής ενός χρήστη U θα ήταν η ανίχνευση μιας διαδρομής που ξεκινά τον U, η οποία είναι ισχυρότερη από τον ACC αυτής της κοινότητας.

2ο Κριτήριο

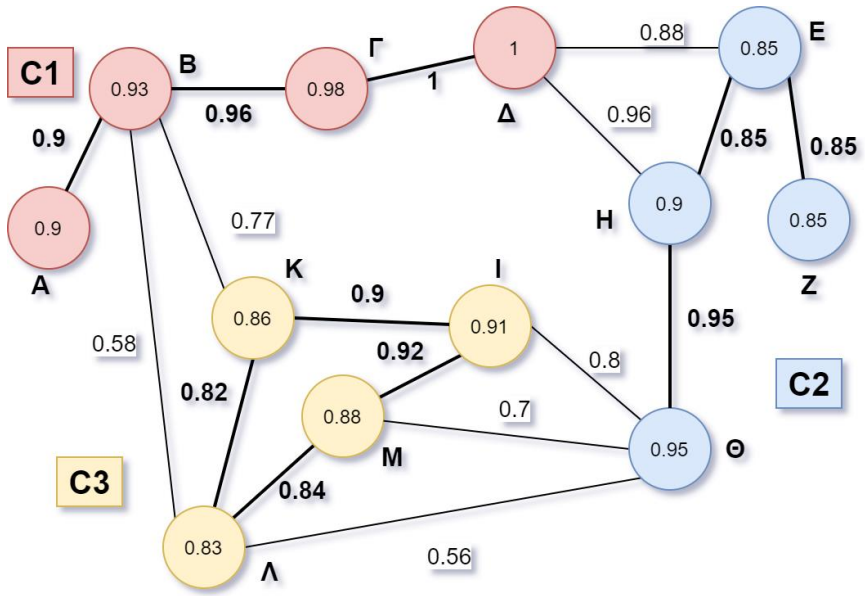
Το μήκος των διαδρομών που βρέθηκαν να ικανοποιούν το πρώτο κριτήριο δεν πρέπει να υπερβαίνει τη διάμετρο d της κοινότητας C, δηλαδή τη μεγαλύτερη διαδρομή που μπορεί να βρεθεί μεταξύ των κόμβων της κοινότητας C. Αυτή η συνθήκη είναι απαραίτητη γιατί, δεδομένης μιας πολύ μεγάλης διαδρομής η ισχύς της διαδρομής μειώνεται. Επιπλέον, οι τιμές ACC είναι αρκετά μεγάλες επομένως η πρώτη συνθήκη μπορεί να μην ικανοποιηθεί ποτέ. Η διάμετρος της κοινότητας C δίνει ένα λογικό όριο για τα αποδεκτά μήκη διαδρομής. Για να συνοψίσουμε, οι συνθήκες που περιγράφονται παραπάνω εκφράζονται από τις ακόλουθες ανισότητες:

Ανίσωση 3.4 - Κριτήριο ισχύος διαδρομής

Ανίσωση 3.5 - Κριτήριο διαμέτρου

$$PS_{i,j} \geq ACC_c \tag{3.4}$$

$$\delta_{i,j} \leq d_c \tag{3.5}$$



Εικόνα 5. Επανάληψη εικόνας 4

Συνοψίζοντας, δίνεται ένα σύντομο παράδειγμα για το διπλό κριτήριο. Η μέση συνδεσιμότητα κοινότητας (ACC_{C1}) της Εικόνας 5 είναι το άθροισμα των NCD για τους κόμβους Α, Β, Γ, Δ διαιρεμένο με 4, $(0{,}9 + 0{,}93 + 0{,}98 + 1)/4 \approx 0{,}95$. Η ισχύς διαδρομής που συνδέει τους κόμβους Μ, Δ μέσω Ι, Θ, Η είναι $(0{,}92 + 0{,}8 + 0{,}95 + 0{,}96)/4 = 0{,}9$. Σ' αυτήν την περίπτωση, $PS_{M\Delta} < ACC_{C1}$ έτσι το κριτήριο της Εξίσωσης (3.4) δεν ικανοποιείται, καθώς επίσης ούτε το κριτήριο της Εξίσωσης (3.5) ($\delta_{M\Delta} = 4 > d_{C1} = 3$). Έτσι ο χρήστης Μ δεν μπορεί να θεωρηθεί μέλος της κοινότητας C1 εξετάζοντας τη διαδρομή $M \rightarrow I \rightarrow \Theta \rightarrow H \rightarrow \Delta$. Αντίθετα, εξετάζοντας εάν ο χρήστης Θ ενδέχεται να ανήκει στην κοινότητα $C1$ διασχίζοντας τον κόμβο Η, έχουμε $PS_{\Theta\Delta} = (0{,}95 + 0{,}96)/2 = 0{,}955 > ACC_{C1}$. Ακόμη $\delta_{\Theta\Delta} = 2 < 3$, άρα τα κριτήρια των Εξισώσεων (3.4), (3.5) ικανοποιούνται και ο χρήστης Θ μπορεί να θεωρηθεί μέλος της κοινότητας C1 εξετάζοντας τη διαδρομή από κόμβο Θ στον Δ μέσω του.

3.2 ΑΛΓΟΡΙΘΜΟΣ ΑΝΙΧΝΕΥΣΗΣ ΚΟΙΝΟΤΗΤΩΝ

Η στρατηγική ανίχνευσης κοινότητας περιλαμβάνει δυο φάσεις:

Φάση Α: η δημιουργία (παραγωγή) δυαδικών δέντρων με νήμα (threated binary trees) και

Φάση Β: η ανάλυση των διαδρομών (tree traversals).

Σε αυτό το κεφάλαιο παρουσιάζεται λεπτομερώς η ανάλυση των υπολογισμών που απαιτούνται για την εκπλήρωση της αλγοριθμικής διαδικασίας του μοντέλου της παρούσας εργασίας.

3.2.1 Παραγωγή Δυαδικών Δέντρων με Νήματα (threated binary tree generation)

Σε αυτή την φάση του αλγορίθμου δημιουργείται ένα δυαδικό δένδρο με νήματα για κάθε έναν από τους κόμβους του δικτύου. Ο κόμβος που εξετάζεται κάθε φορά αποτελεί τον ριζικό κόμβο στο δένδρου που του αντιστοιχεί. Τα δυαδικά δένδρα με νήματα είναι ειδικού τύπου, καθένα από αυτά έχει μία ρίζα και ένα δεξιό υποδένδρο (right subtree) χωρίς την παρουσία αριστερού υποδένδρου (left subtree). Επιπλέον, κάθε κόμβος του δένδρου, εκτός της ρίζας, έχει το πολύ δύο νήματα (threads): ένα δεξί νήμα συνδεδεμένο με τη ρίζα του δένδρου (Right Thread) και ένα αριστερό νήμα (Left Thread) συνδεδεμένο με ένα προσεκτικά επιλεγμένο αδελφό (sibling) όπως περιγράφεται στη συνέχεια.

Οι γείτονες της ρίζας ταξινομούνται πρώτα σε μία λίστα με αύξουσα σειρά, με βάση την τιμή βάρους που τους συνδέει με τη ρίζα. Τώρα, αν $T = T_1, T_2, \dots T_m$ είναι το σύνολο των γειτονικών κόμβων ενός χρήστη (είναι αδέρφια επειδή έχουν τον ίδιο γονέα, τη ρίζα), τότε το δυαδικό δένδρο με νήματα $B(root)$ μπορεί να οριστεί ως εξής:

1. Εάν $m = 0$, τότε το $B(root)$ είναι κενό.
2. Εάν $m > 0$, το δεξί υποδένδρο (subtree) του $B(root)$ είναι $B(T_1), B(T_2), \dots B(T_m)$, δηλαδή σχηματίζεται από τις ρίζες των γειτονικών κόμβων. Πιο συγκεκριμένα:

   α) Τα παιδιά τοποθετούνται, ένα σε κάθε επίπεδο του δένδρου, σύμφωνα με τη σειρά τους στην ταξινομημένη λίστα. Ο δεξιός σύνδεσμος κάθε παιδιού δείχνει τον επόμενο αδελφό (επόμενο στοιχείο της ταξινομημένης λίστας) ο οποίος βρίσκεται στο επόμενο επίπεδο του δένδρου.

   β) ο αριστερός σύνδεσμος κάθε παιδιού αποτελείται από το δικό του υποδίκτυο γειτόνων, με άλλα λόγια ένα δένδρο με αυτό το παιδί ως ρίζα.
3. Το δεξί νήμα κάθε παιδιού (εικονιζόμενο με ένα βέλος) συνδέεται με τη ρίζα του δένδρου.
4. Το αριστερό νήμα κάθε παιδιού T_i, $i \in [1 \dots m]$ συνδέεται με έναν αδελφό $T_{i'}$, $i' \in [1 \dots m], i \neq i'$, έτσι ώστε:

   Ανίσωση 3.6 - Κριτήριο προσθήκης αριστερού νήματος

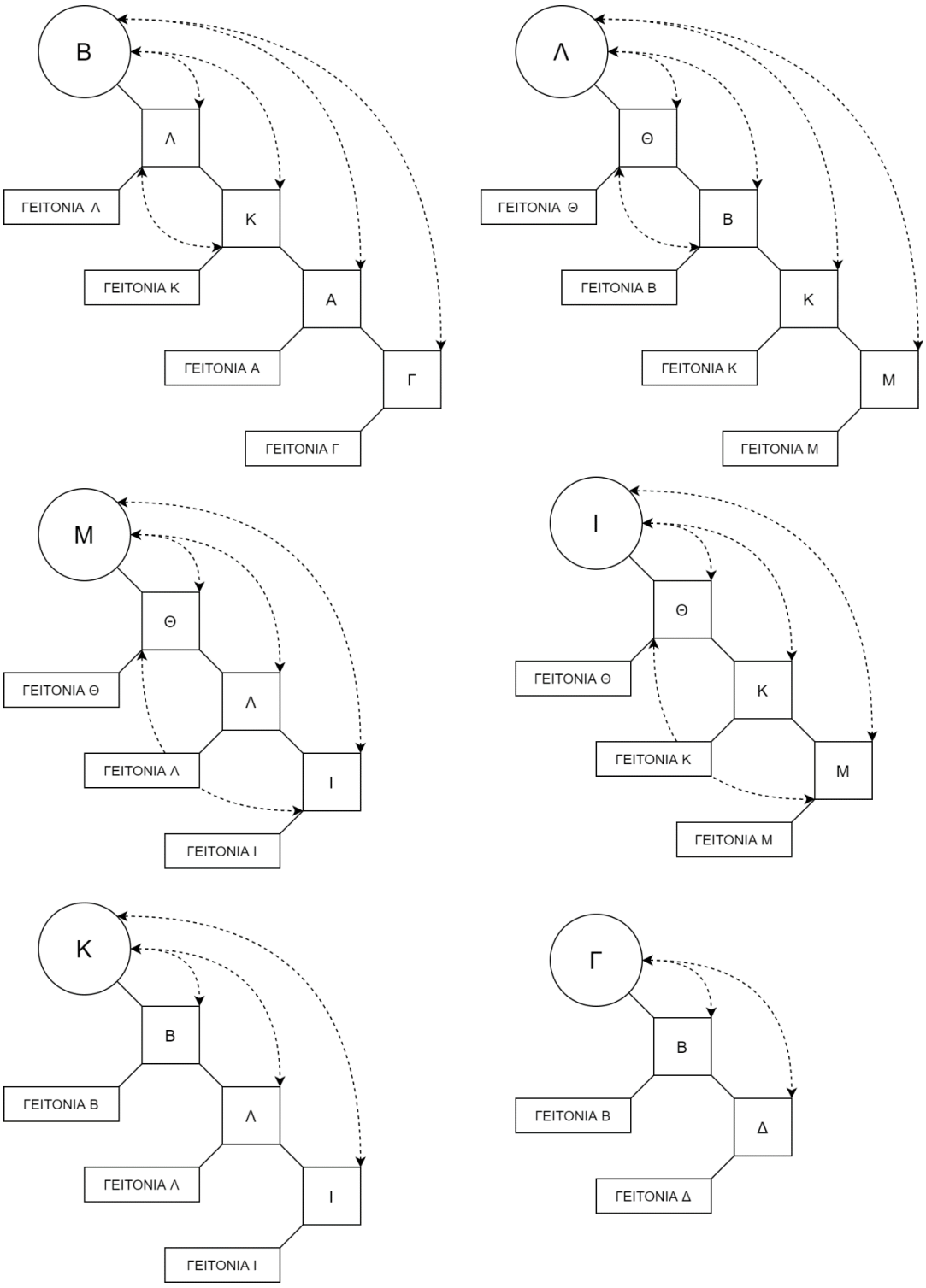
$$\frac{w_{T_i,T_i,} + w_{T_{i'},root}}{2} > w_{root,T_i} \tag{3.6}$$

   Η ανισότητα 3.6 δηλώνει ότι η έμμεση σχέση μεταξύ T_i και ρίζας μέσω του T_i, είναι ισχυρότερη σε σύγκριση με την άμεση σχέση τους. Δεν έχουν όλα τα παιδιά αριστερό νήμα. Για να εντοπίσουμε ένα αριστερό νήμα αναζητούμε μια σχέση μεταξύ T_i και ενός αδελφού $T_{i'}$, έτσι ώστε να ικανοποιείται η σχέση (3.6). Σημειώνεται ότι ο $T_{i'}$ μπορεί να βρεθεί μόνο σε χαμηλότερο επίπεδο από τον T_i, επειδή εάν ο $T_{i'}$ βρισκόταν σε υψηλότερο επίπεδο από τον T_i τότε $w_{T_{i',root}} > w_{T_{i,root}}$ και η ανίσωση (3.6) δεν θα ικανοποιούνταν. Το αριστερό νήμα απεικονίζεται με ένα βέλος που δείχνει από τον T_i προς τον $T_{i'}$.

Για παράδειγμα, για την γειτονιά του κόμβου Β έχουμε. Ο κόμβος Β είναι η ρίζα και τοποθετείται στο επίπεδο μηδέν του δένδρου. Η ταξινομημένη λίστα γειτόνων είναι Λ, Κ, Α, Γ ($w_{B\Lambda} = 0.58$, $w_{BK} = 0.77$, $w_{BA} = 0.9$, $w_{B\Gamma} = 0.96$).

- Τώρα, τα παιδιά τοποθετούνται, ένα σε κάθε επίπεδο του δένδρου κατά αύξουσα σειρά, επομένως, ο κόμβος Λ τοποθετείται στο πρώτο επίπεδο, ο Κ στο δεύτερο, ο Α στο τρίτο και ο Γ στο τέταρτο.
- Το δεξί υποδένδρο (subtree) του $RightSubTree(B)$ είναι τώρα ολοκληρωμένο προσθέτοντας τις αντίστοιχες γειτονιές στους αριστερούς συνδέσμους όλων των παιδιών.
- Έπειτα, προστίθενται τα νήματα (threads). Τα δεξιά νήματα συνδέονται με τη ρίζα Β. Το αριστερό νήμα χρησιμοποιείται για να βρει μια ισχυρότερη διαδρομή από τη ρίζα προς ένα παιδί όπως αναφέρθηκε.

Για παράδειγμα, ο Λ συνδέεται με τη ρίζα με βάρος 0,58. Ωστόσο, υπάρχει ένας αδελφός του Λ σε χαμηλότερο επίπεδο του δένδρου, ο Κ τέτοιος ώστε $w_{\Lambda K} = 0{,}82 > w_{KB}$. Επιπλέον, $w_{BK} = 0{,}77$ και $(w_{\Lambda K} + w_{BK})/2 = \frac{0.82+0{,}77}{2} = 0{,}795 > w_{B\Lambda}$. Έτσι, τοποθετούμε ένα αριστερό νήμα (left thread) από τον κόμβο Λ στον κόμβο Κ. Η Εικόνα 5 απεικονίζει τα δυαδικά δέντρα με νήματα (threaded binary trees) για τους κόμβους Β, Λ, Μ, Ι, Κ, Γ.



