

Εργασία για το μάθημα Θεωρία Δικτύων

Πλευρίδη Βασιλική Βαρβάρα (AEM:10454)

Δεκέμβριος 2024

1 Εισαγωγή

Η εργασία αυτή έχει ως στόχο την εύρεση κοινοτήτων σε δίκτυα με χρήση ενός αλγορίθμου που χρησιμοποιεί την συνάρτηση ποιότητας ομάδοποιήσης απόστασης (distance quality function), αφού πρώτα αυτή σχεδιαστεί και υλοποιήθει σε python. Παρακάτω θα αναλυθούν αρχικά, μαθηματικά και αλγορίθμικά, οι υποσυναρτήσεις που υλοποιήθηκαν και χρησιμοποιήθηκαν για την τελική συνάρτηση υπολογισμού του Q_d , η συνάρτηση μεγιστοποίησης της, οι αλλαγές που έγιναν με στόχο την εύρεση καλύτερων αποτελεσμάτων, οι οπτικοποιήσεις των βέλτιστων κοινοτήτων με την χρήση του Gephi, η σύγχριση των αποτελεσμάτων με αυτά που προκύπτουν από την χρήση της συνάρτησης modularity και τέλος τα συμπεράσματα που προέκυψαν από όλα τα παραπάνω.

2 Βήματα υπολογισμού της distance quality function

Στο αρχείο ‘‘φυντιόν’’ υλοποιούνται οι βοηθητικές συναρτήσεις αλλά και η τελική συνάρτηση compute_qd με σκοπό τον υπολογισμό εν τέλει της Q_d . Η λογική των συναρτήσεων είναι σύμφωνη με την θεωρητική παρουσίαση ‘‘communities’’ του μαθήματος και συγκεκριμένα στις διαφάνειες 53-58. Η ακριβής διατύπωση τους θα οδηγούσε σε περίσσειους υπολογισμούς και έτσι σε μια πιο χρονοβόρα υλοποίηση. Για αυτό, έγινε προσπάθεια βελτιστοποίησης τους είτε με εργαλεία της python είτε με την απαλοιφή αχρείαστων υπολογισμών

2.1 Υπολογισμός αποστάσεων μεταξύ κόμβων

Αρχικά με την βοήθεια της συνάρτησης shortest_path_length της βιβλιοθήκης networkx υπολογίζονται οι αποστάσεις μεταξύ όλων των ζευγών κόμβων. Οι τιμές αυτές αποθηκεύονται σε έναν νέο πίνακα Dv, όπου κάθε κόμβος αναπαρίσταται από τον δείκτη του, και οι τιμές δείχνουν τις αποστάσεις στους γείτονές του.

2.2 Υπολογισμός μήτρας γειτνίασης (adjacency matrix)

Με την βοήθεια της συνάρτησης adjacency_matrix(G), ο υπολογισμός του adjacency matrix ενός γράφου G γίνεται πολύ εύκολα και γρήγορα.

2.3 Υπολογισμός μέγιστων αποστάσεων

Σε αυτό το σημείο υπολογίζεται η μέγιστη απόσταση (κμαχ) μεταξύ οποιουδήποτε ζεύγους κόμβων στον γράφο, χρησιμοποιώντας τον πίνακα αποστάσεων Dv. Ουσιώστικά ισοδυναμεί με την διάμετρο του γράφου $G(V, E)$ και εκφράζεται από την μαθηματική σχέση:

$$\text{diam}(G) = \max\{D_V(i, j) : \forall i, j \in V\}.$$

2.4 Paths Matrix

Ουσιαστικά αυτός ο πίνακας περιέχει τον αριθμό των διαδρομών μήκους k μεταξύ όλων των κόμβων. Στην περίπτωση μη κατευθυνόμενων γράφων, ο πίνακας αυτός είναι συμμετρικός. Έτσι, αποσοπώντας στην αποφυγή περιτών πράξεων, υπολογίζονται μόνο τα στοιχεία του άνω τριγωνικού πίνακα ενώ τα υπόλοιπα καταγράφονται με βάση την συμμετρία αυτού. Η μαθηματική έκφραση του είναι η εξής:

$$P_V(i, j) = A_G^l(i, j) \quad \text{where } l = \min\{k : A_G^k(i, j) \neq 0\}$$

όπου $A_G^l(i, j)$ είναι ο adjacency matrix υψηλότερου στρώματος k με $k \in [1, k_{\max}]$

