Εργασία Θεωρίας Δικτύων

Φίλιππος Ρωσσίδης
(ΑΕΜ 10379)

11 Ιανουαρίου 2025

1 Εισαγωγή

Στην εργασία αυτή εξετάζουμε την αναγνώριση κοινοτήτων. Συγκεκριμένα θα υλοποιούμε την distance quality function θα ελέγξουμε κατά πόσο αυτή είναι καλή επιλογή.

Οι συναρτήσεις αυτές (quality functions) δέχονται ως όρισμα έναν γράφο και μια κατανομή κοινοτήτων και επιστρέφουν μία τιμή. Σκοπώς μας είναι να επιστρέφουν υψηλές τιμές για κατανομές που παρουσιάζουν ισχυρή κοινοτική συμπεριφορά, ώστε να μπορούμε μεγιστοποιώντας αυτή τη συνάρτηση να ανιχνεύσουμε τις κοινότητες.

Θα υλοποιηθεί η συνάρτηση με έναν βέλτιστο ως προς τη ταχύτητα τρόπο, όπως και δύο αλγόριθμοι μεγιστοποίησής της. Θα υλοποιηθούν και κάποια benchmarks ώστε να ποσοτικοποιηθεί η απόδοσή της και θα συγκριθεί με την εδραιωμένη μετρηκότητα modularity.

2 Η συνάρτηση distance quality

2.1 Μαθηματικός Ορισμός

Η συνάρτηση δέχεται σαν όρισμα έναν γράφο και κάποια κατανομή κοινοτήτων.

Ορίζουμε:

$$D_V(i,j) = \min\{k : A_G^k(i,j) \neq 0\}$$

τον πίνακα που περιέχει τις αποστάσεις μεταξύ κάθε δύο κόμβων του γράφου i, j, όπου $A_G^k(i,j)$ ο πίνακας γειτνίασης του γράφου G. υψομένος στην k δύναμη.

Ορίζουμε επίσης:

$$D_V(c) = \sum_{i,j \in c} D_V(i,j)$$

Αυτό αποτελεί μέτρο του πόσο στενά συνδεδεμένες είναι μεταξύ τους οι κοινότητες. Επιθυμούμε να το συγκρίνουμε με την αναμενώμενη τιμή για τυχαίο γράφο.

Για να ορίσουμε τον τυχαίο γράφο επιλέγουμε το γενικευμένο μοντέλο, όπου συνδέουμε τυχαία ακμές μεταξύ κόμβων, διατηρώντας όμως τον βαθμό τους.

Αν $d_k(v)$ είναι ο k βαθμός του κόμβου v, δηλαδή το πλήθος των ελάχιστων μονοματιών μήκους k που ξεκινάν από τον v και $m_k(G) = \frac{1}{2}\sum_{v \in V(G)} d_k(v)$ το πλήθος ελάχιστων μονοπατιών μήκους k στον γράφο G, τότε η πιθανότητα δύο κόμβοι $i, j \in G$ να συνδέονται με ακμή μήκους k είναι:

$$Pr[i,j,k] = \frac{d_k(i)}{2m_k(G)} \frac{d_k(j)}{2m_k(G)}$$

η αναμενώμενη απόσταση των κόμβων:

$$\overline{D_V(i,j)} = \sum_{k=1}^{diam(G)} kPr[i,j,k]$$

όπου $diam(G)$ η διάμετρος του γράφου (μέγιστο ελάχιστο μονοπάτι).

Αν ορίσουμε

$$\overline{D_V(c)} = \frac{1}{2}\sum_{i,j\in c} \overline{D_V(i,j)}$$

το άθροισμα των αναμενώμενων αποστάσεων των κόμβων στην κοινότητα C, τότε η συνάρτηση:

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

δηλώνει την επιθυμητή σύγκριση του μέτρου στενής σύνδεσης των κοινοτήτων C ως προς του τυχαίου γράφου.