Εικόνα 6. Δυαδικά δένδρα με νήματα των κόμβων Β, Λ, Μ, Ι, Κ, Γ

3.2.1.1 Ψευδοκώδικας & διάγραμμα ροής

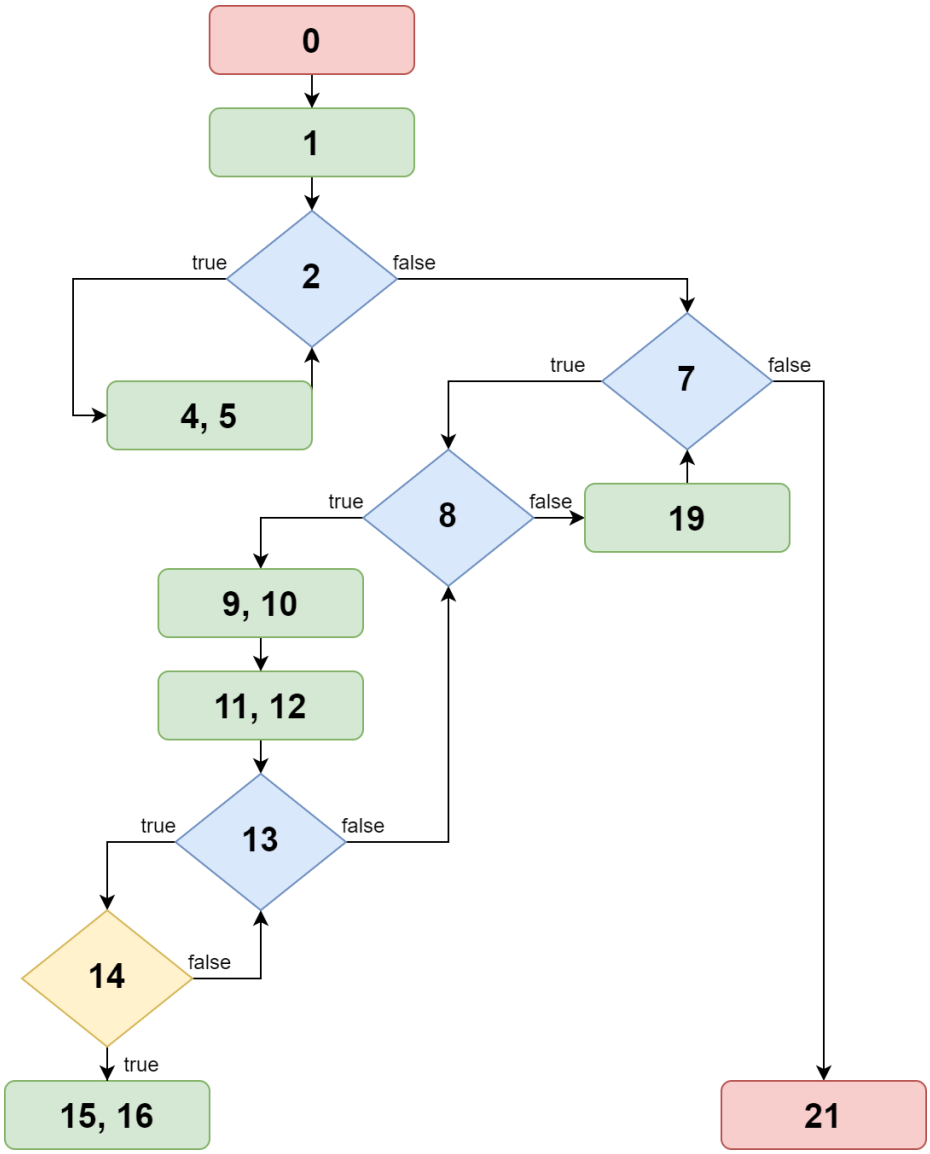
Για $G = (V, E)$ σταθμισμένο μη κατευθυνόμενο γράφημα όπου V, E είναι τα σύνολα των κόμβων και των ακμών αντίστοιχα.

```
(0)   Αρχή
(1)   n ← κόμβος
(2)   όσο n∈V  επανέλαβε
(3)       {#ταξινόμηση γειτόνων T∈m για κόμβο n
(4)       L(n)= [T1, T2,… Tm]  #ταξινομημένη λίστα γειτόνων
(5)        n = επόμενος κόμβος}
(6)   τέλος επανάληψης #n∈V
(7)   Για (n∈V)
(8)        Για i∈m
(9)          LLi ← community(i)  #τιμή αριστερού συνδέσμου
(10)         RLi ← L(i+1)  # τιμή δεξιού συνδέσμου
(11)         RTi ← n # τιμή δεξιού νήματος
(12)         max = w(i,j )  #υπολογισμοί αριστερού νήματος
(13)         Για j∈[i+1,… m]
(14)             Εαν (w_(i,j)+w(j,n))/2  >max
(15)             {max ← w(i,j)+w(j,n))/2
(16)             LTi = j #τιμή αριστερού νήματος}
(17)             Τέλος επανάληψης #j∈[i+1,… m]
(18)       Τέλος επανάληψης #i∈m
(19)  αποθήκευση LLi, RLi, RTi, LTi
(20)  Τέλος επανάληψης #n∈V
(21)  Τέλος
```

Η παραγωγή του δυαδικού δένδρου με νήματα (threaded binary tree) ξεκινά στην γραμμή 7. Λαμβάνοντας ως παράμετρο έναν κόμβο σε μια ταξινομημένη λίστα γειτόνων και εφαρμόζοντας τους απαραίτητους υπολογισμούς που περιγράφηκαν εκτελείται η αλγοριθμική διαδικασία για την παραγωγή

δένδρων. Η μεταβλητή max (γραμμή 12), είναι απαραίτητη για τον υπολογισμό των αριστερών νημάτων των παιδιών, ενημερώνεται κάθε φορά που ικανοποιείται η συνθήκη τις Εξίσωσης (3.6) .

Οι γραμμές του ψευδοκωδικα που αποτελούνται αποκλειστικά από σχόλια δεν αναπαρίστανται στα διαγράμματα ροής.



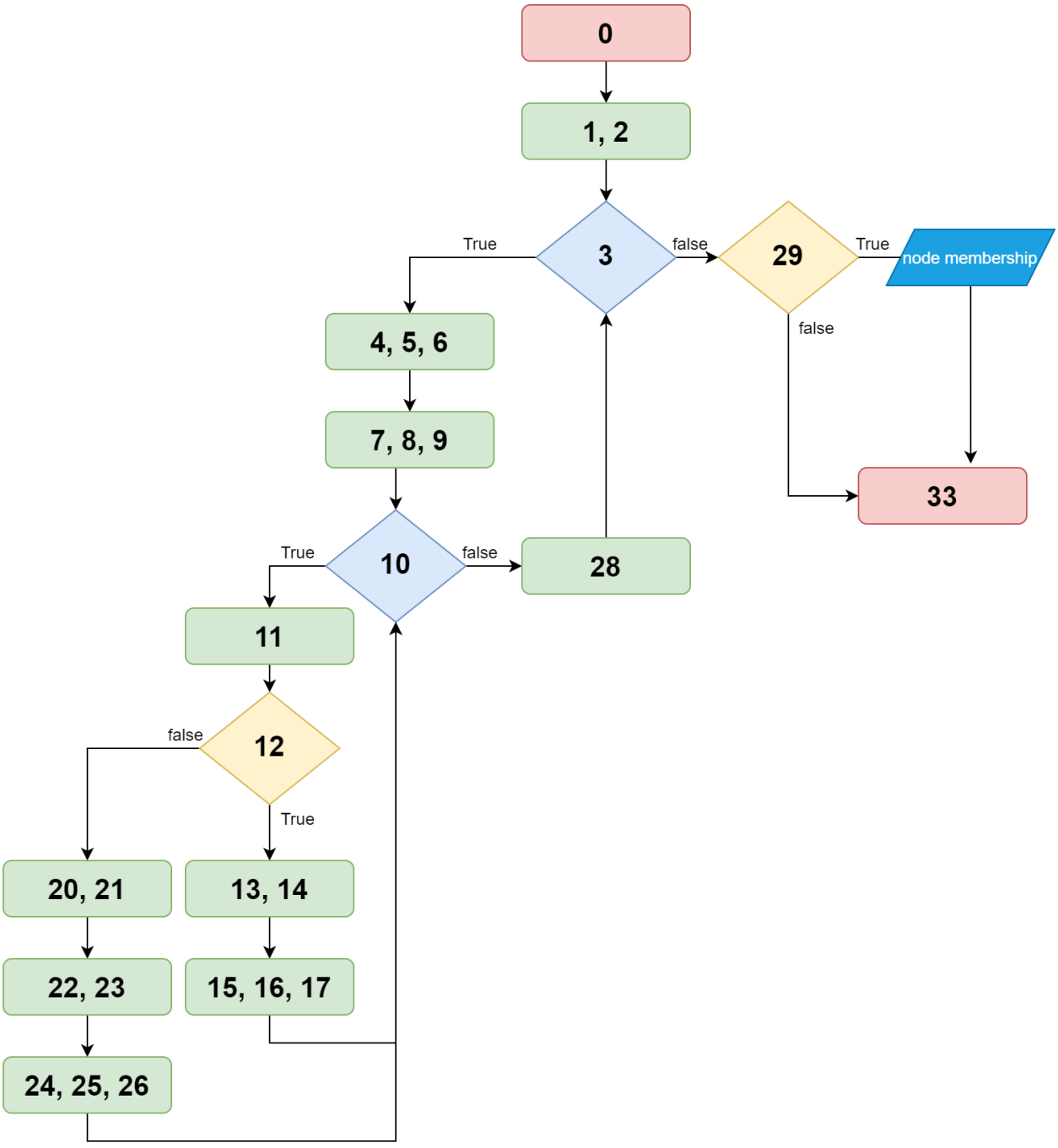
Εικόνα 7. Διάγραμμα ροής παραγωγής διαδίκων δένδρων με νήματα

3.2.2 Ανάλυση διαδρομής

Ο αλγόριθμος για την ανάλυση διαδρομής χρησιμοποιείται για να εξετάσει ποιοι χρήστες του δικτύου ενδέχεται να ανήκουν στην κοινότητα στόχο (Ενότητα 4.2, Κεφάλαιο Υπολογιστική εμπειρία) σύμφωνα

με το μοντέλο της εργασίας. Παρατίθεται ο ψευδοκώδικας και το διάγραμμα ροής όπως ακριβώς και για την προηγούμενη αλγοριθμική διαδικασία.

```
(0)   Αρχή
(1)   C =[M1,M2,…Mk]  #κοινότητα στόχος με μέλη k,M∈k
(2)   d = q #όπου Q η διάμετρος της κοινότητας στόχου
(3)   Για tree∈ trees
(4)      Path(root)  #δυαδικό δένδρο με νήματα input (pointer ριζας)
(5)           curPath ← 0
(6)           Tact ← root
(7)           Updates = 0
(8)           lenght = 0.0
(9)           activeList = set(root.neighbors)
(10)          Για  i∈ activeList
(11)                Ti = RLTact
(12)                Εάν LT_Ti≠0
(13)                    {length ← length + w(root,LTTi)+ w(LTTi,Ti)
(14)                      updates←updates+2
(15)                      curPath ← curPath + set(Tact → LTTi → Ti)
(16)                      Tact ← LTTi
(17)                      activeList←activeList-curPath-set(Ti…LT_Ti)
(18)                      #αφαίρεση curPath&≪χαμηλότερων≫ γειτόνων}
(19)                Αλλιώς
(20)                     {length ← length + w(root,Ti)
(21)                      updates=updates+1
(22)                      curPath = curPath + set(T → Ti)
(23)                      Ti ← Tact
(24)                      activeList ← activeList - curPath}
(25)     Root ← Ti
(26)         Τελος επανάληψης #i ∈ activeList
(27)     pathStrength = length/updates
(28)     Εάν (root∈C)&(pathStrength>δ)
(19)     #ο κόμβος ανήκει στην κοινότητα στόχο σύμφωνα με το μοντέλο
(30)     #τελος Path
(31)  Τέλος επανάληψης #tree∈ trees
(32)  Τέλος
```



Εικόνα 8. Διάγραμμα ροής ανάλυσης διαδρομής

Ο αλγόριθμος ξεκινά ορίζοντας την κοινότητα στόχο C (Ενότητα 4.2, Κεφάλαιο Υπολογιστική Εμπειρία), και τη διάμετρό της, δ_C (γραμμές 1, 2).

- Για όλα τα δυαδικά δένδρα με νήματα του δικτύου, εκτελείται η συνάρτηση $Path(\ treeNode)$, γραμμή 4. Η συνάρτηση αναζητά την ισχυρότερη διαδρομή μέσω των αριστερών νημάτων (left

threads) τα οποία χρησιμοποιούνται για την αύξηση της τρέχουσας ισχύος της διαδρομής $pathStrength$.

- Σε περίπτωση όπου υπάρχει αδελφός T_i, τέτοιος ώστε να ικανοποιείται η ανισότητα (3.6) : Οι ενημερώσεις διαδρομής ($updates$) αυξάνονται κατά δύο (μία από την ρίζα στο αριστερό νήμα του T_i, LT_{Ti} και μία από το LT_{Ti} στον T_i, γραμμή 14.

- Για να περιορίσουμε τις περιττές επαναλήψεις, απενεργοποιούμε τους κόμβους που βρίσκονται πάνω από το αριστερό νήμα LT_{Ti} (γραμμή 24). Οι απενεργοποιημένοι κόμβοι είναι αδύνατο να αυξήσουν την ισχύ της διαδρομής $pathStrength$, στο επόμενο βήμα.

- Η πιο σημαντική δομή στον αλγόριθμο ανάλυσης διαδρομής είναι η λίστα ενεργών κόμβων, $activeList$. Κάθε φορά που υποβάλλεται σε επεξεργασία ένας νέος κόμβος T_i, η λίστα ενεργών κόμβων ενημερώνεται έτσι ώστε να περιλαμβάνει τους κόμβους που μπορούν να ακολουθήσει η τρέχουσα διαδρομή. Αυτό επιτυγχάνεται, εάν καταργήσουμε από τη λίστα των γειτόνων του T_i τους κόμβους που είναι ήδη στη διαδρομή $currPath$ και τους κόμβους που είναι αδύνατο να αυξήσουν την τρέχουσα ισχύ της διαδρομής.

- Η παράμετρος $length$ ενημερώνεται κάθε φορά από το άθροισμα των βαρών των δύο προαναφερθέντων συνδέσμων (γραμμή 13).

Για παράδειγμα, χρησιμοποιώντας τα δυαδικά δέντρα με νήματα της Εικόνας 5 εξετάζεται αν ο κόμβος Β σύμφωνα με το μοντέλο της εργασίας ανήκει στην κοινότητα C_2.

1. Αρχικά ορίζεται η κοινότητα στόχος C_2, με μέλη Θ, Η, Ε, Ζ και διάμετρο 3. Η επεξεργασία ξεκινά από το δεξιό σύνδεσμο RL το κόμβου που δείχνει το T_{act} δηλαδή, ο δεξιός σύνδεσμος του Β που είναι ο Λ (γραμμή 11).
2. Δεδομένου ότι το αριστερό νήμα του Λ είναι το Κ, το $length$ γίνεται $length = w_{BK} + w_{B\Lambda} = 0{,}77 + 0{,}58 = 1{,}35$ (γραμμή 13) και τα $updates = 2$ (γραμμή 14). Το $currPath$ θα είναι $B \rightarrow K \rightarrow \Lambda$ (γραμμή 15).
3. Το T_{act} δείχνει τώρα στο Κ (γραμμή 16). Η νέα $activeList$ θα είναι η $activeList$ του κόμβου Λ, δηλαδή Λ, Θ, Β, Κ, Μ (εικόνα 5) μείον τους κόμβους της τρέχουσας διαδρομής (Β, Λ, Κ), μείον τους κόμβους πάνω από το $LT_{Ti} = K$ (σε αυτή την περίπτωση Β, Θ, Λ εικόνα 5). Σημειώνεται ότι ο σύνδεσμος $\Lambda \rightarrow \Theta$ δεν μπορεί να αυξήσει την ισχύ της διαδρομής. Έτσι η νέα $activeList$ περιλαμβάνει το Μ. Τέλος, ο κόμβος $T_i = \Lambda$ θα γίνει $root$ (γραμμή 25).
4. Τώρα, στη δεύτερη επανάληψη, η επεξεργασία θα ξεκινήσει από το δεξιό σύνδεσμο του Κ που είναι ο Μ, γραμμή 11 (γειτονιά για $root = \Lambda$).
5. Επειδή ο Μ δεν έχει αριστερό νήμα (γραμμή 12), η συνέχεια είναι στην γραμμή 20. Το $length$ γινεται $length = 1{,}35 + 0{,}84 = 2{,}19$ και τα $updates = 2 + 1 = 3$ (γραμμές 20,21).
6. Το $currPath$ ανανεώνεται σε $B \rightarrow K \rightarrow \Lambda \rightarrow M$. Το T_{act} τώρα αναφέρεται στο νέο κόμβο προς επεξεργασία δηλαδή τον $T_i = M$. Όμως αυτή την φορά ο Μ έρχεται ως ρίζα (γραμμή 25) και οι γειτονιές αλλάζουν. Δηλαδή η επεξεργασία συνεχίζεται από τη γειτονιά του Μ και η ενεργή λίστα ($activeList$) θα περιλαμβάνει: Μ, Θ, Λ, Ι (εικόνα 5), μείον τους κόμβους που βρίσκονται ήδη στην

διαδρομή (Β, Κ, Λ, Μ) άρα $activeList = \Theta, I$. Σημειώνεται ότι προηγουμένως εξαιρέθηκε ο κόμβος Θ, όμως πλέον περιλαμβάνεται στην $activeList$ καθώς εξετάζεται διαφορετική γειτονιά.

7. Η επεξεργασία ξεκινά από το δεξιό κόμβο του Μ που είναι το Θ. Επειδή ο Θ έχει αριστερό νήμα (left thread) που δείχνει στο Ι το $length$ γίνεται $length = 2{,}19 + w_{M\Theta} + w_{MI} = 3{,}91$ και τα $updates = 3 + 2 = 5$.
8. Τώρα η τρέχουσα διαδρομή γίνεται $currPath = B \to K \to \Lambda \to M \to I \to \Theta$. Ο επόμενος κόμβος προς επεξεργασία είναι ο Θ αλλά έχουμε φτάσει ήδη στην κοινότητα στόχο C_2.
9. Τέλος, υπολογίζεται η ισχύς διαδρομής $pathStrength = \frac{length}{updates} = \frac{3{,}91}{5} \approx 0{,}78$ (γραμμή 27) η οποία είναι μικρότερη από τη διάμετρο της κοινότητας στόχου άρα ο χρήστης Β ανήκει στην κοινότητα C_2.

ΚΕΦΑΛΑΙΟ 4 ΥΠΟΛΟΓΙΣΤΙΚΗ ΕΜΠΕΙΡΙΑ

4.1 ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ

Ο αλγόριθμος υλοποιήθηκε σε αντικειμενοστραφές περιβάλλον Python 3.8. Κάθε κόμβος του δικτύου αποθηκεύεται με τη μορφή αντικειμένου κλάσης και έπειτα σχηματίζονται τα δυαδικά δένδρα με νήματα (threaded binary trees) που περιγράφονται στη συνέχεια.

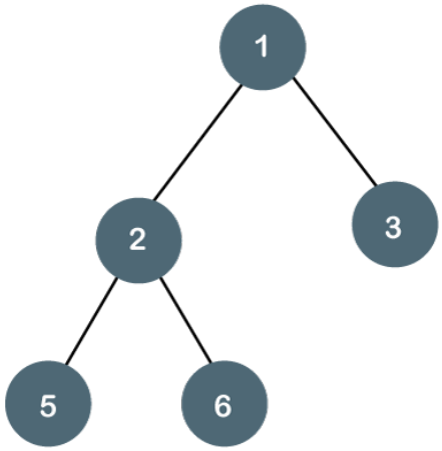
4.1.1 Δομές Δεδομένων & Υλοποίηση με Python

4.1.1.1 Δένδρα

Η κύρια δομή δεδομένων στην υλοποίηση του μοντέλου είναι τα δένδρα. Στην επιστήμη των υπολογιστών, ένα δένδρο είναι ένας ευρέως χρησιμοποιούμενος τύπος αφηρημένων δεδομένων που προσομοιώνει μια ιεραρχική δομή δένδρου, με ριζική τιμή και δευτερεύοντα δένδρα παιδιών με γονικό κόμβο, που αντιπροσωπεύονται ως σύνολο συνδεδεμένων κόμβων. Μια δομή δεδομένων δένδρου μπορεί να οριστεί αναδρομικά ως μια συλλογή κόμβων (ξεκινώντας από έναν ριζικό κόμβο), όπου κάθε κόμβος είναι μια δομή δεδομένων που αποτελείται από μια τιμή, μαζί με μια λίστα αναφορών σε κόμβους (τα "παιδιά"), με περιορισμούς που δεν αναπαράγουν καμία αναφορά και κανένας δεν δείχνει τη ρίζα. Εναλλακτικά, ένα δένδρο μπορεί να οριστεί αφηρημένα ως ένα γενικό σύνολο που εμπεριέχει κάποια μορφή ταξινόμησης, ανάλογα με το είδος των ιεραρχικών δεδομένων που καλείται να αναπαραστήσεις, με βάση μία τιμή που αποδίδεται σε κάθε κόμβο (Wikipedia, 2011). Και οι δύο αυτές προοπτικές είναι χρήσιμες: ενώ ένα δένδρο μπορεί να αναλυθεί μαθηματικά στο σύνολό του, όταν στην πραγματικότητα αντιπροσωπεύεται ως δομή δεδομένων, συνήθως αντιπροσωπεύεται και υπόκειται σε επεξεργασία ανά κόμβο, ξεχωριστά (αντί ως σύνολο κόμβων και λίστα γειτνίασης ακμών μεταξύ κόμβων, όπως μπορεί να αντιπροσωπεύει ένα διάγραμμα, για παράδειγμα). Για παράδειγμα, κοιτάζοντας ένα δένδρο στο σύνολό του, μπορεί κανείς να μιλήσει για το «γονικό κόμβο» ενός δεδομένου κόμβου, αλλά γενικά, ως δομή δεδομένων, ένας δεδομένος κόμβος περιέχει μόνο τη λίστα των παιδιών του και δεν περιέχει αναφορά στο γονέα του (εάν υπάρχει).