2.5 Generilzed degree

Ο generilzed μιας κορυφής ν είναι ο αριθμός των συντομότερων διαδρομών μήκους k στις οποίες η κορυφή αυτή συμμετέχει ως αρχική κορυφή. Για την αποθήκευση όλων αυτών των βαθμών ορίστηκε ένας δισδιάστατος πίνακας $k_{\max} \times \text{num of v}$. Ο πίνακας αυτός υπολογίζεται με την βοήθεια της συνάρτησης generalized_degree η οποία υλοποιεί την παρακάτω μαθηματική έκφραση:

$$d_k(v) = \sum \{P_V(v, i) : D_V(v, i) = k, k \in [1, k_{\max}]\}$$

2.6 Υπολογισμός αναμενόμενων αποστάσεων

Υπολογίζοντας αρχικά τον συνολικό αριθμό των shortest paths μήκους $k \leq k_{\max}$ (m), είναι εύκολο να υπολογιστούν οι εκτιμόμενες αποστάσεις μεταξύ των κορυφών του γράφου από την σχέση:

$$\overline{D_v(C)} = \frac{1}{2} \sum_{i, j \in C} \sum_{k=1}^{\text{diam}(G)} k \cdot \left(\frac{d_k(i)}{2m_k(G)} \cdot \frac{d_k(j)}{2m_k(G)} \right)$$

Η συνάρτηση που υλοποιεί τα παραπάνω είναι η compute_expected_distance.

2.7 Υπολογισμός πραγματικών αποστάσεων

Ουσιαστικά, στόχος μας είναι να υπολογίσουμε το άθροισμα των πραγματικών αποστάσεων μεταξύ κόμβων κάθε cluster C. Για να το πετύχουμε αυτό, χρησιμοποιούμε τις ήδη υπολογισμένες αποστάσεις που έχουν αποθηκευτεί στον distance matrix.

2.8 Τελική τιμή distance quality

Εν τέλει, το τελικό αποτέλεσμα προκύπτει από την διαφορά του αθροίσματός για όλα τα clusters του γράφου των αναμενόμενων με των πραγματικών αποστάσεων, δηλαδή από την σχέση:

$$Q_d(G, \mathcal{C}) = \sum_{C \in \mathcal{C}} (\overline{D_V(C)} - D_V(C))$$

όπου $\overline{D_V(C)}$ οι εκτιμώμενες και $D_V(C)$ οι πραγματικές αποστάσεις που υπολογίστηκαν παραπάνω.

3 Μεγιστοποίηση συνάρτησης ποιότητας

3.1 Σχεδιασμός αλγορίθμου

Για την μεγιστοποίηση της συνάρτησης ποιότητας σχεδιάστηκε και υλοποιήθηκε ο εξής αλγόριθμος:

```

0: function OPTIMIZE_CLUSTERS( Γράφος  $G(V, E)$ )
1: Αρχικοποίηση κάθε κόμβου σε ξεχωριστό cluster
2: Υπολογισμός αρχικού  $Q_d$ 
3: while υπάρχει δυνατότητα βελτίωσης  $Q_d$  do
4:   for all ζεύγη clusters  $C_i, C_j$  do
5:     Υπολόγιση τη νέα τιμή  $Q_d$  αν συγχωνευτούν τα  $C_i$  και  $C_j$ 
6:     if η νέα τιμή  $Q_d$  είναι μεγαλύτερη then
7:       Συγχώνευσε τα  $C_i$  και  $C_j$ 
8:       Ενημέρωσε  $Q_d$ 
9:     end if
10:    end for
11:  end while
12: return: τελικά clusters και  $Q_d$ 

```

Η βασική ιδέα του αλγορίθμου είναι αρκετά απλή και ο παραπάνω ψευδοκώδικας την παρουσιάζει αρκετά καλα. Παρόλα αυτά, κρίνεται σημαντικό να σχολιαστούν μερικά σημεία της τελικής υλοποίησης του:

- Για την αποφυγή προσπάθειας συγχώνευσης clusters τα οποία δεν έχουν καμία ακμή να τα ενώνει, προστέθηκε ένας έλεγχος ο οποίος στην περίπτωση μη εύρεσης ακμών μεταξύ τους, παραλείπει την παρούσα σύγκριση και συνεχίζει στο επόμενο ζευγάρι clusters. Με αυτόν τον τρόπο, επιταχύνεται ο χρόνος εκτέλεσης της συνάρτησης και μειώνεται η πολύπλοκότητά της.
- Επειδή η λογική του αλγορίθμου βασίζεται σε ένα διπλό forloop το οποίο συγχρίνει το i-οστό cluster με όλα τα υπόλοιπα j, σε περίπτωση που βρει μια μεγαλύτερη τιμή για το Q_d , είναι σημαντικό οι συγκρίσεις να ξεκινήσουν από την αρχή. Για αυτό τον λόγο, και στα δύο στάδια του forloop υπάρχει η εντολή break που εξυπηρετεί αυτή την περίπτωση.
- Για τη συγχώνευση των clusters, οι τιμές τους αποθηκεύονται σε ένα νέο κοινό cluster. Στην περίπτωση που η νέα τιμή περάσει τον έλεγχο, το νέο cluster αντικαθιστά το ένα από τα δύο αρχικά, ενώ το άλλο διαγράφεται πλήρως. Σημειώνεται ότι, για πιο αποδοτική και γρήγορη επεξεργασία, τα clusters είναι αποθηκευμένα σε μορφή λίστας.
- Για την αποφυγή περιττών υπολογισμών, σε κάθε σύγκριση δεν υπολογίζεται εξ αρχής το Q_d . Αντίθετα, επικεντρωνόμαστε στις αποστάσεις που αλλάζουν, δηλαδή στις αποστάσεις των clusters που συγχρίνονται κάθε φορά. Έτσι, υπολογίζονται μόνο οι αναμενόμενες και οι πραγματικές αποστάσεις μεταξύ των κόμβων του νέου cluster, το οποίο επιχειρείται να δημιουργηθεί. Με την προσθμαφάρεση των κατάλληλων τιμών, υπολογίζεται εύκολα η νέα αναμενόμενη τιμή του Q_d σε περίπτωση συγχώνευσης.
- Η συνάρτηση υπολογισμού της Q_d , ουσιαστικά δεν χρησιμοποιείται εν τέλει καθόλου. Αντιθέτως οι υπολογισμοί ενσωματώνονται μέσα στην συνάρτηση μεγιστοποίησης με στόχο την πιο έξυπνη και γρήγορη διαχείρηση των δεδομένων, η λογική της οποίας αναφέρθηκε παραπάνω. Παρόλα αυτά στο αρχείο "functions", υπάρχει και η αρχικά υλοποιημένη συνάρτηση.

3.2 Εγγενές Πρόβλημα στη Μέγιστη Τιμή της Συνάρτησης Ποιότητας

Η Q_d τείνει να μεγιστοποιείται όταν ελαχιστοποιούνται οι αποστάσεις εντός κάθε cluster. Σε μια ακραία περίπτωση, όταν κάθε κόμβος ορίζεται ως ξεχωριστό cluster, η ενδο-ομαδική απόσταση είναι η ελάχιστη δυνατή, δηλαδή 0. Αυτός ο τρόπος ομαδοποίησης μεγιστοποιεί μαθηματικά τη συνάρτηση ποιότητας, αλλά οδηγεί σε άτοπο αποτέλεσμα, καθώς παραβλέπει τη βασική έννοια της ομαδοποίησης.

3.3 Τροποποίηση Συνάρτησης Ποιότητας

Για να αποφευχθεί αυτό, απαιτείται μια προσέγγιση που εισάγει περιορισμούς ή τροποποιήσεις στη συνάρτηση, ώστε να λαμβάνονται υπόψη οι άλληλεπιδράσεις μεταξύ των κόμβων και να προάγεται μια πιο ουσιαστική ομαδοποίηση. Μία σκέψη, η οποία είναι και αυτή που εν τέλει δοκιμάστηκε, είναι η εισαγωγή συντελεστών α και β ,

οι οποίοι θα προσδιορίζουν την επίδραση των αναμενόμενων και πραγματικών αποστάσεων μεταξύ των κόμβων. Ο συντελεστής α θα ενισχύει τις αναμενόμενες αποστάσεις, ενώ ο β θα ενισχύει τις πραγματικές αποστάσεις. Ουσιαστικά η ανανεωμένη σχέση υπολογισμού της Q_d θα είναι η:

$$Q_d(G, \mathcal{C}) = \sum_{C \in \mathcal{C}} (\alpha * \overline{D_V(C)} - \beta * D_V(C))$$

Για λόγους απλότητας και αποφυγής ύπαρξης δύο μεταβλητών, το β τέθηκε ίσο με 1.