Περιμένουμε οι τιμές της Q_d να είναι υψηλές για ισχυρές δομές κοινοτήτων.

Στην υλοποίηση μου διάλεξα επίσης να τροποποιήσω την συνάρτηση ως εξίς:

$$D_V(c) = \sum_{i,j\in c} D_V(i,j)$$

$$\overline{D_V(c)} = \sum_{i,j\in c} \overline{D_V(i,j)}$$

$$Q_d(G,C) = \sum_{c\in C}[(1-\gamma)\overline{D_V(c)} - \gamma D_V(c)], \quad \gamma \in (0,1)$$

ώστε για $\gamma = 0.5$ να ισοδυναμεί με τον προηγούμενο ορισμό, αλλά να έχω δυνατότητα να επιλέξω ποιός όρος θα επηρρεάσει περισσότερο.

2.2 Υλοποίηση

Χρησιμοποιήθηκε δυναμική προσέγγιση, δηλαδή οι τιμές του αλγορίθμου που χρησιμοποιούνται συχνά ($D_V(i,j)$, $d_k(i)$, $m_G(k)$, $Pr[i,j,k]$, $\overline{D_V(i,j)}$, ..) υπολογίζονται μία φορά στην αρχή και αποθηκεύονται ώστε να καλούνται σε σταθερό χρόνο στην πορεία. Έτσι γλυτώνουμε τις άσκοπες επαναλύψεις. Ειδικότερα στην περίπτωση του συγκεκριμένου αλγορίθμου, αφού η συνάρτηση ποιότητας θα υπολογιστεί χιλιάδες φορές για διαφορετικές επιλογές κοινοτήτων στην διαδικασία μεγιστοποίησής της, και εφόσον οι μακράν πιο αργές διαδικασίες (όπως θα περιγραφτεί παρακάτω) είναι αυτές των προαναφερθέντων ποσοτήτων, η διαφορά στο χρόνο εκτέλεσης είναι ραγδαία.

Όπου είναι δυνατό χρησιμοποιούνται συναρτήσεις της βιβλιοθήκης $networkx$ μιας και είναι βελτιστοποιημένες και θα συμβάλουν στην ταχύτητα του κώδικα.

Στην πρώτη φάση του αλγορίθμου υπολογίζονται οι παρακάτω ποσότητες και αποθηκεύονται για μελλοντική χρήση:

- **διάμετρος του γράφου**

  Χρησιμοποιείται η συνάρτηση της βιβλιοθήκης $networkx$: $diameter(G)$.

- $\mathbf{D_V(i,j)}$

  Χρησιμοποιείται η συνάρτηση της βιβλιοθήκης $networkx$: $all\_pairs\_shortest\_path\_length(G)$, η οποία επιστρέφει απευθείας το ζητούμενο, την απόσταση κάθε δύο κόμβων του γράφου μεταξύ τους. Τα στοιχεία αποθηκεύονται σε ένα $python\ dict$.

- $\mathbf{d_k(i)}$ και $\mathbf{m_G(k)}$

  Χρησιμοποιείται η συνάρτηση της βιβλιοθήκης $networkx$: $singe\_source\_shortest\_path\_length(G, source)$, η οποία υπολογίζει για κάποιον κόμβο $source$ όλα τα ελάχιστα μονοπάτια που τον έχουν σαν άκρο. Αν υπολογιστούν αυτά τα μονοπάτια για κάθε κόμβο ως $source$ τότε θα έχουμε υπολογίσει όλα τα ελάχιστα μονοπάτια του γράφου δύο φορές (μία για $i \rightsquigarrow j$ και μια για $j \rightsquigarrow i$).