4.1.1.1.1 Δυαδικά δένδρα

Δυαδικό δένδρο (binary tree εικόνα 8): Ένα δένδρο του οποίου τα στοιχεία έχουν το πολύ 2 παιδιά (απογόνους) ονομάζεται δυαδικό δένδρο. Δεδομένου ότι κάθε στοιχείο σε ένα δυαδικό δένδρο μπορεί να έχει μόνο 2 παιδιά, συνήθως τα ονομάζουμε αριστερά και δεξιά.

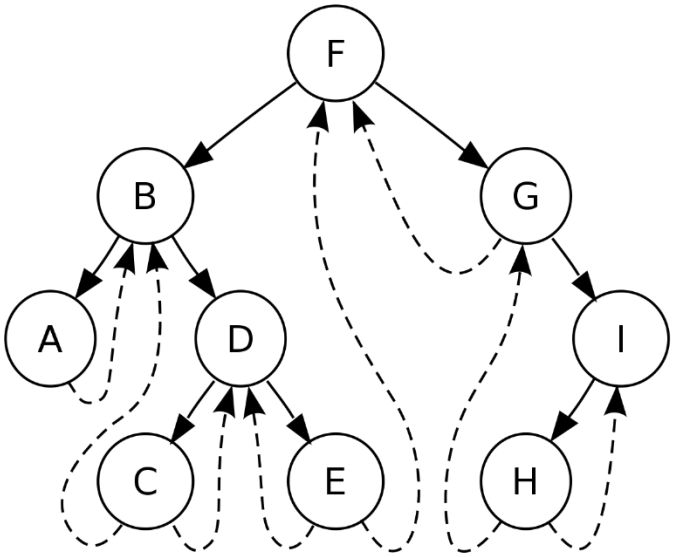


Εικόνα 9 - Δυαδικό δένδρο

Για το δυαδικό δένδρο της Εικόνας 8, ο κόμβος 1 αποτελεί τη ρίζα και βρίσκεται στο επίπεδο 0 του δένδρου. Ο κόμβος 2 (επίπεδο 1) είναι το αριστερό παιδί και υποδένδρο της ρίζας, και ο 3 το δεξί αντίστοιχα. Τέλος στο επίπεδο 2 του δένδρου βρίσκονται οι κόμβοι 5 (αριστερό παιδί του κόμβου 2) και 6 (δεξί), οι οποίοι κόμβοι είναι αδέλφια καθώς έχουν τον ίδιο γονέα.

4.1.1.1.2 Δυαδικά δένδρα με νήματα

Ένα δυαδικό δένδρο με νήματα (threaded binary tree, Εικόνα 9) είναι μια δυαδική παραλλαγή δένδρου που διευκολύνει τη διέλευση από έναν κόμβο του δένδρου σε κάποιον άλλο που ανήκει σε διαφορετικό επίπεδο. Κάθε κόμβος του δυαδικού δένδρου με νήματα μπορεί να έχει δεξί και αριστερό νήμα (left thread, right thread). Τα νήματα στην Εικόνα 9 απεικονίζονται με βέλη.



Εικόνα 10. Δυαδικό δένδρο με νήματα

4.1.1.2 Δομές ακολουθίας δεδομένων

Οι υπόλοιπες σημαντικές δομές δεδομένων για την υλοποίηση είναι, η λίστα (list), η πλειάδα (tuple), το λεξικό (dictionary) και το σύνολο (set), οι οποίες αποτελούν ακολουθιακές (σειριακές) δομές δεδομένων (sequence data structure). Οι ακολουθίες επιτρέπουν να αποθηκεύονται πολλές τιμές με οργανωμένο και αποτελεσματικό τρόπο στην θέση μνήμης ενός αντικείμενου (οποιοδήποτε δεδομένο στη γλώσσα Python καταχωρείται στην μνήμη ως αντικείμενο). Δεδομένα που δεν αποτελούν ακολουθία είναι οι ακέραιοι και οι δεκαδικοί.

- Λίστες (lists): Οι λίστες είναι ταξινομήσιμες και ένας από πιο ευρέα διαδεδομένους, γενικού σκοπού τύπος δεδομένων στην Python. Χρησιμοποιούνται για την αποθήκευση μιας ακολουθίας δεδομένων σε μια μεταβλητή.

Οι υπόλοιπες δομές (πλειάδα, λεξικό, σύνολο) έχουν ορισμένα πλεονεκτήματα απέναντι στις λίστες.

- Πλειάδες (tuples): η πλειάδα είναι δεδομένο ακολουθίας αμετάβλητου τύπου (immutable data structure), σε αντίθεση με τις λίστες που είναι μεταβλητού τύπου (mutable data structure). Κάτι το οποίο σημαίνει ότι σε μια πλειάδα δεν μπορούν να αλλάξουν, να προσδεθούν ή να αφαιρεθούν δεδομένα από τη στιγμή που δημιουργείται. Η λειτουργία αυτή είναι σημαντική για δεδομένα σταθερού τύπου (fixed data).

- Σύνολο (set): η συγκεκριμένη δομή δεδομένων αποτέλεσε δομικό στοιχείο στην υλοποίηση της $activeList$ (Κεφάλαιο 3, υποενότητα 3.2.2 ανάλυση διαδρομής) και δεν επιτρέπει διπλότυπα (duplicated values). Με άλλα λόγια κάθε στοιχείο σε ένα σύνολο είναι μοναδικό.

- Λεξικό (dictionary): αποτελείται από ζεύγη μεταβλητών κλειδιών και τιμών (key value, pairs). Το λεξικό όπως και η πλειάδα δεν επιτρέπει διπλότυπα και από την έκδοση Python 3.7 και έπειτα είναι ταξινομήσιμο.

Το σημαντικότερο πλεονέκτημα γι' αυτές τις δομές δεδομένων είναι ότι μπορούν να διακρατήσουν οποιονδήποτε τύπο δεδομένων όπως ακεραίους (integers), δεκαδικούς (floating point values), αλφαριθμητικά (strings), αντικείμενα (objects) ακόμη και άλλα λεξικά, σύνολα, λίστες, και πλειάδες. Τέλος οι δομές δεδομένων ακολουθίας δεν χρειάζεται να είναι ομογενείς που σημαίνει ότι μια λίστα μπορεί να αποτελείται από ένα δεκαδικό, μια άλλη λίστα, ένα σύνολο και ένα λεξικό.

4.1.1.3 Κλάση TreeNode

Κάθε κόμβος του δικτύου αποτυπώνεται ως αντικείμενο και υπόκειται σε διαχείριση με τη βοήθεια της κλάσης $TreeNode$. Για τη διεκπεραίωση του μοντέλου τα μέλη της κλάσης (class members) είναι:

- $NodeID$: ο κωδικός του κόμβου που εξετάζεται.
- $right\ link\ value$: η τιμή του δεξιού συνδέσμου του κόμβου.
- $right\ link\ object$: το αντικείμενο του δεξιού συνδέσμου του κόμβου (θέση μνήμης).
- $right\ thread$: η τιμή του δεξιού νήματος του εξεταζόμενου κόμβου.
- $left\ thread$: η τιμή του αριστερού νήματος του κόμβου.
- $left\ link$: αριστερός σύνδεσμος, για την αποθήκευση των γειτονιών των παιδιών των δένδρων.

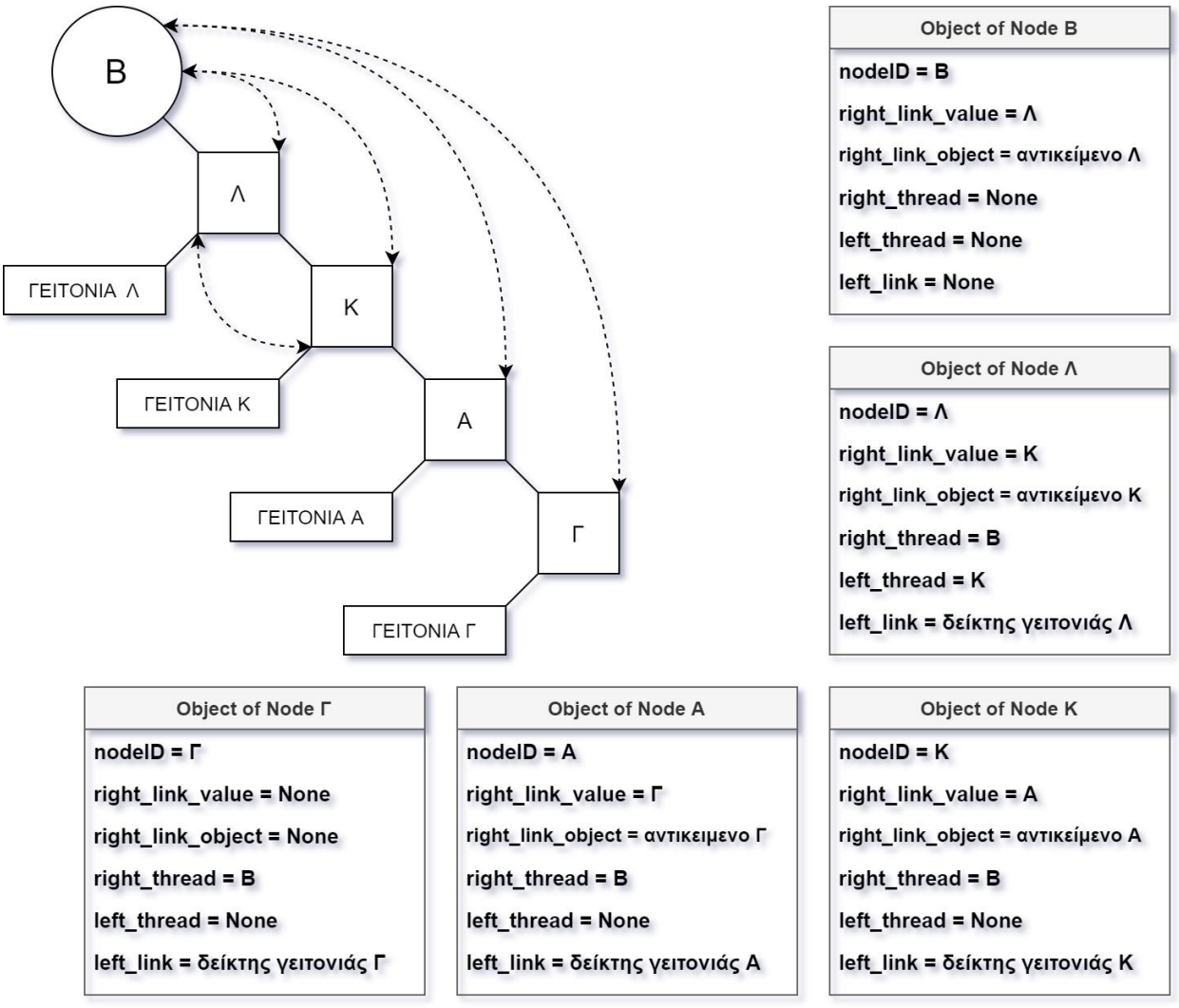
Για τη δημιουργία αντικειμένου κόμβου είναι απαραίτητη η ύπαρξη μη διπλοτύπου $NodeID$, τα υπόλοιπα μέλη αρχικοποιούνται με την τιμή $None$ και έπειτα αποδίδονται τιμές σε αυτά με την κλήση

καταλλήλων συναρτήσεων της κλάσης (class constructors) οι οποίες περιγράφονται στην επόμενη υποενότητα Python Files.

Πηγαίος κώδικας

```
class TreeNode:
    def __init__(self, node):
        self.nodeID = node
        self.right_link_value = None
        self.right_link_object = None
        self.right_thread = None
        self.left_thread = None
        self.left_link = None
```

Για παράδειγμα, τα αντικείμενα-κόμβοι του δυαδικού δένδρου νημάτων με γονικό κόμβο Β (Εικόνα 5) αποθηκεύονται με τον τρόπο που παρουσιάζεται στην Εικόνα 10.



Εικόνα 11. Αντικείμενα-κόμβοι του δυαδικού δένδρου με νήματα και ριζικό κόμβο Β

4.1.1.4 Python files

Ο πηγαίος κώδικας της εργασίας βρίσκεται στο τέλος του εγγράφου και στο https://github.com/kotsinas/thesis_repository. Το repository αποτελείται από τέσσερα αρχεία:

- main.py,
- tree_traversals.py,
- tree.py,
- communities.py

Τα κύρια αρχεία είναι τα main.py και tree_traversals.py τα άλλα δύο είναι βοηθητικά και χρησιμοποιήθηκαν ως εργαλεία debugging για τα τμήματα του κώδικα που αφορούν τα δένδρα και τις οριοθετημένες κοινότητες αντίστοιχα.

Το αρχείο tree_traversals.py περιέχει τη παραγωγή των δένδρων καθώς και τις απαραίτητες προσπελάσεις σε αυτά για την εκπλήρωση των υπολογισμών που περιγράφονται στην ενότητα του Κεφαλαίου 3 Αλγόριθμος Ανίχνευσης Κοινοτήτων.

Οι κόμβοι των δένδρων αποτελούν αντικείμενα της κλάσης $TreeNode$ με συναρτήσεις constructors,

- $add\_right\_child()$: προσθήκη δεξιών συνδέσμων στους κόμβους.
- $add\_right\_thread()$: απόδοση τιμών στα δεξιά νήματα των κόμβων.
- $add\_left\_thread()$: απόδοση τιμών στα αριστερά νήματα (όπου υπάρχουν).
- $add\_community()$: προσθήκη των αντίστοιχων κοινοτήτων των παιδιών ως αριστερών συνδέσμων.
- $right\_travers()$: προσπέλαση δεξιού υποδένδρου.
- $get\_node\_ids()$: εργαλείο debugging.

Οι υπόλοιπες συναρτήσεις του αρχείου tree_traversals.py που καλούνται έπειτα στη main.py είναι:

- $build\_tree()$: κατασκευή δένδρων, κλήση των class constructors $add\_right\_child()$, $add\_right\_thread()$, $add\_left\_thread()$.
- $build\_communities()$: προσθήκη κοινοτήτων, κατάλληλη κλήση του class constructor $add\_community()$.
- $get\_subtree()$: για την απομόνωση του δένδρου από τα αριστερά νήματα (left threads) και κάτω (ενέργεια που απαιτείται για τη μείωση των περιττών επαναλήψεων στο αλγοριθμικό μέρος της ανάλυσης διαδρομής)
- $path\_sim()$: η συνάρτηση $Path$ στην Ανάλυση Διαδρομής.

Η κύρια βιβλιοθήκη που χρησιμοποιείται στο αρχείο main.py είναι η $NetworkX(nx)$, η οποία αποτελεί ένα πακέτο της Python για τη δημιουργία, χειρισμό και τη μελέτη της δομής, της δυναμικής και των λειτουργιών σύνθετων γράφων και υποστηρίζεται από τις εκδόσεις Python 3.6 και έπειτα.

- $g = nx.read\_edgelisT()$: δημιουργία αντικειμένου γράφου g από λίστα με τις ακμές κόμβων.
  - $g.nodes$: iterator (αντικείμενο προς προσπέλαση) κόμβων.
  - $g.edges$: iterator ακμών.
  - $g.add\_weighted\_edges()$: προσθήκη βαρών στις ακμές του γράφου.

- $g.adjacency(\,)$: iterator με μορφή: ((κόμβος 1, γειτνίασης του 1), (κόμβος2, γειτνίασης του 2),…, (κόμβος m, γειτνίασης του m)) για όλους τους κόμβους m του δικτύου.

- $boolean\ value = nx.is\_conected(g)$: επιστρέφει true αν ο Γράφος είναι συνεκτικός και false στην αντίθετη περίπτωση. Στα μαθηματικά και την επιστήμη των υπολογιστών, η συνεκτικότητα είναι μια από τις βασικές έννοιες της θεωρίας γραφημάτων. Ένα γράφημα καλείται συνεκτικό εάν κάθε ζεύγος κορυφών του είναι συνδεδεμένο. Αυτό σημαίνει ότι υπάρχει μια διαδρομή μεταξύ κάθε ζεύγους κορυφών. Ένα μη κατευθυνόμενο γράφημα που δεν είναι συνεκτικό ονομάζεται μη συνεκτικό. Επομένως, ένα μη κατευθυνόμενο γράφημα G αποσυνδέεται εάν υπάρχουν δυο κορυφές, τέτοιες ώστε καμία διαδρομή του G να μην περιέχει αυτές τις κορυφές ως τελικά σημεία.
- $d = nx.diameter(g)$: επιστρέφει τη διάμετρο του γράφου. Η διάμετρος d ενός γραφήματος είναι η μέγιστη εκκεντρότητα οποιασδήποτε κορυφής στο γράφημα. Δηλαδή, είναι η μεγαλύτερη απόσταση μεταξύ οποιουδήποτε ζεύγους κορυφών.

Οι υπόλοιπες συναρτήσεις του αρχείου main.py είναι:

- $sort\_neighbors(\,)$: αύξουσα ταξινόμηση των κόμβων γειτνίασης ενός κόμβου.
- $count\_set\_bits(\,)$: καταμέτρηση των 1 στις δεκαψηφιες δυαδικές τιμές των κόμβων που αφορούν τα χαρακτηριστικά που εκπληρώνουν οι χρήστες.
- $str\_bitwise\_xor(\,)$: υπολογισμός διανύσματος XOR, ενέργεια απαραίτητη για την απόδοση βαρών στους κόμβους.
- $rand\_key(\,)$: ορίζεται πόσα κοινά χαρακτηριστικά θα έχουν οι χρήστες του δικτύου και πόσα θα είναι τυχαία.
- $ncd(\,)$: υπολογισμός του βαθμού συνεκτικότητας των κόμβων (Network connectivity degree)
- $compute\_acc(\,)$: υπολογισμός του μέσου βαθμού συνεκτικότητας (average community connectivity) για τις οριοθετημένες κοινότητες του δικτύου που εξετάζεται.
- $dia(\,)$: υπολογισμός των διαμέτρων των κοινοτήτων με κλήση $d = nx.diameter(C)$
- $declare\_membership(\,)$: υλοποίηση απαραίτητων ελέγχων και εξακρίβωση ποιοι χρήστες του δικτύου ανήκουν στη κοινότητα στόχο σύμφωνα με το μοντέλο της εργασίας.