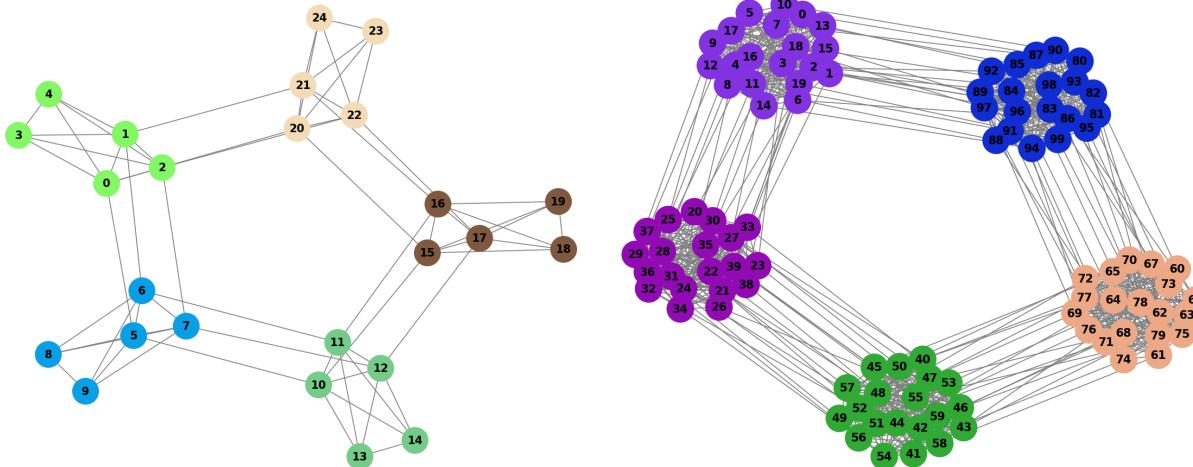
4 Παράγοντας α και Παραδείγματα Λειτουργικότητας

Όπως είναι λογικό, ο παράγοντας α που προστέθηκε στην έκφραση του Q_d δεν μπορεί να είναι ίδιος για όλους τους γράφους αλλά εξαφτάται από τα χαρακτηριστικά του εκάστοτε γράφου. Η εύρεση της κατάλληλης τιμής του αποτελεί το κλειδί για μία καλή ομαδοποίηση αυτού. Παρακάτω θα παρουσιαστούν παραδείγματα γράφων που με την κατάλληλη επιλογή του α οδήγησαν σε σωστά αποτελέσματα clustering και επαλήθευσαν την λειτουργικότητα της συνάρτησης της Ενότητας 3.

4.1 Δοκιμή αλγορίθμου σε μικρούς γράφους

Για να ελέγξουμε την λειτουργικότητα του αλγορίθμου μεγιστοποίησης, σχεδιάστηκε και υλοποιήθηκε η συνάρτηση `generate_custom_graph(x, z, y)` η οποία δημιουργεί έναν γράφο με x κοινότητες, με κάθε μία να έχει z κόμβους και να αποτελεί ένα πλήρες υπογράφημα. Τέλος, μπορεί να ρυθμιστεί και ο αριθμός των ακμών για μεταξύ των κοινοτήτων. Η σύνδεση των κοινοτήτων γίνεται σε κύκλο.

Χρησιμοποιώντας ως ορίσματα τις τιμές $x = 5, z = 5, y = 3$ και έπειτα από διερεύνηση της κατάλληλης τιμής της παραμέτρου $\alpha = 120$ για την χρήση του αλγορίθμου μεγιστοποίησης της Q_d , προέκυψε ο παρακάτω γράφος (Σχήμα 1) με τα clusters του:



Σχήμα 1: $\alpha = 120$

Σχήμα 2: $\alpha = 2000$

Όπως φαίνεται από παραπάνω, η χρήση της συνάρτησης οδήγησε στα σωστά clusters. Αξίζει όμως να δοκιμάστε και σε γράφο με πιο ισχύρες σχέσεις ανάμεσα στις κοινότητες του, οι οποίες θα αποτελούνται από περισσότερους κόμβους ή κάθε μία. Για αυτό καλέστηκε η συνάρτηση με παραμέτρους $x = 5, z = 20, y = 20$ και $\alpha = 2000$ (Σχήμα 2). Ο εντοπισμός των κοινοτήτων είναι και πάλι εύκολος καθώς η σύνδεση των κόμβων ενός cluster είναι πολύ πιο ισχυρή συγκριτικά με αυτή των clusters μεταξύ τους. Πιο σημαντική είναι η παρατήρηση ότι όπως και περιμέναμε, το α χρειάστηκε να αυξηθεί κατα πολύ για να ανταποχριθεί ο αλγόριθμος στον νέο γράφο.