  Τρέχουμε την παραπάνω συνάρτηση για κάθε κόμβο του γράφου ($source$). Στο $dict$ που αυτή επιστρέφει, περιέχονται για κάθε άλλο κόμβο ($target$) μια λίστα με μήκη από όλα τα μονοπάτια που τους ενώνουν ($source \rightsquigarrow target$). Έτσι για κάθε μονοπάτι που βρίσκουμε στη λίστα, έχουμε το μήκος του k, αυξάνουμε κατά 1 τον βαθμό $d_k(source)$ και κατά 0.5 τη τιμή $m_G(k)$. Χρησιμοποιούμε 0.5 διότι $m_k(G) = \frac{1}{2}\sum_{v \in V(G)} d_k(v)$ (μετράμε κάθε μονοπάτι 2 φορές).

- $\mathbf{Pr[i,j,k]}$ και $\overline{\mathbf{D_V(i,j)}}$

  Υπολογίζονται απλά με τους τύπους:

$$Pr[i,j,k] = \frac{d_k(i)}{2m_k(G)} \frac{d_k(j)}{2m_k(G)}$$

$$\overline{D_V(i,j)} = \sum_{k=1}^{diam(G)} kPr[i,j,k]$$

  και αποθηκεύονται.

  Συγκεκριμένα οι πίνακες αυτοί είναι συμμετρικοί (αφού $\frac{d_k(i)}{2m_k(G)} \frac{d_k(j)}{2m_k(G)} = \frac{d_k(j)}{2m_k(G)} \frac{d_k(i)}{2m_k(G)}$) οπότε οι τιμές αυτές υπολογίζονται μία φορά για κάθε ζευγάρι i, j και προστίθονται στο τέλος μία φορά για $i = j$ και δύο για $i \neq j$.

Προσοχή χρειάζεται το γεγονός ότι οι παραπάνω διαδικασίες δε μπορούν να λειτουργήσουν για μη συνδεδεμένους γράφους. Έτσι υιοθετούμε τη σύμβαση ότι δύο μη συνδεδεμές συνεκτικές συνιστώσες δεν μπορούν να βρίσκονται στην ίδια κοινότητα. Όποτε διαχωρίζουμε τον γράφο στις συνεκτικές του συνιστώσες και τρέχουμε τα παραπάνω για κάθε μία ξεχωριστά. Αυτή η προσέγγιση διατηρεί την πολυπλοκότητα ίδια (αφού οι συνεκτικές συνιστώσες διαμερίζουν τον γράφο).

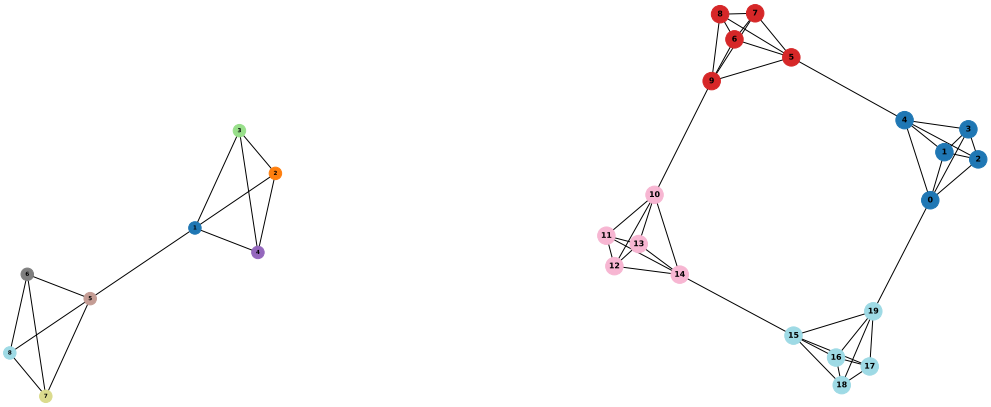
Στη δεύτερη φάση του αλγορίθμου, αφού έχουν αποθηκευτεί τα παραπάνω, υπολογίζονται για τις δοσμένες κοινότητες C τα αθροίσματα:

$$D_V(C) = \sum_{i,j \in C} D_V(i,j), \qquad \overline{D_V(C)} = \sum_{i,j \in C} \overline{D_V(i,j)}$$

για $i, j \in C$, και η τιμή:

$$Q_d(G,C) = \sum_{c \in C} [(1-\gamma)\overline{D_V(C)} - \gamma D_V(C)]$$

Συνολική πολυπλοκότητα της δεύτερης φάσης: $O(|V|^2)$.