4.1.2 Υλοποίηση με Αναδρομή

Στην επιστήμη των υπολογιστών οι αναδρομικές συναρτήσεις καλούν μόνες τους τον εαυτό τους, επαναληπτικά, έως ότου ολοκληρωθεί η διαδικασία την οποία καλούνται να επιτελέσουν. Γενικά τα δένδρα αποτελούν μια αναδρομική μορφή δεδομένων. Έτσι αρχικά, για τη δημιουργία των δένδρων της περίπτωσης που εξετάζεται εφαρμόστηκε αναδρομή στις συναρτήσεις $add\_right\_child(\,)$, $right\_travers(\,)$.

- Η συνάρτηση $add\_right\_child(\,)$, καλείται για τη ρίζα του δένδρου, δέχεται ως παράμετρο το δεύτερο στοιχείο (κόμβο) των ταξινομημένων γειτόνων και ανακαλείται έως ότου προσπελαστεί όλη η ταξινομημένη λίστα, προσθέτοντας κάθε φορά ως δεξί σύνδεσμο τον επόμενο κόμβο στη λίστα.

Πηγαίος κώδικας

```
(1)  def   add_right_child(self,next_element)
(2)    if  self.righ_link:
(3)        self.right_link.add_right_child(next_element)
(4)    else
(5)        self.right_link = next_element_of_the_list
```

Στη γραμμή 2 πραγματοποιείται ο έλεγχος εάν υπάρχει δεξιός σύνδεσμος στον κόμβο που εξετάζεται ($self$). Εάν υπάρχει ανακαλείται η συνάρτηση για τον δεξί σύνδεσμο του κόμβου ($self.right\_link$). Αλλιώς στην αντίθετη περίπτωση το επόμενο στοιχείο της λίστας ($next\_element\_of\_the\_list$) γίνεται ο δεξιός σύνδεσμος του κόμβου.

- Η συνάρτηση $right\_travers( )$ καλείται, για τη ρίζα αναδρομικά, έως ότου φτάσει στο μέγιστο βάθος του δένδρου και μας επιστρέφει μια λίστα με όλους τους κόμβους του δεξιού υποδένδρου.

Πηγαίος κώδικας

```
(1)  def    right_travers(self)
(2)         rightSubtree=[]
(3)         rightSubtree.append(self)
(4)         if self.right_link:
(5)            rightSubtree+=self.right_link.right_travers()
(6)         return rightSubtree
```

Στη γραμμή 2 αρχικοποιείται η κενή λίστα $rightSubtree$ και στο επόμενο βήμα προστίθεται στη λίστα ο κόμβος που βρίσκεται σε επεξεργασία, $self$, (στο πρωτο call function η ρίζα, γραμμή 3). Έπειτα γίνεται ο έλεγχος εάν υπάρχει δεξιός σύνδεσμος (γραμμή 4) και όσο αυτή η συνθήκη ικανοποιείται η συνάρτηση ξανακαλείται για τον δεξιό σύνδεσμο. Μόλις το δένδρο φτάσει στο μέγιστο βάθος, δηλαδή δεν υπάρχει επόμενος δεξιός σύνδεσμος η συνάρτηση μας επιστρέφει τη λίστα $rightSubtree$.

4.1.3 Υλοποίηση χωρίς Αναδρομή

Παρόλα αυτά μια αναδρομή μεγάλου βάθους αποτελεί μια διεργασία υψηλού υπολογιστικού κόστους. Πιο συγκεκριμένα οι μεγάλου βάθους αναδρομές υπερφορτώνουν τη stack memory (την περιοχή της μνήμη του προγράμματος που συγκρατεί τους pointers των μεταβλητών και τις κλήσεις των συναρτήσεων - function calls. Όσο μεγαλύτερη αναδρομή τόσα περισσότερα function calls καταχωρούνται στο stack memory) που παραχωρείται στο πρόγραμμα για την εκτέλεσή του, με αποτέλεσμα το πρόγραμμα να δανείζεται συνεχώς μνήμη έως την υπερχείλιση του stack (stack over flow). Αυτό αποτέλεσε κύριο πρόβλημα σε εκτελέσεις του κώδικα για δίκτυα με περισσότερους από 30.000 χρήστες. Έτσι οι αναδρομικές συναρτήσεις $add\_right\_child( )$, $right\_travers( )$ αναδιαμορφώθηκαν ώστε να εκτελούνται απρόσκοπτα παρέχοντας ολοκλήρωση της εκτέλεσης και το

αντίστοιχο αποτέλεσμα χωρίς την ενσωμάτωση αναδρομική κλήσης. Η αλλαγή αυτή βελτίωσε αισθητά τους χρόνους εκτέλεσης σε όλα τα εξεταζόμενα μεγέθη δικτύων (διαγράμματα εικόνων 11, 12, 13).

- $add\_right\_child()$: ουσιαστικά και για τις δύο συναρτήσεις, μεταφέρθηκε η επαναληπτική διαδικασία στο κύριο μέρος της συνάρτησης ώστε να εκτελούνται όλες οι επαναλήψεις μέσα σε ένα function call χωρίς να χρειάζεται η αναδρομική κλήση για περαιτέρω επανάληψη.

  Πηγαίος κώδικας

```
(1)  def   add_right_child(self,next_element_of_the_list)
(2)        current=self
(3)        while current is not None:
(4)              current.right_link:
(5)              current=current.right_link
(6)        else:
(7)              current.right_link= next_element_of_the_list
(8)        return
```

  Η επαναληπτική δομή $while$ (γραμμη 3) αντικατέστησε τις αναδρομικές κλίσεις με τη βοήθεια της μεταβλητής $current$ (γραμμή 2) η οποία αναφέρεται στο κόμβο που υφίσταται επεξεργασία κάθε φορά και ελέγχει την περατότητα της ταξινομημένης λίστας γειτνίασης.

- Ομοίως προσαρμόστηκε και η συνάρτηση $right\_travers()$

  Πηγαίος κώδικας

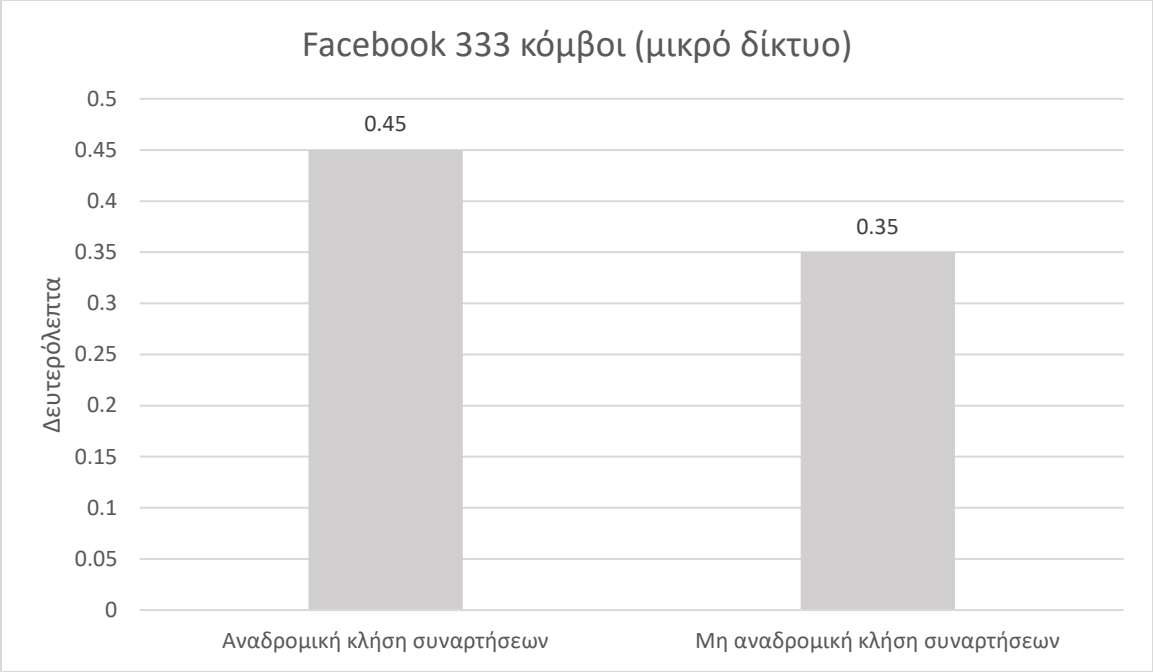
```
(1)  def    right_travers(self)
(2)         rightSubtree=[]
(3)         current=self
(4)         while True:
(5)               if current is not None
(6)               rightSubtree.append(current)
(7)               current=current.right_link
(8)         else:
(9)         return rightSubtree
```

RECURSION-NONRECURSION CHARTS

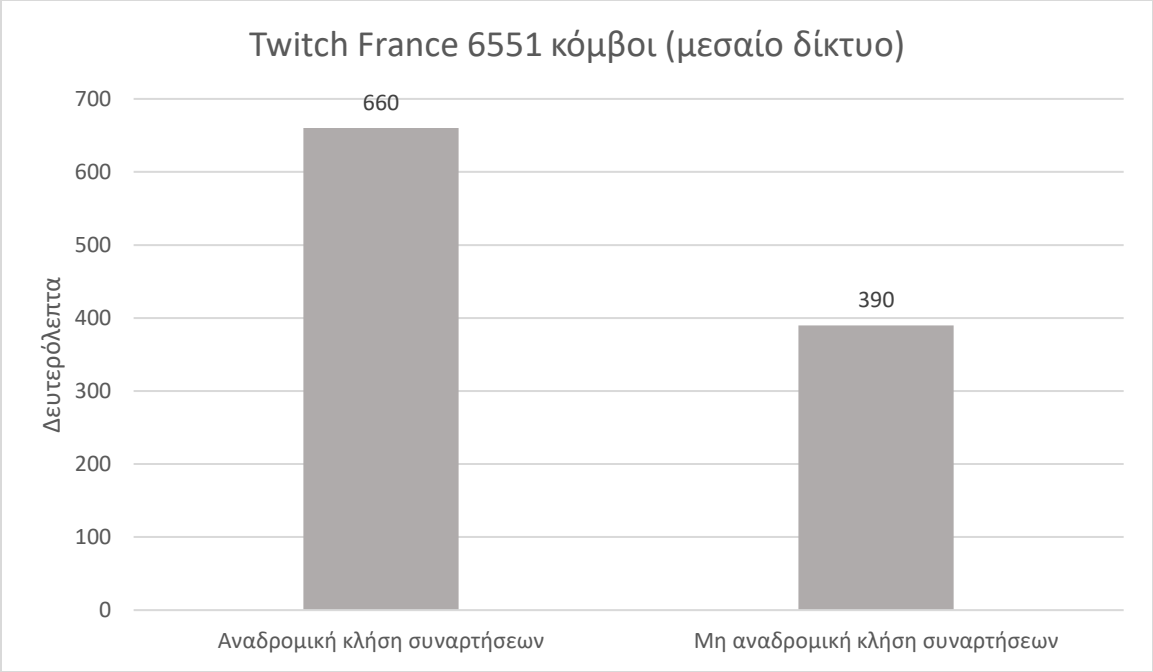
Επιλέχτηκαν τυχαία τρία δίκτυα, ένα από κάθε κατηγορία για να εξεταστεί το χρονικό όφελος από τις χρήσεις μη αναδρομικών συναρτήσεων (Πίνακας 1, διαγράμματα εικόνων 11, 12, 13). Στον Πίνακα 1 φαίνονται οι χρόνοι ολοκλήρωσης του κώδικα σε δευτερόλεπτα με αναδρομή και χωρίς αναδρομή και στα διαγράμματα των εικόνων 11, 12, 13 οι συγκρίσεις τους για κάθε ένα από τα τρία δίκτυα που επιλέχτηκαν.

Πίνακας 1. Αναδρομική και μη κλήση συναρτήσεων τριών δικτύων

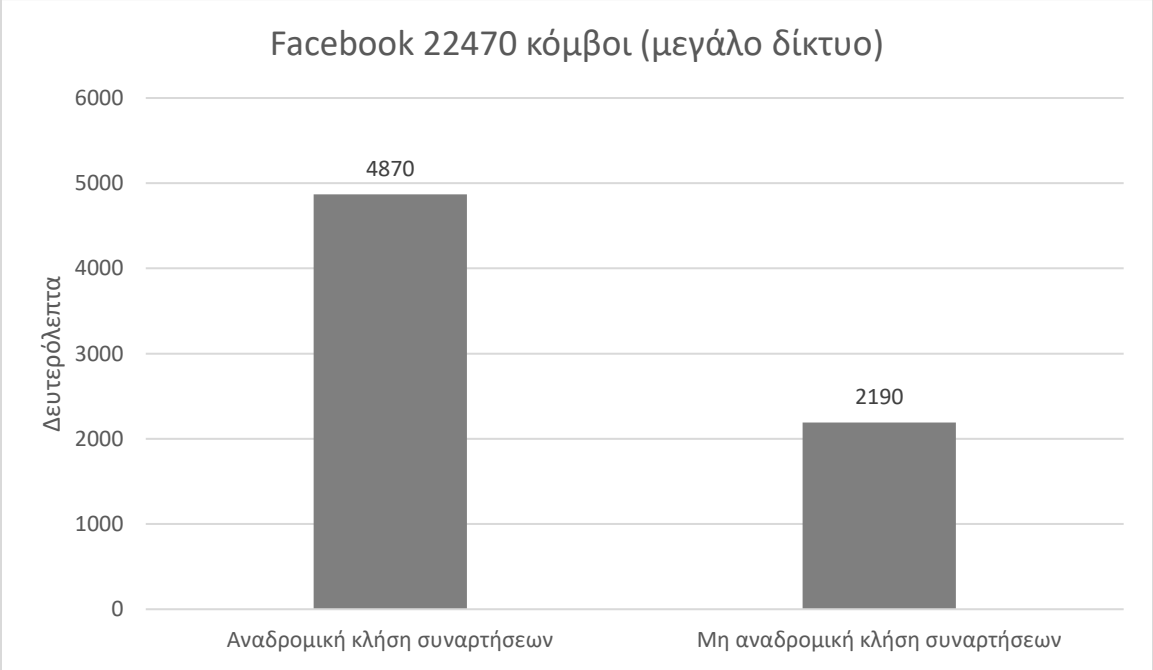
	Facebook 333 κόμβοι	Twitch France 6551 κόμβοι	Facebook 22470 κόμβοι
Αναδρομική κλήση συναρτήσεων	0,45s	660s	3870s
Μη αναδρομική κλήση συναρτήσεων	0,35s	390s	2190s



Εικόνα 12. Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλήσης για δίκτυο 333 κόμβων



Εικόνα 13. Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλήσης για δίκτυο 6.551 κόμβων



Εικόνα 14 - Διάγραμμα χρόνου ολοκλήρωσης αναδρομικής και μη αναδρομικής κλήσης για δίκτυο 22.470 κόμβων

Η βελτίωση του χρόνου εκτέλεσης είναι ορατή και στα τρία δίκτυα και αντιστρόφως ανάλογη των κόμβων. Δηλαδή, όσο μεγαλώνει το δίκτυο, από πλευρά κόμβων, η εκτέλεση χωρίς αναδρομή ολοκληρώνεται σε πολύ λιγότερο χρόνο σε σχέση με την αναδρομική κλήση.

4.2 ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ

Για την πειραματική διαδικασία χρησιμοποιήθηκαν 14 κοινωνικά δίκτυα (Πίνακας 4.2, κατάλογος δικτύων) (Krevl, 2014), τα οποία χωρίστηκαν σε τρεις κατηγορίες ανάλογα με των αριθμό των κόμβων (χρηστών). Μικρά δίκτυα (0-1.000 κόμβοι), μεσαία δίκτυα (1.000-8.000 κόμβοι), μεγάλα δίκτυα (8.000-40.000 κόμβοι). Για κάθε δίκτυο του δείγματος εργαστήκαμε με τον ίδιο τρόπο, αποφεύγοντας τη χρήση αναδρομικής κλήσης συναρτήσεων.

Πίνακας 2. Κατάλογος δικτύων

Προέλευση	Κόμβοι	Πυκνότητα
Twitter	126	0,06
Facebook	150	0,12
Facebook	224	0,04
Twitter	231	0,08
Facebook	333	0,04
Facebook	1035	0,05
Twitch Portugal	1914	0,01
Facebook	4039	0,01
Twitch Spain	4650	0,005
Twitch France	6551	0,005
Twitch Deutschland	9499	0,03
Facebook	22.470	0,07
GitHub	37.700	0.0004

4.2.1 Δημιουργία Γράφου με Οριοθετημένες Κοινότητες

Τα σύνολα δεδομένων που χρησιμοποιήθηκαν από την αναφορά (Krevl, 2014) αποτελούνται από δύο αρχεία. Στο ένα είναι αποθηκευμένες οι ακμές του γράφου και στο άλλο οι κοινότητες που σχηματίζουν οι κόμβοι αυτού. Το αρχείο των ακμών είναι της μορφής text (.txt), και σε κάθε γραμμή του βρίσκεται μια ακμή του δικτύου. Π.χ. 12 --> 17. Το αρχείο των κοινοτήτων είναι διαφόρων μορφών αναλόγως του δικτύου (json, csv, txt), και κάθε γραμμή αυτού μας δίνει τον κωδικό της κοινότητας και τους κόμβους από τους οποίους αποτελείται. Π.χ. community: 15 members 1, 5, 7, 13, 190. Έτσι με χρήση βιβλιοθήκης $NetworkX$ που αναφέρεται στην προηγουμένη ενότητα, προσομοιώνεται το δίκτυο με οριοθετημένες κοινότητες.