4.2 Προσδιορισμός κατάλληλου α

Για τον προσδιορισμού του α, το οποίο εξαρτάται από τα χαρακτηριστικά του εκάστοτε γράφου, στα πλαίσια της εργασίας, δεν υλοποιήθηκε κάποια συνάρτηση. Παρόλα αυτά, μια ίδεα που θα οδηγούσε σχετικά γρήγορα σε σωστές τιμές είναι ο σχεδιασμός μιας συνάρτησης που θα βασιζόταν στην μέθοδο της διχοτόμου με κριτήριο επιλογής να είναι καποια μέθοδος σύγκρισης ομοιότητας clustering όπως το Jaccard similarity. Για την εργασία, οι τιμές του α, επιλέχθηκαν με βάση οπτικής παρατήρησης των γράφων.

5 Χρήση του αλγορίθμου σε μεγάλους γράφους

Ο αλγόριθμος δοκιμάστηκε σε ένα δίκτυο το οποίο αναπαριστά την επικοινωνία μέσω email μεταξύ μελών ενός μεγάλου ερευνητικού οργανισμού. Πρόκειται για έναν κατευθυνόμενο γράφο, όπου οι κόμβοι αντιστοιχούν στα μέλη του οργανισμού, και οι κατευθυνόμενες ακμές αναπαριστούν την αποστολή email από ένα μέλος σε ένα άλλο. Ο γράφος αποτελείται από 1005 κόμβους και έχει 25571 ακμές, γεγονός που τον χαρακτηρίζει ως πυκνό. Κάθε κόμβος ανήκει σε μία από τις 42 προκαθορισμένες κοινότητες, οι οποίες αντιστοιχούν στα τμήματα του οργανισμού. Τα δεδομένα πάρθηκαν από το Stanford Network Analysis Project (SNAP), συγκεκριμένα από το dataset "email-Eu-core".

5.1 Επιλογή παράγοντα α

Όσο ο γράφος μεγαλώνει η επιλογή αυτού του παράγοντα γίνεται όλο και πιο δύσκολη. Αρχικά έγινε προσπάθεια επιλογής του α τέτοιο ώστε να επιτευχθεί η ύπαρξη 42 κοινοτήτων ($\alpha = 411000$). Η επιλογή αυτή ήταν λανθασμένη καθώς απαιτούσε τον ακριβή εντοπισμό των clusters, γεγονός που η συνάρτηση μεγιστοποίησης του Q_d δεν μπορεί να μας παρέχει. Επομένως, σε πρώτο επίπεδο, στόχος μας δεν είναι να προσεγγίσουμε τον αριθμό των clusters αλλά να εντοπίσουμε στον γράφο τις κύριες κοινότητες του. Έπειτα από διερεύνηση, ως range που οδηγεί σε καλύτερα αποτελέσματα ορίστηκε το $\alpha \in [60000, 65000]$, με την καλύτερη δοκιμασμένη τιμή να είναι το $\alpha = 65000$. Τα γραφικά αποτελέσματα και ένας παραπάνω σχολιασμός θα παρουσιαστούν στην επόμενη υποενότητα.

5.2 Οπτικοποίηση κοινοτήτων μέσω Gephi

Για την οπτικοποίηση των βέλτιστων κοινοτήτων, όπες αυτές προκύπτουν από την συνάρτησης μεγιστοποίησης συνάρτησης ποιότητας, χρησιμοποιήθηκε η εφαρμογή Gephi. Για τον σκοπό αυτό, οι τελικές κοινότητες αποθηκεύτηκαν σε αρχείο csv, και εφαρμόστηκαν στον γράφο.