(α′) Αποτελέσματα ωμής δύναμης για $\gamma = 0.5$. Κάθε κόμβος βρίσκεται σε δική του κοινότητα.

(β′) Αποτελέσματα αλγορίθμου Newman για $\gamma = 0.02$. Οι κοινότητες έχουν αναγνωριστεί σωστά.

Σχήμα 1

2.2.1 Μια πρώτη ματία στα αποτελέσματα

Για μια πρώτη εικόνα της λειτουργίας της παραπάνω μετρηκής, δοκιμάστηκε τεχνική ωμής δύναμης σε διάφορους γράφους, για $\gamma = 0.5$ (δηλαδή τον αρχικό ορισμό της). Ένα παράδειγμα βρίσκεται στο σχήμα 1α′, όπου κάθε κοινότητα χρωματίζεται με δικό της χρώμα.

Στο σχήμα φαίνονται οι κοινότητες για τις οποίες η τιμή Q_d είναι μέγιστη. Ενώ περιμέναμε για τον γράφο αυτό να επιστραφούν δύο κοινότητες των 4 κόμβων, βλέπουμε ότι αυτό δεν συμβαίνει. Συγκεκριμένα για $\gamma = 0.5$ βρέθηκε πειραματικά ότι τη μέγιστη τιμή Q_d παίρνουμε πάντα όταν κάθε κόμβος έχει δική του κοινότητα. Σε αυτό θα χρησιμεύσει η τιμή γ, όπου με την αλλαγή της μπορούμε να έχουμε τη επιθυμητή συμπεριφορά (βλ. ενότητα 3.4).

Παρατίθεται προκαταβολικά στο σχήμα 1β′ το αποτέλεσμα του αλγορίθμου Newman για $\gamma = 0.02$ για να φανεί το γεγονός αυτό.

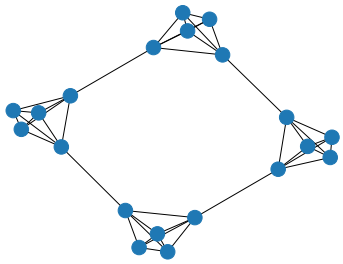
3 Αλγόριθμοι μεγιστοποίησης του Q_d

3.1 Πρώτη προσπάθεια άπληστου αλγορίθμου

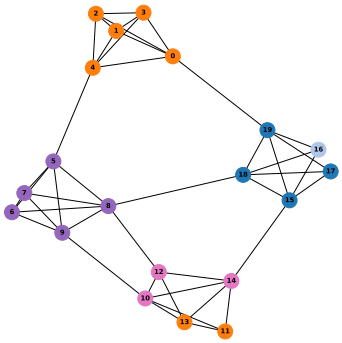
Θα περιγράψω τον πρώτο αλγόριθμο που υλοποιήθηκε συνοπτικά, διότι παρότι αντικαταστάθηκε από αυτόν του Newman, παρουσιάζει κάποια διαφορά στην λογική και τα αποτελέσματα:

1. Υπολογίζουμε τις απαραίτητες ποσότητες και τις αποθηκεύουμε $(D_V(i,j),\ d_k(i),\ m_G(k),\ Pr[i,j,k],\ \overline{D_V(i,j)}, diam(G))$.
2. Αρχικοποιούμε κάθε κόμβο σε δική του κοινότητα.
3. Για κάθε κόμβο, βρίσκουμε τον γειτονικό του κόμβο στου οποίου την κοινότητα εάν μεταφερθεί θα υπάρχει η μέγιστη μεταβολή στο Q_d. Εάν η μεταβολή αυτή είναι θετική, τότε τον μεταφέρουμε σε εκείνη την κοινότητα.
4. Επαναλαμβάνουμε το βήμα 3 μέχρι να μην γίνεται καμία μεταβολή.