4.2.2 Υπολογισμός Βαρών Ακμών

Κάθε κόμβος των δικτύων καταχωρείται στο σύστημα με μια δεκαψήφια δυαδική τιμή. Κάθε ψηφίο της οποίας αντιστοιχεί σε ένα χαρακτηριστικό το οποίο είτε ικανοποιεί (1) είτε όχι (0) ο χρήστης. Στην (Krevl, 2014), από την οποία αντλήθηκαν τα δείγματα που χρησιμοποιήθηκαν, μπορεί κάποιος να βρει διάφορες και ποικίλες τέτοιες κατηγορίες χαρακτηριστικών, για τους κόμβους, που αφορούν την

εκπαίδευση, την ηλικία, το επάγγελμα, τα χόμπι τους κτλ. Τα χαρακτηριστικά και οι κόμβοι είναι κωδικοποιημένα για να διατηρείται η ανωνυμία. Εκτός αυτού, η εξαψήφια δυαδική τιμή για κάθε χρήστη τροφοδοτείται από πηγή τυχαίων αριθμών 0 και 1, για διευκόλυνση.

Π.χ. Node ID: 123 binary Value: 1000101010

Πίνακας 3. Δεκαψήφια δυαδική τιμή του κόμβου 123

Χαρακτηριστικό εκπαίδευσης	ID: 5	1
Χαρακτηριστικό κατοικίας	ID: 4	0
Χαρακτηριστικό φύλου	ID: 2	0
Χαρακτηριστικό επαγγέλματος	ID: 8	0
Χαρακτηριστικό καταγωγής	ID: 3	1
Χαρακτηριστικό γλώσσας	ID: 5	0
Χαρακτηριστικό χόμπι	ID: 9	1
Χαρακτηριστικό μουσικής	ID: 11	0
Χαρακτηριστικό διατροφής	ID: 10	1
Χαρακτηριστικό ενδιαφερόντων	ID: 7	0

Για να υπολογίσουμε το βάρος της ακμής που συνδέει δυο κόμβους I, J του γράφου υπολογίζουμε πόσες φορές εμφανίζεται το (1) στο διάνυσμα W που προκύπτει από τον υπολογισμό του bitwise Exclusive-OR (XOR) μεταξύ των δύο δεκαψήφιων τιμών των κόμβων I, J και το διαιρούμε με το πλήθος των χαρακτηριστικών που λήφθηκαν υπόψη, στην περίπτωσή μας 10.

Εξίσωση 4.1 - Διάνυσμα W bitwise Exclusive-OR (XOR)

$$W = I \oplus J \tag{4.1}$$

Εξίσωση 4.2 - Υπολογισμός βάρους ακμής

$$\beta\acute{\alpha}\rho o\varsigma\ \alpha\kappa\mu\acute{\eta}\varsigma = s/f \tag{4.2}$$

Όπου s: πλήθος των 1 στο διάνυσμα W, f: πλήθος χαρακτηριστικών.

Με αυτόν τον τρόπο οι ακμές των δικτύων των γράφων αποκτούν βάρη οι τιμές των οποίων κυμαίνονται από 0 έως 1 (Stavros Souravlas, 2019).

4.2.3 Κοινότητα Στόχος

Για κάθε δίκτυο με βάση το αρχείο των κοινοτήτων του επιλέγεται μία κοινότητα η οποία θα χρησιμοποιηθεί ως στόχος και ο αλγόριθμος θα εξετάσει πόσοι κόμβοι του γράφου (εκτός των μελών της κοινότητας στόχου) ανήκουν σε αυτήν σύμφωνα με το μοντέλο που παρουσιάστηκε. Οι κοινότητες που

επιλέχτηκαν ως στόχοι ήταν αυτές με τους περισσότερους χρήστες ώστε τα αποτελέσματα να είναι ορατά.

4.3 ΠΕΡΙΠΤΩΣΕΙΣ ΜΕΛΕΤΗΣ

Για την πειραματική διαδικασία ο αλγόριθμος (Κεφάλαιο 3) εκτελέστηκε 7 φορές για κάθε δίκτυο, (γράφο) έχοντας την ίδια κοινότητα-στόχο κάθε φορά. Σε κάθε μια από τις εκτελέσεις όλοι οι χρήστες του δικτύου είχαν τουλάχιστον επιπλέον ένα χαρακτηριστικό μεταξύ τους κοινό. Κάτι το οποίο επιτεύχθηκε, κρατώντας ίσα με (1) στη δεκαψήφια δυαδική τιμή, όσα χαρακτηριστικά επιλέγονται κάθε φορά να έχουν κοινά οι χρήστες μεταξύ τους. Με τον τρόπο αυτό παρατηρούμε πώς αυξάνονται τα αποτελέσματα (χρήστες του συνολικού δικτύου που ανήκουν στην κοινότητα-στόχο) καθώς αυξάνονται τα κοινά χαρακτηριστικά μεταξύ των κόμβων του γράφου. Οι εκτελέσεις για κάθε δίκτυο έλαβαν χώρα ως εξής.

4.3.1 Εκτελέσεις

1η εκτέλεση, η δεκαψήφια δυαδική τιμή όλων των κόμβων έχει 3 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 7 είναι τυχαία. Έτσι όλοι οι χρήστες του δικτύου έχουν τουλάχιστον 3 κοινά χαρακτηριστικά.

2η εκτέλεση, η δεκαψήφια δυαδική τιμή των κόμβων έχει 4 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 6 είναι τυχαία. Με αυτόν τον τρόπο όλοι οι χρήστες έχουν τουλάχιστον 4 κοινά χαρακτηριστικά.

3η εκτέλεση, η δεκαψήφια δυαδική τιμή των κόμβων έχει 5 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 5 είναι τυχαία. Έτσι όλοι οι κόμβοι του δικτύου έχουν τουλάχιστον 5 κοινά χαρακτηριστικά,

4η εκτέλεση, η δεκαψήφια δυαδική τιμή των κόμβων έχει 6 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 4 είναι τυχαία. Έτσι οι χρήστες του δικτύου έχουν τουλάχιστον 6 κοινά χαρακτηριστικά.

5η εκτέλεση, ο δεκαψήφια δυαδική τιμή των κόμβων έχει 7 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 3 είναι τυχαία. Με αυτόν τον τρόπο όλοι οι χρήστες έχουν τουλάχιστον 7 κοινά χαρακτηριστικά.

6η εκτέλεση, η δεκαψήφια δυαδική τιμή όλων των κόμβων έχει 8 ίδια ψηφία στις ίδιες θέσεις και τα υπόλοιπα 7 είναι τυχαία. Έτσι όλοι οι χρήστες του δικτύου έχουν τουλάχιστον 3 κοινά χαρακτηριστικά.

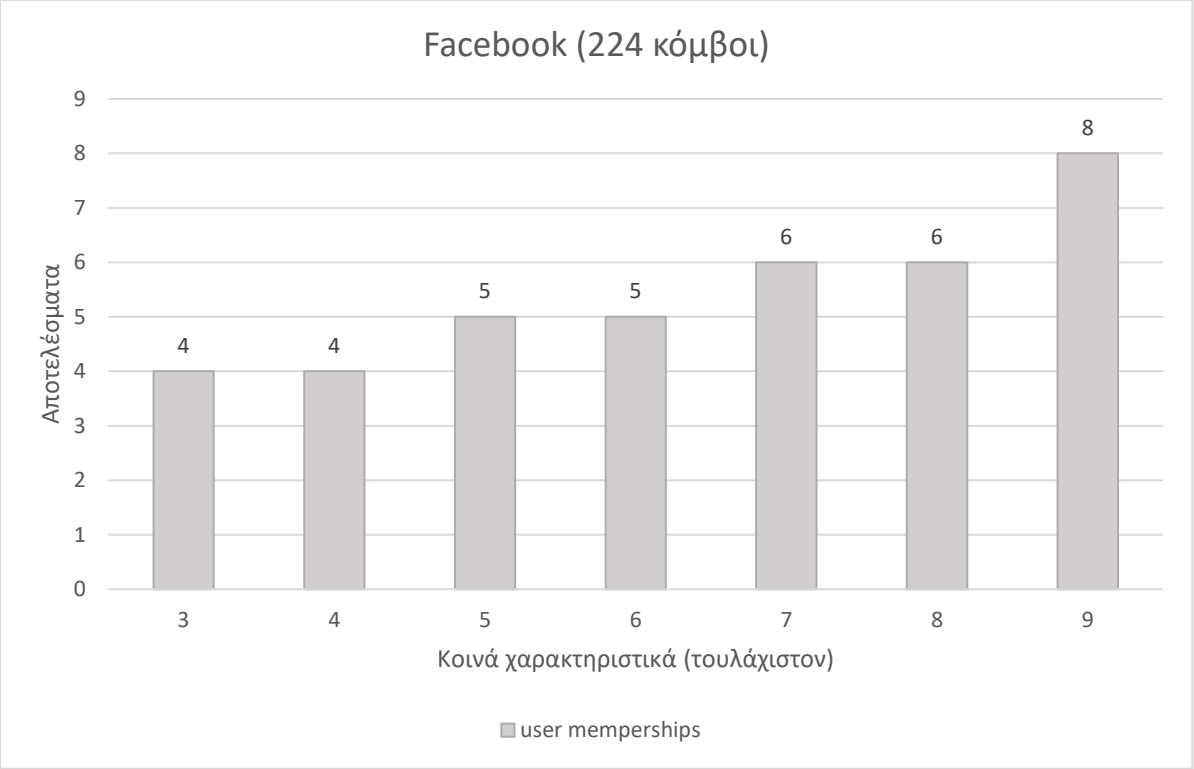
7η εκτέλεση, η δεκαψήφια δυαδική τιμή των κόμβων έχει 9 ίδια ψηφία στις ίδιες θέσεις και το τελευταίο ψηφίο είναι τυχαίο για όλους τους χρήστες. Έτσι όλοι οι κόμβοι του δικτύου έχουν τουλάχιστον 9 κοινά χαρακτηριστικά.

Πίνακας 4. Διαδοχικές εκτελέσεις

Εκτέλεση	Κοινά χαρακτηριστικά	δεκαψήφια δυαδική τιμή
$1^{η}$	3 (τουλάχιστον)	3 ίδια ψηφία και 7 τυχαία
$2^{η}$	4 (τουλάχιστον)	4 ίδια ψηφία και 6 τυχαία
$3^{η}$	5 (τουλάχιστον)	5 ίδια ψηφία και 5 τυχαία
$4^{η}$	6 (τουλάχιστον)	6 ίδια ψηφία και 4 τυχαία
$5^{η}$	7 (τουλάχιστον)	7 ίδια ψηφία και 3 τυχαία
$6^{η}$	8 (τουλάχιστον)	8 ίδια ψηφία και 2 τυχαία
$7^{η}$	9 (τουλάχιστον)	9 ίδια ψηφία και 1 τυχαίο

4.3.2 Αποτελέσματα

Αναζητώντας λοιπόν πόσοι χρήστες του συνολικού δικτύου (πλην των χρηστών που ήδη ανήκουν στην κοινότητα-στόχο) είναι μέλη αυτής της κοινότητας, καθώς μοιράζονται κάθε φορά τουλάχιστον ένα παραπάνω κοινό χαρακτηριστικό, λάβαμε τα αποτελέσματα που εμφανίζονται στα διαγράμματα των Εικόνων 14, 15, 16. Παρατηρεί κανείς ότι όσα περισσότερα ίδια χαρακτηριστικά έχουν οι κόμβοι του γράφου τόσο αυξάνονται τα πιθανά μέλη της κοινότητας-στόχου. Κάτι το οποίο διακρίνεται και στις τρεις κατηγορίες δεδομένων (Μικρά δίκτυα 0-1.000 κόμβοι, μεσαία 1.000-8.000, μεγάλα 8.000-40.000). Εμφανέστερες είναι οι μεταβολές στα μικρότερα δίκτυα, με την αύξηση των κόμβων, τα αποτελέσματα μεταβάλλονται πιο ήπια.



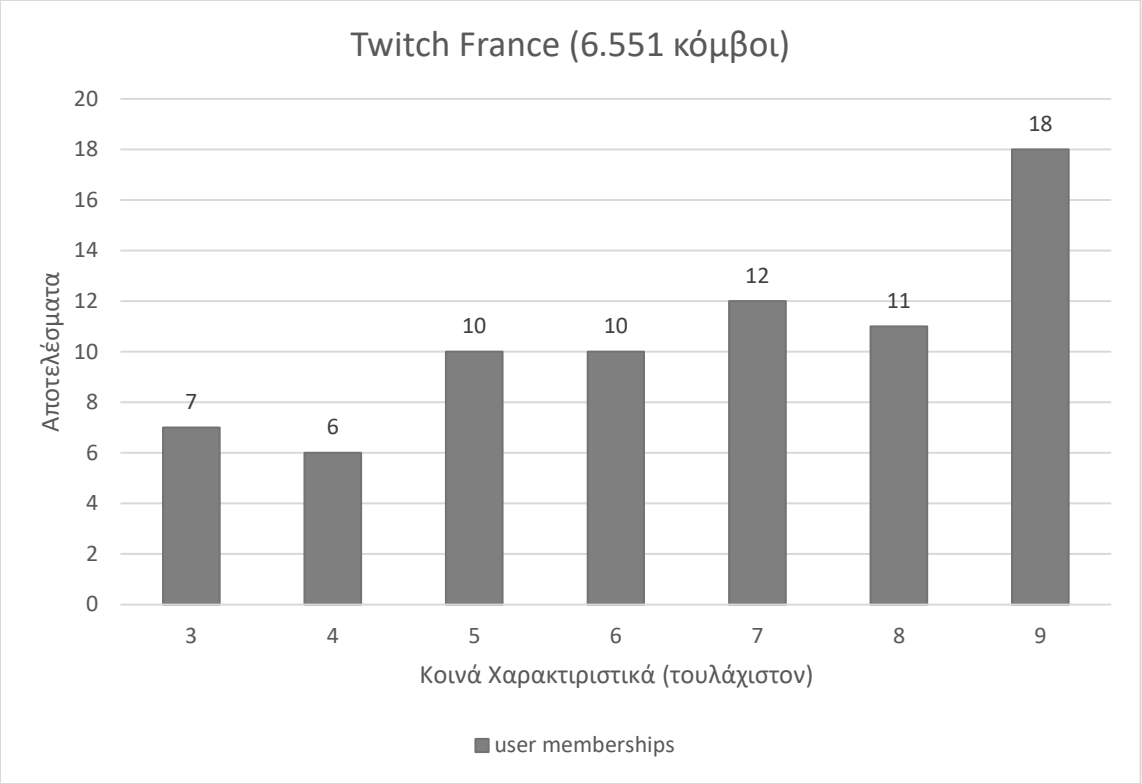
Εικόνα 15. Διάγραμμα αριθμού αποτελεσμάτων ανά εκτέλεση για δίκτυο 224 κόμβων

Πίνακας 5. Χαρακτηριστικά δικτύου 224 κόμβων

Προέλευση	Facebook
Κόμβοι	224
Πυκνότητα	0,12

Πίνακας 6. Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (224 κόμβων)

Κοινά χαρακτηριστικά (τουλάχιστον)	Νέα μέλη
3	4
4	4
5	5
6	5
7	6
8	6
9	8



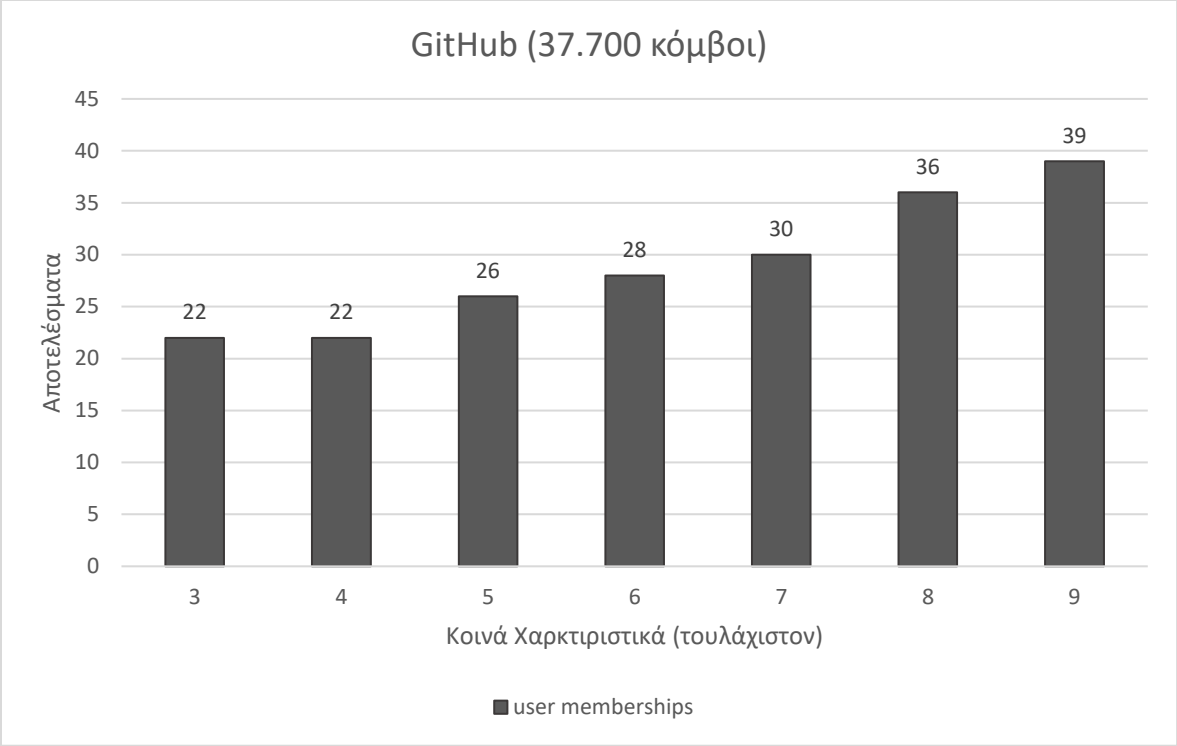
Εικόνα 16. Διάγραμμα αριθμού αποτελεσμάτων ανά εκτέλεση για δίκτυο 6.551 κόμβων

Πίνακας 7. Χαρακτηριστικά δικτύου 6.551 κόμβων

Προέλευση	Twitch France
Κόμβοι	6.551
Πυκνότητα	0,0053

Πίνακας 8. Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (6.551 κόμβων)

Κοινά χαρακτηριστικά (τουλάχιστον)	Νέα μέλη
3	7
4	6
5	10
6	10
7	12
8	11
9	18



Εικόνα 17. Διάγραμμα αριθμού αποτελεσμάτων ανά εκτέλεση για κοινωνικό δίκτυο 37.700 κόμβων