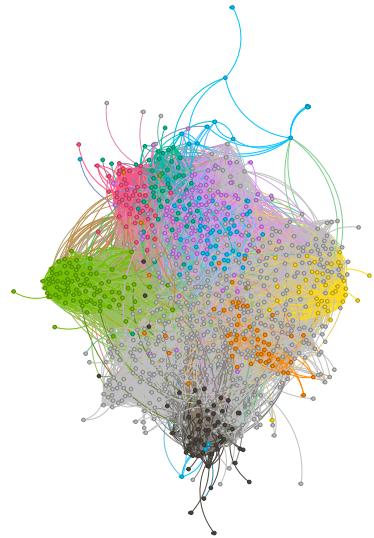
5.2.1 Σύγκριση με χρήση Modularity

Ένας άλλος τρόπος ανίχνευσης κοινοτήτων είναι ένας αλγόριθμος που επιλέγει τις κοινότητες με βάση την μεγιστοποίηση του modularity. Ουσιαστικά, όπως θα δούμε και στις απεικονίσεις, ο αλγόριθμος αυτός λειτουργεί έτσι ώστε οι ακμές εντός των κοινοτήτων να είναι όσο το δυνατόν περισσότερες ενώ αντίθετα οι ακμές μεταξύ διαφορετικών κοινοτήτων να είναι όσο το δυνατόν λιγότερες. Αυτό συμβαίνει γιατί το modularity, όν και ισχυρό εργαλείο, έχει την τάση να ενώνει μικρές κοινότητες σε μεγαλύτερες, όταν αυτές οι μεγαλύτερες κοινότητες αυξάνουν το συνολικό modularity score.

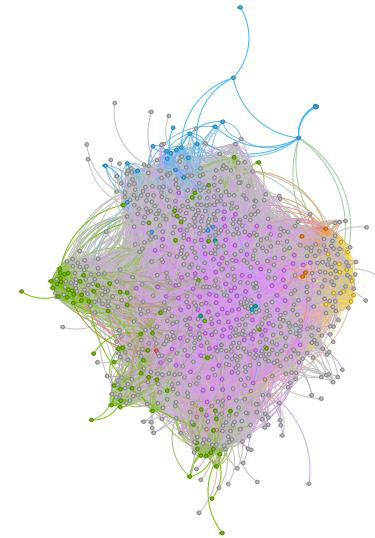
Μέσω του Gephi, και συγκεκριμένα από την επιλογή Community Detection Modularity, παρακάτω θα οπτικοποιηθεί και ο γράφος με κοινότητες βασισμένες στο Modularity.

5.3 Οπτικά αποτελέσματα clustering

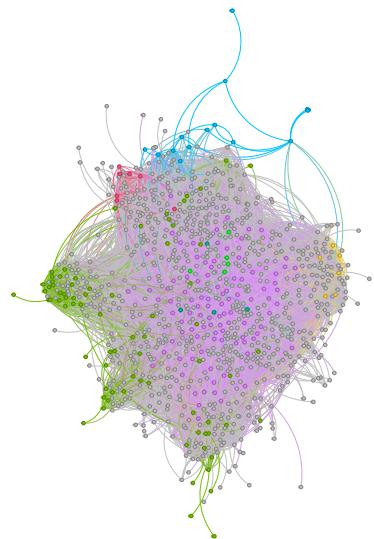
Παρακάτω παρουσιάζονται τα αποτελέσματα clustering (Σχήμα 4 και 5) με τιμές του α 65000 και 60000 αντίστοιχα, συγκρινόμενα με τις πραγματικές κοινότητες (Σχήμα 3) και αυτές που προκύπτουν από την χρήση Community Detection Modularity (Σχήμα 6):