Ο αλγορίθμος αυτός δέχεται βελτιστοποίηση, ειδικότερα στο κομμάτι όπου υπολογίζει την ποσότητα Q_d για κάθε μεταβολή τον κόμβων σε κοινότητες. Ένα μεγάλο μέρος της εξίσωσης $Q_d(G,C) = \sum_{c\in C}[(1-\gamma)\overline{D_V(c)} - \gamma D_V(c)]$ παραμένει σταθερό όταν μετακινούμε μόνο έναν κόμβο

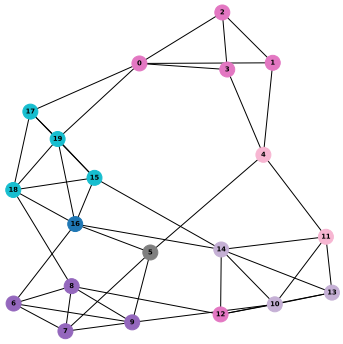


(α΄) Αρχικός γράφος benchmark

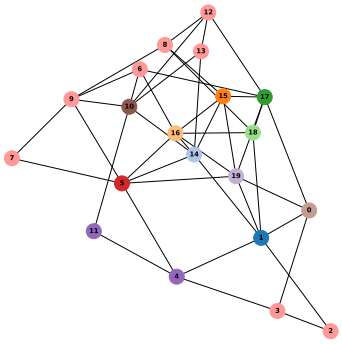


(β΄) 2ο βήμα

Σχήμα 2



(α΄) 7ο βήμα



(β΄) 16ο βήμα

Σχήμα 3

και κρατάμε τις υπόλοιπες κοινότητες σταθερές, και το γεγονός αυτό θα εκμεταλευτούμε στη συνέχεια.

Επίσης εδώ τίθεται το ζήτημα της σύγκλισης, εάν ο αλγόριθμος συγκλίνει (εξαρτάται από τη τιμή του γ) και πόσο γρήγορα, το οποίο λύνεται στην συνέχεια.

3.1.1 Απόδοση

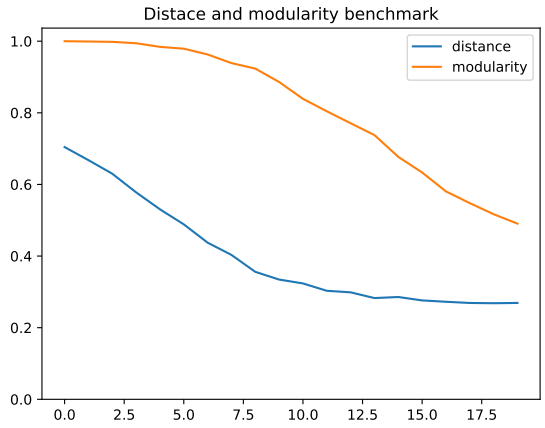
Θα περιγραφτεί εδώ η διαδικασία με την οποία μετρήθηκε η απόδοση του αλγορίθμου, η οποία θα χρησιμοποιηθεί και στη συνέχεια για τον αλγόριθμο του Newman.

Αρχικά δημιουργείται ένας γράφος ο οποίος αποτελείται από 4 πλήρεις υπογράφους των 5 κόμβων συνδεδεμένους με μία ακμή σε ένα κύκλο (σχήμα 2α΄). Σε αυτόν τον γράφο περιμένουμε οι κοινότητες να είναι οι τέσσερεις πλήρεις υπογράφοι.

Έπειτα σε κάθε βήμα αφαιρούμε μία ακμή (τυχαία) από το εσωτερικό των κοινοτήτων και την τοποθετούμε στο εξωτερικό τους (δηλαδή να ενώνει δύο διαφορετικές κοινότητες), έτσι 'χαλάμε' σε κάθε βήμα την ποιότητα των κοινοτήτων. Για κάθε βήμα εκτιμούμε με τον αλγόριθμό μας τις κοινότητες και τις συγκρίνουμε με τις πραγματικές (Jaccard similarity).