Πίνακας 9. Χαρακτηριστικά δικτύου 6.551 κόμβων

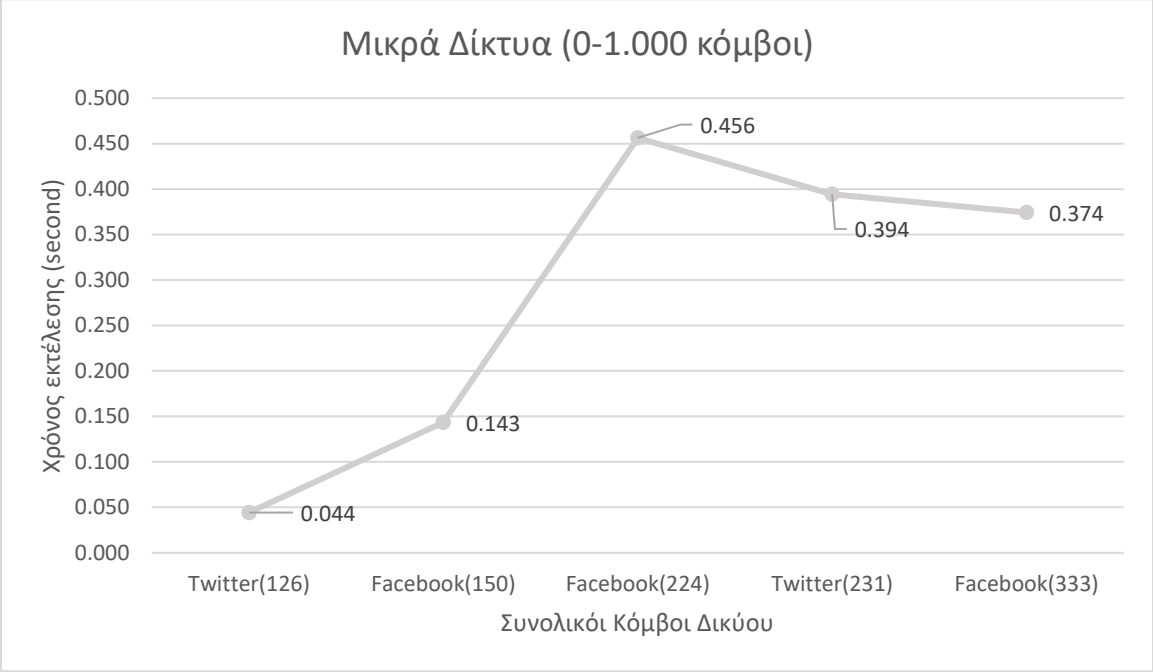
Προέλευση	GitHub
Κόμβοι	37700
Πυκνότητα	0.0004

Πίνακας 10. Κατάλογος αναλογίας αποτελεσμάτων-κοινών χαρακτηριστικών δικτύου (6.551 κόμβων)

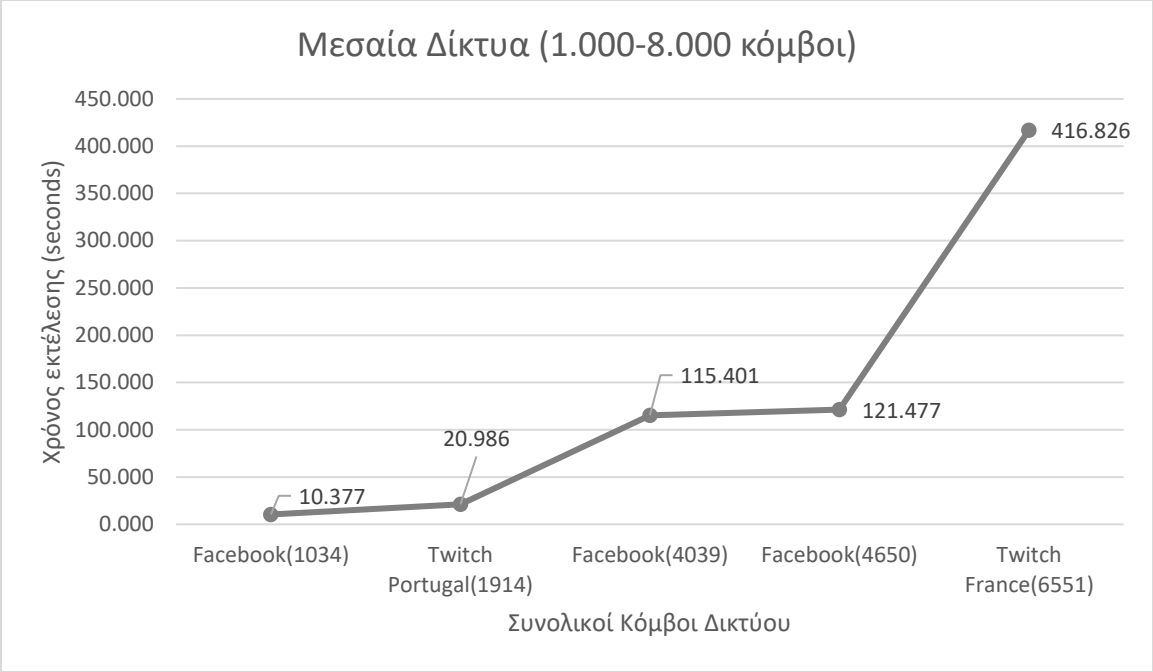
Κοινά χαρακτηριστικά (τουλάχιστον)	Νέα μέλη
3	22
4	22
5	26
6	28
7	30
8	36
9	39

4.3.3 Χρόνος Εκτέλεσης

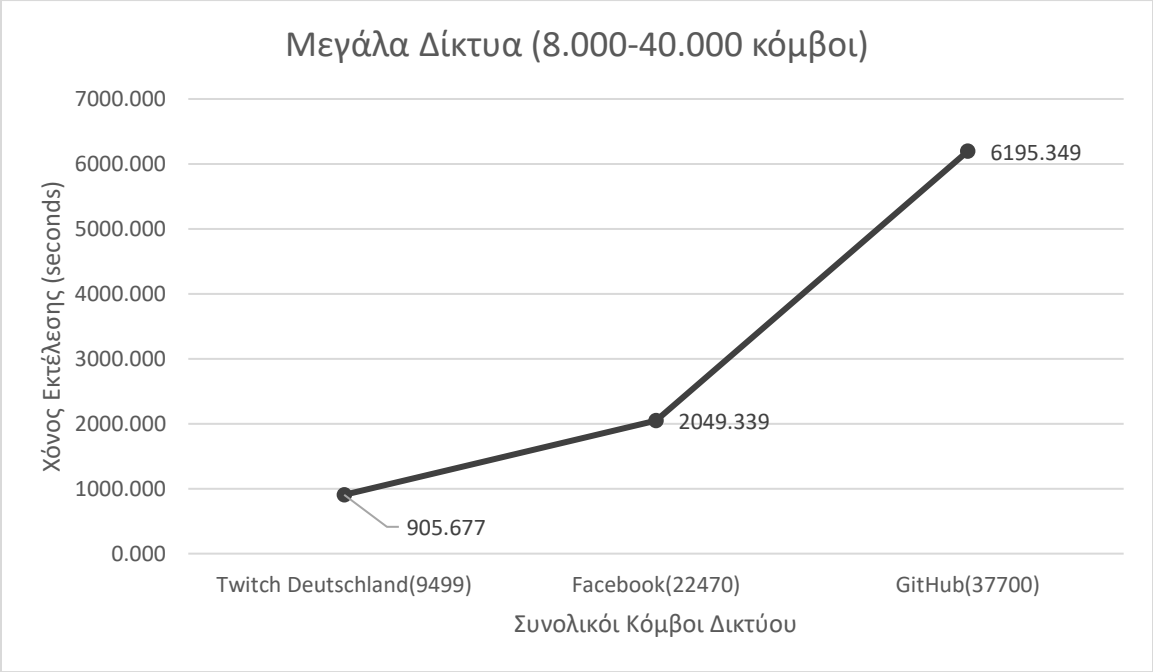
Η Python αποτελεί μια δυναμικού τύπου γλώσσα προγραμματισμού, υψηλού επιπέδου. Κάτι το οποίο την καθιστά πολύ αργή συγκριτικά με άλλες γλώσσες όπως η C, C++ και Java. Παρόλα αυτά αξίζει να σημειωθεί η ραγδαία αύξηση του χρόνου εκτέλεσης του αλγορίθμου, που παρατηρείται με την αύξηση των κόμβων στα δίκτυα (διαγράμματα 1,2,3). Ο χρόνος εκτέλεσης για τις τρεις κατηγορίες δικτύων κυμαίνεται στις εξής τιμές: μικρά δίκτυα (0,04-0,45 δευτερόλεπτα), μεσαία δίκτυα (10 δευτερόλεπτα–7 λεπτά), μεγάλα δίκτυα (15-103 λεπτά), διαγράμματα των Εικόνων 17, 18, 19.



Εικόνα 18. Διάγραμμα χρόνων εκτελέσεων μικρών δικτύων



Εικόνα 19. Διάγραμμα χρόνων εκτελέσεων μεσαίων δικτύων



Εικόνα 20. Διάγραμμα χρόνου εκτελέσεων μεγάλων δικτύων

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ, ΕΦΑΡΜΟΓΕΣ ΚΑΙ ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ

5.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα διπλωματική εργασία παρουσιάζει ένα μοντέλο ανίχνευσης νέων μελών για συγκεκριμένη κοινότητα χρηστών ενός κοινωνικού δικτύου με ήδη αρχικοποιημένες κοινότητες χρηστών. Το κοινωνικό δίκτυο αναπαρίσταται ως Γράφος όπου οι κόμβοι του αποτελούν τους χρήστες και οι ακμές τις συσχετίσεις μεταξύ αυτών. Από το κεφάλαιο της υπολογιστικής εμπειρίας συμπεραίνεται εύκολα ότι όσα περισσότερα κοινά χαρακτηριστικά έχουν οι χρήστες ενός δικτύου στο οποίο εκτελείται η αλγοριθμική διαδικασία που παρουσιάστηκε, τόσα περισσότερα νέα μέλη ενδέχεται να έχει η κοινότητα χρηστών που έχει οριστεί ως στόχος πριν από την εκτέλεση του κώδικα. Αξίζει να σημειωθεί ότι τα αποτελέσματα ήταν ελαφρός πιο ευνοϊκά για μικρότερα δίκτυα (μέχρι 8.000 κόμβοι) τόσο από άποψη εμβέλειας αποτελεσμάτων όσο και από θέμα χρόνου. Σημαντικό να αναφερθεί επίσης ότι τα αποτελέσματα, σε όλα τα δίκτυα, αντιπροσωπεύουν πειράματα με μεγάλες κοινότητες στόχους (20 κόμβοι και πάνω). Συνοψίζοντας, κατανοεί κανείς ότι η αποτελεσματική εκμετάλλευση του μοντέλου προϋποθέτει μεγάλες κοινότητες στόχους και η βελτιστοποίηση των αποτελεσμάτων από τη μεριά της, δίκτυα των οποίων οι χρήστες μοιράζονται όσο το δυνατόν περισσότερα κοινά χαρακτηριστικά.

5.1.1 Εφαρμογές στο Marketing

Το μάρκετινγκ των μέσων κοινωνικής δικτύωσης έχει γίνει πλέον απαραίτητο συστατικό του online marketing κάθε επιχείρησης και ελεύθερου επαγγελματία. Τα διαδικτυακά κοινωνικά δίκτυα ενδιαφέρουν το κλάδο του μάρκετινγκ για τέσσερις σημαντικούς λόγους:

1. Είναι εξαιρετικά δημοφιλή: 2,85 δισεκατομμύρια μέλη μόνο το Facebook για το 2021 τα οποία μάλιστα ξοδεύουν αρκετό προσωπικό χρόνο.
2. Έδωσαν ένα νέο εργαλείο επικοινωνίας για τους καταναλωτές και ένα νέο εργαλείο διαφήμισης για τις επιχειρήσεις.
3. Για τον τεράστιο αριθμό προσωπικών πληροφοριών που εθελούσια δημοσιεύουμε σε αυτά τα δίκτυα (Mansfield-Devine, 2008). Είναι χαρακτηριστικό το παράδειγμα που χρησιμοποιεί η Grabner-Krauter (2009) τονίζοντας ότι σε τυπικό προφίλ χρήστη των διαδικτυακά κοινωνικών δικτύων υπάρχουν πληροφορίες όπως η διεύθυνση κατοικίας, το όνομα του κατοικίδιού του, σε ποιο δημοτικό σχολείο ήταν και άλλες προσωπικές πληροφορίες οι οποίες πολύ συχνά χρησιμοποιούνται από τις τράπεζες ή άλλες διαδικτυακές υπηρεσίες σαν δικλείδα ασφαλείας σε περίπτωση απώλειας του κωδικού εισόδου.
4. Τέλος μέχρι πρόσφατα οι εταιρείες ήταν σε θέση να ελέγχουν τις διαθέσιμες πληροφορίες σχετικά με αυτές μέσα από ανακοινώσεις τύπου και οργανωμένο τμήμα δημόσιων σχέσεων. Σήμερα, ωστόσο, οι επιχειρήσεις έχουν όλο και περισσότερο υποβιβαστεί στο περιθώριο ως απλοί παρατηρητές, αφού ούτε τις γνώσεις ούτε την ευκαιρία – ή, μερικές φορές, ακόμα και το δικαίωμα – να τροποποιήσουν τα δημόσια σχόλια που υποβάλλονται από τους πελάτες τους (Kaplan & Haenlein, 2010). Η Wikipedia, για παράδειγμα, απαγορεύει ρητά τη συμμετοχή των

επιχειρήσεων στη συγγραφή λημμάτων χωρίς γνωστοποίηση της σύγκρουσης κινήτρων. Έτσι οι επιστήμονες του μάρκετινγκ δημιούργησαν ορισμένες νέες μεθόδους διαφήμισης και επικοινωνίας με τους καταναλωτές όπως το eWOM (electronic Word Of Mouth ή ηλεκτρονική δια στόματος διαφήμιση), το viral marketing (ή ιογενές μάρκετινγκ) και το direct marketing (ή άμεσο μάρκετινγκ).

Το μάρκετινγκ των μέσων κοινωνικής δικτύωσης απευθύνεται σε μεγάλες αλλά και μικρές επιχειρήσεις. Διάσημες εταιρείες της αγοράς χρησιμοποιούν τα μέσα κοινωνικής δικτύωσης για να προσελκύσουν νέους πελάτες καθώς και να διατηρήσουν τη φήμη και το όνομα τους, ενώ παράλληλα στοχεύουν ένα ευρύ κοινό από καταναλωτές. Στα πλαίσια μιας ολοκληρωμένης στρατηγικής για την παρουσία στα μέσα κοινωνικής δικτύωσης, είναι απαραίτητη, η σχεδίαση, η ανάπτυξη και η υλοποίηση των απαραίτητων τακτικών ενεργειών προκειμένου να υπάρξουν τα βέλτιστα οφέλη για το εμπορικό σήμα μιας επιχείρησης. Επιλέγουμε ποια είναι τα καταλληλότερα μέσα κοινωνικής δικτύωσης για το κάθε εμπορικό σήμα και καταστρώνεται στρατηγικά η καμπάνια στο καθένα από αυτά.

Μέσω της κοινωνικής δικτύωσης μια επιχείρηση, μπορεί να αλληλοεπιδρά με το κοινό της, που σημαίνει πως θέτει τις βάσεις για το χτίσιμο μακροχρόνιων σχέσεων με όλους τους μετόχους και στην ουσία, να πετύχει τη συμμετοχή. Ως γνωστό, στα μέσα κοινωνικής δικτύωσης έχουν παρουσία όχι μόνο το κοινό, αλλά και οι συνεργάτες, οι προμηθευτές, το προσωπικό της επιχείρησης και άλλοι που σχετίζονται με αυτήν. Επικοινωνώντας τακτικά με τα μέλη η επιχείρηση δεν έχει απλά μια παρουσία στα μέσα κοινωνικής δικτύωσης αλλά τα χρησιμοποιεί με ουσιαστικό τρόπο προσθέτοντας παράλληλα αξία. Σύμφωνα όμως με τον B. Borges στο βιβλίο του (Borges, 2009) υπάρχουν 4 στάδια αλληλεπίδρασης (interaction):

- Συμμετοχή (Engaging): Χρειάζεται συστηματική και τακτική χρήση της σελίδας, έτσι ώστε να αυξηθεί η αλληλεπίδραση με τους χρήστες της σελίδας. Το πότε και πόσο συχνά μέσα στην ημέρα ή εβδομάδα είναι διαφορετικό για τον καθένα.

- Να ακούει (Listening): Θετικά και αρνητικά σχόλια που γράφονται από τους χρήστες στις σελίδες των επιχειρήσεων μπορούν να αντιμετωπιστούν κατάλληλα, αλλά σίγουρα όχι με την αδιαφορία. Εκτός αυτού, απαντώντας η επιχείρηση δείχνει πως ενδιαφέρεται να χτίσει σχέσεις με τους «οπαδούς της».

- Αλληλεπίδραση (Interacting): Η αλληλεπίδραση στα μέσα κοινωνικής δικτύωσης έρχεται με φυσικό τρόπο. Για παράδειγμα, ένας σύνδεσμος που θα ανεβάσει κανείς, μια φωτογραφία, μια χρήσιμη ή ενδιαφέρουσα πληροφορία ή ακόμα και μια δημοσκόπηση είναι αρκετά για να αρχίσουν να κάνουν «μου αρέσει», σχόλιο ή κοινοποίηση τα μέλη. Η φωτογραφία επίσης, ενός νέου προϊόντος ή η περιγραφή μιας νέας υπηρεσίας έχει δώσει τη δυνατότητα να πληροφορηθεί περισσότερο το κοινό αλλά και να τους προτρέψει να τα δοκιμάσουν. Με την κατάλληλη αντίδραση μπορεί κανείς να δημιουργήσει συζητήσεις μαζί τους, οι οποίες θα διαδοθούν στους φίλους των μελών και κατ' επέκταση στους φίλους των φίλων των μελών (viral effect).

- Μέτρηση (Measuring): Όλα τα κοινωνικά δίκτυα είναι ένα επιπλέον μέσο που μπορεί να προσδώσει αξία σε μια επιχείρηση. Η αποτελεσματικότητά τους, λοιπόν, είναι επιτακτική ανάγκη να μετριέται. Η συγκεκριμένη μέτρηση, ωστόσο, έχει σημασία όταν έχουν τεθεί κάποιοι στόχοι, έτσι ώστε να γίνει μια εποικοδομητική αξιολόγηση. Κάποιος στόχος μπορεί να αφορά τον αριθμό των υποστηρικτών, άρα αυτό που ενδιαφέρει να είναι ο αριθμός αντίστοιχα των νέων «μου αρέσει» ανά χρήστη ή ανά περίοδο και συνολικά, ο αριθμός των «μου αρέσει». Άλλος στόχος μπορεί να είναι η αλληλεπίδραση, και επομένως να ενδιαφέρουν η ποσότητα και η ποιότητα των σχολίων, καθώς και η συναισθηματική χροιά (θετικό/αρνητικό).