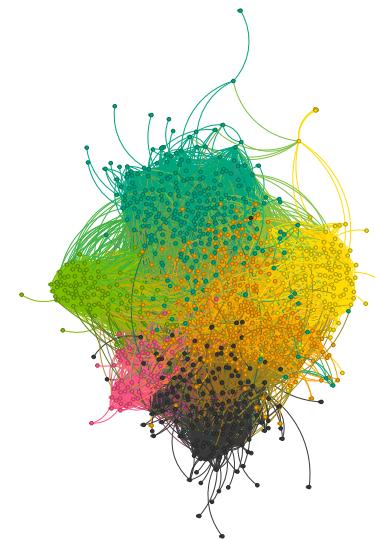
Σχήμα 3: Πραγματικές κοινότητες



Σχήμα 4: Κοινότητες με $\max Q_d, \alpha = 65000$



Σχήμα 5: Κοινότητες με $\max Q_d, \alpha = 60000$



Σχήμα 6: Κοινότητες με max Modularity

5.4 Σχολιασμός αποτελεσμάτων

Τα αποτελέσματα clustering με μεγιστοποίησης της Q_d , δεν φαίνονται πάρα πολύ ικανοποιητικά. Ενώ εντοπίζει μερικές κοινότητες, είτε τις έχει ενώσει με κάποια άλλη (πράσινη και μάυρη), είτε δεν τις εντοπίζει ολόκληρες (γαλαζιά ή και κίτρινη). Παρόλα αυτά, συγχριτικά με του modularity που όπως αναφέρθηκε και πιο πάνω τείνει να δημιουργήσει μεγάλες κοινότητες, με αποτέλεσμα να τις αλλοιώνει τελικά αρκετά, είναι πιο κοντά στις πραγματικές κοινότητες.

Για $\alpha = 65000$, εντοπίζονται καλύτερα οι κοινότητες και επίσης φαίνεται να μειώνεται η μεγάλη λανθασμένη κοινότητα (μωβ) που δημιουργείται. Από την άλλη για $\alpha = 60000$, γίνεται εντοπισμός μίας ακόμης κοινότητας (φουξ). Σίγουρα μπορεί κάποια τιμή μέσα σε αυτό το φάσμα, να οδηγούσε σε καλύτερα αποτελέσματα.

Ο παρατηρούμενος αριθμός κοινοτήτων (600-700) είναι σημαντικά μεγαλύτερος από τον πραγματικό αριθμό (42). Αυτό μπορεί να αποδούθει στη φύση του αλγορίθμου που χρησιμοποιείται για την ανίχνευση κοινοτήτων. Ο συγκεκριμένος αλγόριθμος τείνει να υπερδιασπά τον γράφο, δημιουργώντας πολλές μικρές κοινότητες που αποτελούνται από λίγους κόμβους.

Το παραπάνω δημιούργησε έναν προβληματισμό σχετικά με την σύγκριση αλγορίθμων μέσω του Jaccard similarity. Η μεγάλη απόκλιση στον αριθμό των κοινοτήτων, με 42 πραγματικές κοινότητες και 600-700 ανιχνευμένες κοινότητες, προκαλεί σημαντική δυσκολία στην άμεση σύγκριση τους. Ο Jaccard similarity υπολογίζει την ομοιότητα δύο συνόλων με βάση τη διασταύρωσή τους, όμως όταν έχουμε τόσο μεγάλη διαφορά στον αριθμό των κοινοτήτων, η άμεση σύγκριση γίνεται προβληματική.

6 Σχολιασμός

Η μεγιστοποίηση της συνάρτησης ποιότητας δυστυχώς δεν οδήγησε σε πολύ ικανοποιητικά αποτελέσματα. Τσως με μια καλύτερη επιλογή του παράγοντα α να βελτιώνονταν αλλά σίγουρα δεν θα πρόσεγγιζαν σχετικά καλά τις πραγματικές κοινότητες. Η δημιουργία μιας συνάρτησης επιλογής του παράγοντα αυτού θα βοηθούσε αρκετά στο πλαίσιο αυτής της διερεύνησης. Ο χρόνος όμως εκτέλεσης με μία μόνο τιμή του α κυμαίνεται στα 4 με 5 λεπτά, γεγονός που θα οδηγούσε σε πολύωρη αναζήτηση. Τσως αν ο αλγόριθμος είχε βελτιστοποιηθεί κι άλλο, μειώνοντας την πολυπλοκότητά του, να είχε περισσότερο νόημα μια τέτοια υλοποίηση.

7 Αρχεία Παραδότεου

Τα αρχεία του τελικού παραδότεου περιλαμβάνουν:

1. Δύο αρχεία Python:
 - functions.py: Περιέχει όλες τις συναρτήσεις που σχεδιάστηκαν και υλοποιήθηκαν.
 - main.py: Περιλαμβάνει την κλήση των συναρτήσεων για την επεξεργασία και οπτικοποίηση των γράφων που δοκιμάστηκαν.
2. Ένα αρχείο Gephi:
 - Περιέχει τις γραφικές απεικονίσεις των 4 διαφορετικών clustering που παρουσιάστηκαν παραπάνω.
3. Πέντε αρχεία CSV:
 - Το αρχικό σύνολο δεδομένων του γράφου.
 - Τα clusters για τα τέσσερα διαφορετικά clustering που παρουσιάζονται στο Gephi.