5.1.2 Εφαρμογές Ανίχνευσης Κοινότητας

Στη ψηφιακή εποχή, όλοι είναι πλέον οικείοι με τα πληροφοριακά συστήματα. Στο σημερινό κόσμο, τα συστήματα πληροφοριών αντιπροσωπεύονται από πολύπλοκα δίκτυα τα οποία εμπεριέχουν τη δομή των κοινοτήτων.
Τα σύνθετα δίκτυα μοντελοποιούνται είτε στατικά είτε δυναμικά, παρόλα αυτά μπορεί να γίνει ανίχνευση κοινότητας και στις δύο περιπτώσεις. Το στατικό δίκτυο μπορεί να θεωρηθεί ένα απλώς παγωμένο δίκτυο για μια καθορισμένη χρονική στιγμή. Ωστόσο, κοινότητες στο δίκτυο μπορεί να αναπτυχθούν ή να συρρικνωθούν σε μέγεθος, ακόμη και νέες κοινότητες μπορεί να εμφανιστούν και αντίστοιχα ορισμένες από αυτές να εξαφανιστούν με το πέρασμα του χρόνου. Κατάσταση η οποία γίνεται αντιληπτή με τη δυναμική ανίχνευση κοινότητας. Το μοντέλο που παρουσιάστηκε στην παρούσα εργασία κατατάσσεται στη στατική μελέτη των δικτύων και η δυναμική ανίχνευση αποτελεί αντικείμενο μελλοντικής έρευνας. Καταλήγοντας η στατική ανίχνευση κοινότητας επικεντρώνεται στην αποκάλυψη της κοινωνικής δομής ενός δικτύου την παρούσα χρονική στιγμή ενώ η δυναμική είναι υπεύθυνη για τη μελέτη της εξέλιξης των κοινωνικών δομών με το πέρασμα του χρόνου.
Η κοινοτική ανίχνευση γίνεται από πολλούς ερευνητές από διαφορετικούς κλάδους μέχρι σήμερα. Ακολουθούν διάφορες πρακτικές εφαρμογές κατηγοριοποιώντας τις σύμφωνα με τους τομείς τους και όχι απλώς με χρονολογική σειρά.

Εγκληματολογία

Η ανίχνευση κοινοτήτων χρησιμοποιείται για την αναγνώριση λογαριασμών εγκληματιών σε κοινωνικά δίκτυα. Αυτοί οι λογαριασμοί μπορεί να αντιστοιχούν σε πραγματικό πρόσωπο είτε να αποτελούν λογαριασμούς ρομπότ (software bot). Οι συγγραφείς του "Constructing and analyzing criminal networks" (H. Sarvari, 2014) χρησιμοποιούν την ανίχνευση κοινότητας για την αποκάλυψη δικτύων εγκληματιών. Ο Pinheiro (Pinheiro, 2012) έκανε μια μελέτη για τον εντοπισμό της απάτης εκδηλώσεων σε δίκτυα τηλεπικοινωνιών χρησιμοποιώντας ανίχνευση κοινότητας που βοηθά στον προσδιορισμό των συμπεριφορών των πελατών και εξετάζει τα άτομα που ξεχωρίζουν γνωστά και ως outliers ως πιθανή απάτη. Ομοίως, ο Waskiewicz παρέχει μια μελέτη (Waskiewicz, 2012) για την ανίχνευση τρομοκρατικών δραστηριοτήτων σε ορισμένα διαδικτυακά κοινωνικά δίκτυα χρησιμοποιώντας την ανίχνευση κοινότητας. Επιπλέον, αρκετά έργα επικεντρώνονται συνήθως στην ανίχνευση λογαριασμών ρομπότ, οι οποίοι χρησιμοποιούνται από εισβολείς για πλαστοπροσωπία, για απάτες ταυτότητας και επιθέσεις botnet. Ένα τέτοιο έργο αποτελεί το (Şahin, 2017).

Δημόσια υγεία

Στον τομέα της υγείας, ο εντοπισμός κοινότητας χρησιμοποιείται κυρίως για την αποκάλυψη ομάδων επιρρεπών σε επιδημία. Ο Salathe και ο Jones στο (M. Salathé, 2010) δείχνουν το αντίκτυπο αυτής της εφαρμογής στον συγκεκριμένο τομέα. Εφαρμογές της κοινοτικής ανίχνευσης χρησιμοποιούνται για τον εντοπισμό όγκων. Ο Bechtel στο (J. J. Bechtel, 2005) προτείνει μια κοινοτική προσέγγιση για την ανίχνευση καρκίνου του πνεύμονα. Επιπλέον, η κοινοτική ανίχνευση χρησιμοποιείται για την εύρεση οργάνων. Τα (N. Haq, 2016) αποτελούν παραδείγματα για αυτήν την κατηγορία.

Πολιτική

Στην Πολιτική, ο εντοπισμός κοινότητας χρησιμοποιείται για παρατήρηση επιρροών πολιτικών ιδεολογιών ή μεμονωμένων πολιτικών σε κάποια κοινωνική ομάδα. Μελλοντικά, τέτοια συστήματα

μπορεί να είναι εξειδικευμένα για την παρακολούθηση της εξέλιξης επιρροών με την πάροδο του χρόνου. Αυτές οι επιρροές δημιουργούνται ή από πολιτικούς ή από astroturfer bots (ρομπότ που προσπαθούν να δημιουργήσουν μια ψεύτικη ιδέα για την υποστήριξη μια πολιτικής ιδέας η μίας καμπάνιας προϊόντων σε πραγματικούς λαούς). Εντοπισμός κοινότητας χρησιμοποιείται για την ανίχνευση αυτών των τύπων ρομπότ (Şahin, 2017).

Τμηματοποίηση πελατών, έξυπνη διαφήμιση και στοχευμένο μάρκετινγκ

Ο εντοπισμός κοινότητας χρησιμοποιείται άμεσα για την τμηματοποίηση των πελατών, έξυπνη διαφήμιση και στοχευμένο μάρκετινγκ και από εταιρείες. Εταιρείες μπορεί να παρέχουν καλύτερη λύση εξυπηρέτησης εάν γνωρίζουν στενά τις ομάδες πελατών. Στη συνέχεια, μπορούν να επικεντρώσουν την διαφήμιση και το μάρκετινγκ για ανίχνευση σε συγκεκριμένες ομάδες πελατών (M. J. Mosadegh, 2011)

Recommendation systems

Τα συστήματα προτάσεων αποτελούν άλλη μια σημαντική κατηγορία υπηρεσιών που χρησιμοποιείται καθημερινά για την απόκτηση προϊόντων ή υπηρεσιών. Κάνοντας κράτηση από έναν ιστότοπο, παρακολουθώντας ένα βίντεο ή ακούγοντας μουσική σε ένα ιστότοπο κοινωνικών μέσων κλπ. επιδιώκουν να προτείνουν κάτι που πιθανώς θα θέλατε να γνωρίσετε. Το έργο ανίχνευσης της κοινότητας μοιάζει με το διαχωρισμό των ανθρώπων με ομοειδή χαρακτηριστικά. Αν και υπάρχουν πολλά έργα στην βιβλιογραφία που χρησιμοποιούν ανίχνευση κοινότητας γι' αυτόν τον σκοπό δίνονται μόνον μερικά παραδείγματα μελετών στο (S. B. Abdrabbah, 2014)

Ανάλυση κοινωνικών δικτύων

Η κοινοτική ανίχνευση μπορεί να ένα καλό μέσο κατανόησης λειτουργίας κοινοτήτων σε επίπεδο δικτύωσης που συσχετίζονται όμως με πραγματικές σχέσεις. Για παράδειγμα, η ανίχνευση κοινότητας σε διαδικτυακά κοινωνικά δίκτυα όπως το Facebook, το Twitter, το LinkedIn κ.λπ. είναι καλά παραδείγματα γι' αυτήν την περιοχή εφαρμογών.

Πρόβλεψη συνδέσμου

Η πρόβλεψη συνδέσμου αξιολογεί την πιθανότητα μελλοντικών συνδέσμων μεταξύ μελών ενός δικτύου και χρησιμοποιείται για τον καθορισμό ψεύτικων συνδέσμων, συνδέσμων που λείπουν ή μελλοντικών συνδέσμων. Η υποκείμενη δομή του δικτύου βρίσκεται μέσω ενός αλγόριθμου ανίχνευσης κοινότητας. Ο Valverde Rebaza και ο Lopes (J. C. Valverde-Rebaza, 2012) προτείνουν μια προσέγγιση βασισμένη σε εντοπισμό κοινότητας για πρόβλεψη συνδέσμου. Ομοίως, οι Soundarajan και Hopcroft (S. Soundarajan, 2012,) δείχνουν αυτήν την κοινότητα χρησιμοποιώντας πληροφορίες που λαμβάνονται με μεθόδους ανίχνευσης της κοινότητας, και έχουν αυξήσει την ακρίβεια της πρόβλεψης συνδέσμου.

Πρόβλεψη εξέλιξης της κοινότητας

Η πρόβλεψη της κοινοτικής εξέλιξης αφορά την πρόβλεψη της μελλοντικής μορφής μιας κοινότητας δεδομένου του παρελθόντος και της παρούσας μορφής της σε όρους κοινοτικών εκδηλώσεων όπως η ανάπτυξη, η συρρίκνωση, η συγχώνευση, ο σχηματισμός, η επίλυση κ.λπ. Είναι ένα από τα θέματα στον τομέα της ανάλυσης της κοινότητας. Φυσικά, υπάρχουν κάποια έργα στη λογοτεχνία. Για να επιτευχθεί όμως πρόβλεψη εξέλιξης της κοινότητας, απαιτείται δυναμική ανίχνευση κοινότητας. Από τις αναφερόμενες περιοχές εφαρμογών η κοινοτική ανίχνευση έχει ένα ευρύ φάσμα εφαρμογών σε διάφορους τομείς. Μερικοί από αυτούς χρησιμοποιούν στατική ανίχνευση κοινότητας, άλλοι δυναμική ανίχνευση κοινότητας ή και τα δύο. Εν κατακλείδι η παρούσα μελέτη αποτελεί σημαντικό εργαλείο διαδικτυακής ανάπτυξης επιχειρήσεων καθώς, εντάσσεται στις τακτικές της κατανόησης της συμπεριφοράς των καταναλωτών και της ανακάλυψης πόρων, εργαλείων και πληροφοριών για

επιτυχημένη πλοήγηση στη δυναμική αγορά του Διαδικτύου. Έτσι σε ένα κόσμο που αλλάζει συνεχώς λόγω της τεχνολογίας, το ψηφιακό περιβάλλον έχει αναδειχθεί ως το πιο ισχυρό πεδίο παρουσίας και εδραίωσης των brands

5.2 ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Ορισμένες προτάσεις για την περαιτέρω εξέλιξη της παρούσας διπλωματικής εργασίας είναι οι εξής:

- Μετατροπή του κώδικα σε C. Όπως αναφέρθηκε προηγουμένως η γλώσσα προγραμματισμού στην οποία προσομοιώθηκε το μοντέλο (Python) είναι πιο αργή σε σχέση με άλλες και συνήθως χρησιμοποιείται ως glue language. Γι' αυτό, η βελτίωση του χρόνου εκτέλεσης και η εισαγωγή αυτής της μελέτης στην παραγωγή προϋποθέτει τη μετατροπή του κώδικα σε C, γλώσσα από την οποία δημιουργήθηκε η Python, κάτι το οποίο καθιστά αυτή την μετατροπή απόλυτα εφικτή.
- Γενικά, η ιδέα της μετατροπής ενός δικτύου σε μια δομή δένδρου είναι ενδιαφέρουσα, ωστόσο, αυτή η εφαρμογή θα πρέπει να εφαρμοστεί προσεκτικά, καθώς μπορεί να είναι πολύ δαπανηρή όσον αφορά το κόστος υπολογισμού, ειδικά κατά την επεξεργασία δικτύων εκατομμυρίων κόμβων ή ακμών. Μια λύση σε αυτό το πρόβλημα θα ήταν ο προσεκτικός παραλληλισμός των διάφορων διεργασιών που απαιτούνται.
- Δυναμική ανίχνευση κοινότητας, μελέτη εξέλιξης των δικτύων με το πέρασμα του χρόνου. Αφορά την προσαρμογή της αλγοριθμικής διαδικασίας ώστε να λαμβάνει υπόψη τους νέους χρήστες που εισέρχονται στο δίκτυο ή τη διαγραφή κόμβων, αντίστοιχα.

ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

1. main.py

```
import networkx as nx
import random
import time
import tree_traversals
import sys
# import json
sys.setrecursionlimit(10**6)


def sort_neighbors(d):
    new_dict = {}

    for e in d[1]:
        sub_dict = d[1][e]
        for d2 in sub_dict:
            # print(sub_dict[d2])
            new_dict.update({e: sub_dict[d2]})

    new_dict = dict(sorted(new_dict.items(), key=lambda kv: kv[1]))
    return d[0], new_dict


def count_set_bits(b):
    count = 0
    while b:
        count += b & 1
        b >>= 1
    return count


def str_bitwise_xor(array, t):
    result = ""
    max_len = -1
    for i in range(t):
        max_len = max(max_len, len(array[i]))
        array[i] = arr[i][::-1]

    for i in range(t):
        s = ""
        for j in range(max_len - len(arr[i])):
            s += "0"

        arr[i] = arr[i] + s

    for i in range(max_len):
        pres_bit = 0

        for j in range(t):
            pres_bit = pres_bit ^ (ord(arr[j][i]) - ord('0'))

        result += chr(pres_bit + ord('0'))
```

```
    result = result[::-1]
    return result


def rand_key(p):
    p = p - 3

    key1 = ""
    for a in range(p):
        temp = str(random.randint(0, 1))
        key1 += temp
    return '111'+key1


def ncd(t):
    # network connectivity degree for every node of the network
    count = 0
    s = 0.0

    for user in t[1]:
        s += t[1][user]
        count += 1

    return s/count


def compute_acc(circle_, n_hoods):
    # returns (community id, number of members, average community
connectivity)
    id_ = circle_[0]
    # circle_.pop(0)
    s = 0.0
    count = 1

    for j in range(1, len(circle_)):

        for n_hood in n_hoods:

            if circle_[j] == n_hood[1][0]:
                s += n_hood[0]
                count += 1

    if count != 0:
        return id_, count, s/count


def dia(circle_, graph):

    circle_.pop(0)
    q = nx.Graph()

    for o in circle_:
        q.add_node(o)

    for (j, p, w) in graph.edges.data('weight'):
```

```
            if j in q.nodes:
                if p in q.nodes:
                    q.add_weighted_edges_from([(j, p, w)])

        if len(circle_) == 0:
            pass
        else:
            if nx.is_connected(q):
                return nx.diameter(q)+5
            else:
                return 3


def declare_membership(path_sim_t, diam, acc):
    # path_sim_t = (path_strength, updates)
    # diam = diameter
    # acc = average community connectivity

    if path_sim_t[1] <= diam+5:
        if path_sim_t[0] >= acc-1:
            return 'the node can be considered a member of the target
community'
        else:
            return


if __name__ == '__main__':
    start = time.perf_counter()
    with open('C:\\Users\\giorgos\\Desktop\\0.edges', 'r') as fh:
        # new_file = fh.read()
        # new_file = new_file.replace(',', ' ')
        # print(new_file)
        g = nx.read_edgelist(fh, delimiter=' ', create_using=nx.Graph(),
nodetype=str)

    # g =
nx.read_edgelist('C:\\Users\\giorgos\\Desktop\\twitter_combined.txt',
create_using=nx.Graph(), nodetype=str)
    # print(nx.info(g))
    print('Origin of the graph: Facebook')
    print('Number of nodes:', g.number_of_nodes())
    print('Density of the graph:', format(nx.density(g), '.4f'))
    print('Features in common: 3 (at least)')

    circles = []
    binary_values = {}
    features = 10

    with open('C:\\Users\\giorgos\\Desktop\\0.circles', 'r') as f:
        for line in f:
            k = line.split()
            circles.append(k)
            c = f.readline().split('\t')# Driver code
            # print(c)
            c[-1] = c[-1].strip('\n')
            circles.append(c)
```

```
    # music communities
    '''music1 = ['Folk']
    music2 = ['Techno/House']
    with open('C:\\Users\\giorgos\\Desktop\\HU_genres.json', 'r') as f:
        data = json.load(f)

    for key, value in data.items():
        # print(key, '-->', value)
        for kind in value:
            if kind == 'Folk':
                music1.append(key)
            elif kind == 'Techno/House':
                music2.append(key)
    circles.append(music1)
    circles.append(music2)'''

    w = 0.66
    for i in g.nodes:
        binary_values[i] = rand_key(features)
    # print(binary_values)

    for (u, v) in g.edges():
        arr = [binary_values[u], binary_values[v]]
        XOR = str_bitwise_xor(arr, 2)
        number_of_1s = count_set_bits(int(XOR, 2))
        weigh = number_of_1s/features
        g.add_weighted_edges_from([(u, v, weigh)])
        # print(f"({u}, {v}, {weigh:.3})")

    neighborhoods = []

    for n in g.adjacency():
        # print(sort_neighbors(n))
        # print(ncd(sort_neighbors(n)))
        element = (ncd(sort_neighbors(n)), sort_neighbors(n))
        # print(element)
        neighborhoods.append(element)

    diameters = {}
    # diameters = {id: diameter, ...}
    comm = []
    # comm = [(community id, number of members, average community
connectivity), ...()]
    for circle in circles:
        comm.append(compute_acc(circle, neighborhoods))
        diameters.update({circle[0]: dia(circle, g)})

    for _ in comm:
        pass
        # print(_)

    for _ in diameters.items():
        pass
        # print(_)

    target_com = circles[11]
    d = diameters['circle11']
```

```
    a = comm[11][2]

    trees = []
    for node in neighborhoods:
        r = tree_traversals.build_tree(node[1], w)
        trees.append(r)

    for tree in trees:
        tree_traversals.build_communities(tree, trees)

    stronger_paths = []
    for tree in trees:
        stronger_paths.append(tree_traversals.path_sim(tree, target_com))

    s = 0
    for path in stronger_paths:
        if path is not None:
            if declare_membership(path, d, a) is not None:
                s = s + 1
            # print(path)
            # print(declare_membership(path, d, a))

    print('Nodes that can be considered members of the target community:', s)

    finish = time.perf_counter()
    print(f'Finished in {round(finish - start, 2)} seconds')
    # end
```

2. tree traversals.py

```
class TreeNode:
    def __init__(self, node):
        self.nodeID = node
        self.right_link_value = None
        self.right_link_object = None
        self.right_thread = None
        self.left_thread = None
        self.left_link = None

    def add_right_child(self, next_element_of_the_list):

        # NON-RECURSIVE FUNCTION
        current = self
        while current is not None:
            if current.right_link_object:
                current = current.right_link_object

            else:
                current.right_link_object = next_element_of_the_list
                current.right_link_value = next_element_of_the_list.nodeID
                return

        # RECURSIVE FUNCTION
        '''if self.right_link_object:
```

```
            self.right_link_object.add_right_child(next_element_of_the_list)

        else:
            self.right_link_object = next_element_of_the_list
            self.right_link_value = next_element_of_the_list.nodeID'''

    def add_right_thread(self, root):
        self.right_thread = root

    def add_left_thread(self, j, w_ij, w_jroot, w_iroot):
        if (w_ij+w_jroot)/2 > w_iroot:
            self.left_thread = j

    def add_community(self, community):
        self.left_link = community

    def right_travers(self):

        # NON-RECURSIVE FUNCTION
        stack = list()
        current = self

        while True:
            if current is not None:
                stack.append(current)
                current = current.right_link_object

            else:
                return stack

        # RECURSIVE FUNCTION
        # elements = list()
        # elements.append(self)

        # if self.right_link_object:
            # elements += self.right_link_object.right_travers()

        # return elements

    def get_node_ids(self):

        # NON-RECURSIVE FUNCTION
        elements = list()
        current = self

        while True:
            if current is not None:
                elements.append(current.nodeID)
                current = current.right_link_object

            else:
                return elements
```

```
        # RECURSIVE FUNCTION
        # elements = list()
        # elements.append(self.nodeID)

        # if self.right_link_object:
            # elements += self.right_link_object.get_node_ids()

        # return elements


def build_tree(sorted_list, edge):

    root = TreeNode(sorted_list[0])

    neighbor_list = [(k, v) for k, v in sorted_list[1].items()]
    # print(neighbor_list)
    m = len(neighbor_list)

    for i in range(m):
        node = TreeNode(neighbor_list[i][0])
        node.add_right_thread(sorted_list[0])
        root.add_right_child(node)

        for j in range(i+1, m):
            node.add_left_thread(neighbor_list[j][0], edge,
neighbor_list[j][1], neighbor_list[i][1])
            break

    return root


def build_communities(tree, trees):
    for node in tree.right_link_object.right_travers():
        for t in trees:
            if node.nodeID == t.nodeID:
                node.add_community(t.right_link_object)


def get_subtree(tree, lt):
    subtree = []

    for node in tree.right_travers():
        subtree.append(node.nodeID)

        if node.nodeID == lt:
            return subtree


def path_sim(root, c):
    if root.nodeID in c:
        # the node already member of the target community
        return
```

```
    cur_path = []
    t_act = root
    active_list = set(root.get_node_ids())
    updates = 0
    length = 0.0

    for element in active_list:
        t_i = t_act.right_link_object

        if t_i is None:
            # the node cant be considered a member of the target community
            return
        else:
            if t_i.left_thread is not None:
                length = length + 1.3 + 1.2
                updates = updates + 2
                cur_path.append(t_act.nodeID)
                cur_path.append(t_i.left_thread)
                cur_path.append(t_i.nodeID)

                for node in t_i.left_link.right_travers():
                    if node.left_link is not None:
                        if node.nodeID == t_i.left_thread:
                            t_act = node
                            break

                s1 = set(t_i.left_link.get_node_ids())
                s2 = set(get_subtree(t_i, t_i.left_thread))
                active_list = s1.difference(set(cur_path), s2)

            else:
                length = length + 4.20
                updates = updates + 1
                cur_path.append(t_i.nodeID)
                t_i = t_act

                if t_i.left_link is not None:
                    s1 = set(t_i.left_link.get_node_ids())
                    active_list = s1.difference(set(cur_path))

        root = t_i

    if root.nodeID in c:
        path_strength = length/updates

        return path_strength, updates
    else:
        # the node cant be considered a member of the target community
        return


# Driver code
if __name__ == '__main__':
```

```
    # neighborhood = (1, {'9': 0.03, '7': 0.06, '77': 0.07, '88': 0.08, '8':
0.09, '2': 0.16, '3': 0.84, '6': 0.93})
    neighborhoods = [('A', {'E': 0.58, 'F': 0.77, 'J': 0.9, 'B': 0.96}),
('E', {'H': 0.56, 'A': 0.58, 'F': 0.82, 'I': 0.84}), ('F', {'A': 0.77, 'E':
0.82, 'G': 0.9}), ('I', {'H': 0.7, 'E': 0.84, 'G': 0.94}), ('G', {'H': 0.8,
'F': 0.9, 'I': 0.92}), ('B', {'A': 0.96, 'C': 1}), ('H', {'E': 0.56, 'I':
0.7, 'G': 0.8}), ('C', {'B': 1}), ('J', {'A': 0.9})]

    com = ['H', 'G', 'F', 'B', 'J']

    weight = 0.66

    dentra = []
    for node_nbr in neighborhoods:
        r = build_tree(node_nbr, weight)
        dentra.append(r)

    for tree in dentra:
        build_communities(tree, dentra)

    for tree in dentra:
        print(path_sim(tree, com))
```

3. tree.py

```
class TreeNode:
    def __init__(self, node):
        self.nodeID = node
        self.right_link_value = None
        self.right_link_object = None
        self.right_thread = None
        self.left_thread = None
        self.left_link = None

    def add_right_child(self, next_element_of_the_list):

        if self.right_link_object:
            self.right_link_object.add_right_child(next_element_of_the_list)

        else:
            self.right_link_object = next_element_of_the_list
            self.right_link_value = next_element_of_the_list.nodeID

    def add_right_thread(self, root):
        self.right_thread = root

    def add_left_thread(self, j, w_ij, w_jroot, w_iroot):
        if (w_ij+w_jroot)/2 > w_iroot:
            self.left_thread = j

    def add_community(self, community):
        self.left_link = community
```

```
    def right_travers(self):
        elements = list()
        elements.append(self)

        if self.right_link_object:
            elements += self.right_link_object.right_travers()

        return elements

    def get_node_ids(self):
        elements = list()
        elements.append(self.nodeID)

        if self.right_link_object:
            elements += self.right_link_object.get_node_ids()

        return elements


def build_tree(sorted_list, edge):

    root = TreeNode(sorted_list[0])

    neighbor_list = [(k, v) for k, v in sorted_list[1].items()]
    # print(neighbor_list)
    m = len(neighbor_list)

    for i in range(m):
        node = TreeNode(neighbor_list[i][0])
        node.add_right_thread(sorted_list[0])
        root.add_right_child(node)
        for j in range(i+1, m):
            node.add_left_thread(neighbor_list[j][0], edge,
neighbor_list[j][1], neighbor_list[i][1])
            break

    return root


def build_communities(tree, trees):
    for node in tree.right_link_object.right_travers():
        for t in trees:
            if node.nodeID == t.nodeID:
                node.add_community(t.right_link_object)


# Driver code
if __name__ == '__main__':

    # neighborhood = (1, {'9': 0.03, '7': 0.06, '77': 0.07, '88': 0.08, '8':
0.09, '2': 0.16, '3': 0.84, '6': 0.93)
    neighborhoods = [(0.1, ('A', {'E': 0.58, 'F': 0.77, 'J': 0.9, 'B':
0.96})), (0.2, ('E', {'H': 0.56, 'A': 0.58, 'F': 0.82, 'I': 0.84})), (0.7,
('F', {'A': 0.77, 'E': 0.82, 'G': 0.9})), (0.8, ('I', {'H': 0.7, 'E': 0.84,
'G': 0.94})), (0.7,('G', {'H': 0.8, 'F': 0.9, 'I': 0.92})), (0.8, ('B', {'A':
0.96, 'C': 1}))]
```

```
    weight = 0.66

    dentra = []
    for node_nbr in neighborhoods:
        r = build_tree(node_nbr[1], weight)
        dentra.append(r)

    for tree in dentra:
        build_communities(tree, dentra)

    # testing trees
    for tree in dentra:
        # root and the right tree
        print('tree with:')
        print('root =', tree.nodeID, 'right subtree--->',
tree.right_link_object.get_node_ids())

        # right thread pointing to the root
        '''for node in tree.right_link_object.right_travers():
            print('node.s', node.nodeID, 'right thread points-->',
node.right_thread)
        print()'''

        # left thread pointing to a child as described in the paper
        '''for node in tree.right_link_object.right_travers():
            if node.left_thread is not None:
                print('node.s', node.nodeID, 'left thread points-->',
node.left_thread)'''

        # left link communities of the children
        for node in tree.right_link_object.right_travers():
            if node.left_link is not None:
                print('node.s', node.nodeID, 'left link community =',
node.left_link.get_node_ids())
```

4. test.py

```
# import json

circles = []
music1 = ['Folk']
music2 = ['Techno/House']
p = 0

# json communities
'''with open('C:\\Users\\giorgos\\Desktop\\musae_git_features.json', 'r') as
f:
    data = json.load(f)
for key, value in data.items():
    if len(value) > 35:
        value.insert(0, p)
        circles.append(value)
        p = p + 1
        # print(key, '-->', value)'''
```

```
# music communities
'''with open('C:\\Users\\giorgos\\Desktop\\HU_genres.json', 'r') as f:
    data = json.load(f)

for key, value in data.items():
    # print(key, '-->', value)
    for kind in value:
        if kind == 'Folk':
            music1.append(key)
        elif kind == 'Techno/House':
            music2.append(key)
circles.append(music1)
circles.append(music2)'''

# normal communities
with open('C:\\Users\\giorgos\\Desktop\\0.circles', 'r') as f:
    for line in f:
        k = line.split()
        circles.append(k)
        c = f.readline().split('\t')
        # print(c)
        c[-1] = c[-1].strip('\n')
        circles.append(c)

for circle in circles:
    print(circle)
```

ΒΙΒΛΙΟΓΡΑΦΗΚΕΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΕΣ ΑΝΑΦΟΡΕΣ

Βιβλιογραφία

A. Clauset, M. N. (2004, Δεκέμβριος). Finding community structure in very large networks. *Phys. Rev. E Stat. Phys*, σσ. 66-111.

Borges, B. (2009). *Marketing 2.0: Bridging the Gap between Seller and Buyer through Social Media Marketing.* Wheatmark.

D. Chen, M. S. (2010, Οκτώβρης). Detecting overlapping communities of weighted networks via a local algorithm. *Phys. A Stat. Mech. Appl.*, σσ. 4177-4187.

D. Džamić, D. A. (2019, Ιανουάριος). "Ascent–descent variable neighborhood decomposition search for community detection by modularity maximization". *Ann. Oper. Res.*, σσ. 273-287.

danah m. boyd, N. B. (2007, Οκτώβριος 01). Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, σσ. 210-230.

Evans, J. S. (2008, Ιανουάριος). Dual-Processing Accounts of Reasoning, Judgment, and Social Cognition. *Center for Thinking and Language, School of Psychology*.

Gregory, S. (2010, Οκτώμβριος 10). Finding overlapping communities in networks by label propagation. *New J. Phys*.

H. Sarvari, E. A. (2014). Constructing and analyzing criminal networks. *IEEE* .

I. Farkas, G. P. (2007, Ιούνιος). "Weighted network modules". *New J. Phys*, σ. 180.

J. A. Bondy, U. S. (1976). *Graph Theory woth Applications.* Ontiario, Canada: Department of Combinatorics and Opimaization, University of Waterloo.

J. C. Valverde-Rebaza, a. A. (2012). "Link prediction in complex networks based on cluster information. *Brazilian Conf. in Artificial Intelligence,*.

J. Huang, H. S. (2013, Αύγουστος). "Revealing density-based clustering structure from the core-connected tree of a network"IEEE Trans. *IEEE Trans.*, σσ. 1876-1889.

J. J. Bechtel, W. A. (2005). “Lung cancer detection in patients with airflow obstruction identified in a primary care outpatient practice,. *Chest*.

J. M. Kumpula, M. K. (2008, Αυγουστος). "Sequential algorithm for fast clique percolation". *Phys. Rev. E Stat*.

James Evans, E. M. (2019). *Optimization Algoritms for Network and Graphs.* New york: Marcel Dekker.

Joinson, A. (2008). Looking at, Looking up or Keeping up with People? Motives and Use of Facebook. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,. *Sientific research* , σσ. 1027-1036.

Krevl, J. L. (2014, Ιούνιος). *SNAP datasets: Stanford largenetwork dataset collection*. Ανάκτηση από http://snap.stanford.edu/data.

M. J. Mosadegh, a. M. (2011). “Using social network paradigm for developing a conceptual framework in CRM,. *Australian J. Bus*.

M. Salathé, a. J. (2010). “Dynamics and control of diseases in networks with community structure. *PLoS Computational Biology,*.

mcQuail, D. (1994). Mass communication theory. *Scientific research*.

Meyerhenke, C. L. (2016, Ιανουάριος). "Engineering parallel algorithms for community detection in massive networks". *IEEE Trans. Parallel Distrib*, σσ. 171-184.

N. Haq, a. Z. (2016). Community detection from genomic datasets across human cancers. *IEEE Global Conf. on Signal and Infor. Process*.

Newman, M. E. (2006). Modularity and community structure in networks. *Nat. Acad. Sci.*

Padrol-Sureda, G. P.-L.-M. (2010, Μάρτιος). Overlapping community search for social networks. *Proc. 26th IEEE Int*, σσ. 992-995.

Pinheiro, C. A. (2012). “Community detection to identify fraud events in . *SAS SUGI proceedings: customer* .

Pitsoulis, M. C. (2013, Δεκεμβρης). Community detection by modularity maximization using GRASP with path relinking. *Comput. Oper. Res.*, σσ. 3121-3131.

Prat-Pérez, D. D.-S.-L.-P. (2012, Οκτώβριος). Shaping communities out of triangles. *Proc. CIKM*, σσ. 1677-1681.

Qiao et al, 2. (2018, Σεπτέμβρης). "A fast parallel community discovery model on complex networks through approximate optimization. *IEEE Trans.*, σσ. 1638-1651.

RominaCachia, R. C. (2007, Οκτώβριος). Grasping the potential of online social networks for foresight. *Technological Forecasting and Social Change*, σσ. 1179-1203.

S. B. Abdrabbah, R. A. (2014). Collaborative filtering based on dynamic community detection. *Dynamic Networks and Knowledge Discovery*.

S. Soundarajan, a. J. (2012,). Using community information to improve the precision of link prediction methods. *Conf. on World Wide Web*.

Saad, J. C. (2012, Ιούλιος). Dense subgraph extraction with application to community detection. *IEEE Trans.*, σσ. 1216-1230.

Şahin, A. K. (2017). A Review on Social Bot Detection Techniques and Research Directions. *in Proc. Int. Security and*.

Shannon, W. (1948). A Mathematical Theory of Communication or or Shannon-Weaver model of communication. *Bell System Technical Journal* .

Sifaleras, S. S. (2018, Μάρτιος). "Efficient community-based data distribution over multicast trees. *IEEE Trans.*

Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication. (2017, October 2007). *Journal of Computer-Mediated Communication*.

Stavros Souravlas, A. S. (2019, February 7). A Parallel Algorithm for Community Detection in Social Networks, Based on Path Analysis and Threaded Binary Trees. *IEEE Access*, σσ. 20499 - 20519.

Tung, F. Z. (2012, Δεκέμβρης). "Large scale cohesive subgraphs discovery for social network visual analysis". *VLDB Endowment*, σσ. 85-96.

U. N. Raghavan, R. A. (2007). "Near linear time algorithm to detect community structures in large-scale networks". *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat.*

V. D. Blondel, J.-L. G. (2008, Οκτώβρης). "Fast unfolding of communities in large network". *J. Stat. Mech. Theory Exp*.

W. Zhi-Xiao, L. Z.-c.-f.-h. (2016, Αύγουστος). "Overlapping community detection based on node location analysis". *Knowl.-Based Syst*, σσ. 225-235.

Waskiewicz, T. (2012). "Friend of a friend influence in terrorist social . *in Proc Int. Conf. on Artificial Intelligence*.

Wikipedia. (2011, October). Ανάκτηση από Threaded Binary Tree: https://en.wikipedia.org/wiki/Threaded_binary_tree

Wilson, R. J. (1996). *Introduction to Graph Theory.* Harlow England: Addison Wesley Longman Limited Edinburgh Gate.

X. Zhang, C. W.-F. (2017, Δεκέμβριος). "A fast overlapping community detection algorithm based on weak cliques for large-scale networks". *IEEE Trans. Comput*, σσ. 218-230.

Z. Lu, X. S. (2015, Νοέμβρης). "Algorithms and applications for community detection in weighted networks". *IEEE Trans*, σσ. 2916-2926.

Σύνδεσμοι Διαδικτύου

- https://algorithms.tutorialhorizon.com/
- https://networkx.org/documentation/stable/tutorial.html
- https://networkx.org/documentation/stable/reference/algorithms/index.html
- https://networkx.org/documentation/stable/reference/generated/networkx.classes.function.density.html
- https://www.geeksforgeeks.org/python-sort-dictionary-key-and-values-list/
- https://code.activestate.com/recipes/522995-priority-dict-a-priority-queue-with-up-updatabple-prio/
- https://en.wikipedia.org/wiki/Threaded_binary_tree

- https://en.wikipedia.org/wiki/Bitwise_operation#XOR
- https://www.geeksforgeeks.org/bitwise-xor-of-a-binary-array/
- https://www.geeksforgeeks.org/binary-representation-of-a-given-number/
- https://www.geeksforgeeks.org/python-slicing-extract-k-bits-given-position/
- https://www.geeksforgeeks.org/find-position-of-the-only-set-bit/
- https://www.geeksforgeeks.org/count-set-bits-in-an-integer/
- https://www.geeksforgeeks.org/threaded-binary-tree-insertion/?ref=lbp
- https://www.geeksforgeeks.org/binary-search-tree-data-structure/
- https://medium.com/analytics-vidhya/deep-dive-into-threaded-binary-tree-step-by-step-aa8f90400c5a
- https://www.programmersought.com/article/1248255375/
- https://github.com/topics/binary-search-tree?l=python
- https://realpython.com/python-mmap/
- http://snap.stanford.edu/data/
- https://www.geeksforgeeks.org/how-are-variables-stored-in-python-stack-or-heap/
- https://medium.datadriveninvestor.com/how-does-memory-allocation-work-in-python-and-other-languages-d2d8a9398543
- https://www.geeksforgeeks.org/inorder-tree-traversal-without-recursion/
- https://towardsdatascience.com/dont-use-recursion-in-python-any-more-918aad95094c
- https://realpython.com/python-memory-management/?fbclid=IwAR3l2YSULpbV27HmSmaIMnW8eHULTrSCQGtAKQxK-IdvtILqkUuaCaUYdPY
- https://docs.python.org/3.8/c-api/memory.html