Στα σχήματα 2β΄, 3α΄, 3β΄ βλέπουμε τις εκτιμήσεις του αλγορίθμου για βήματα 2,7,16. Βλέπουμε ότι ο αλγόριθμος δεν έχει πολύ καλή απόδοση (ειδικά στο βήμα 1 όπου οι κοινότητες θα έπρεπε να είναι ξεκάθαρες), αλλά καταφέρνει να ανιχνεύσει κάποια δομή. Στο βήμα 16 όπου η δομή κοινοτήτων δεν είναι καθόλου ξεκάθαρη, τα αποτελέσματα είναι αρκετά κακά.

Στο σχήμα 4 αποτυπώνουμε την απόδοση (Jaccard similarity) του αλγορίθμου μας και την αντίστοιχη της modularity για 20 βήματα. Για να εξαλειφθεί ο τυχόν θόρυβος από τις τυχαίες



Σχήμα 4: Μέση απόδοση με Jaccard similarity των δύο μετρηκών.
Στον οριζόντιο άξονα τα βήματα της διαδικασίας φθοράς του γράφου, στον κάθετο η απόδοση.

διαδικασίες πάρθηκε ο μέσος όρος για 100 ίδια πειράματα.

Για τη modularity χρησιμοποιήθηκε η συνάρτηση *greedy_modularity_communities* της βιβλιοθήκης networkx, η οποία λειτουργεί με τον αλγόριθμο των Clauset-Newman-Moore [1].

Η distance έχει σαφώς χειρότερα αποτελέσματα.

3.2 Newman

Για καλύτερα αποτελέσματα υλοποιήθηκε ο αλγόριθμος που προτάθηκε από τον Newman [3].

Ο αγλόριθμος αυτός προτάθηκε για τη μεγιστοποίηση της modularity, αλλά μπορεί με τις κατάλληλες τροποποιήσεις να χρησιμοποιηθεί και εδώ.

Ξεκινάμε με μια κοινότητα για κάθε κόμβο αλλά αυτή τη φορά ενώνουμε διαδοχικά *τις κοινότητες* οι οποίες θα έχουν μέγιστη αύξηση του Q. Μετά από $|V|-1$ συνδέσεις θα έχει μείνει μόνο μία κοινότητα, όποτε απαιτούνται το πολύ $|V|-1$ συνδέσεις.

Στην ταχύτητα βοηθάει η παρατήρηση ότι δε χρειάζεται να υπολογίζεται συνέχεια η τιμή της Q_d αλλά μόνο η διαφορά $\Delta Q_d(c_i, c_j)$ της Q_d αν συνενώσουμε τις κοινότητες c_i και c_j.

Αναλυτικά:

Έστω ότι έχω κοινότητες $C = c_1, .., c_n$ και συνενώνω τις κοινότητες $c_i, c_j \in C$ ώστε να αποκτήσω στο επόμενο βήμα κοινότητες C'. Έστω επίσης ότι Q_d είναι η τιμή της distance quality στις κοινότητες C, ενώ Q'_d στις C'.

Για ευκολία συμβολίζω:

$$I(c) = [(1-\gamma)\overline{D_V(c)} - \gamma D_V(c)]$$

και,

$$J(i,j) = (1-\gamma)\overline{D_V(i,j)} - \gamma D_V(i,j)$$

τότε,

$$Q_d = \sum_{c \in C} I(c) = \sum_{c \in C} \sum_{i,j \in c} J(i,j)$$

Έχουμε:

$$Q_d = \sum_{c \in C} I(c) = \sum_{c \in C \setminus c_i, c_j} [I(c)] + I(c_i) + I(c_j) \Leftrightarrow$$

$$Q_d = \sum_{c \in C \setminus c_i, c_j} [I(c)] + \sum_{i,j \in c_i} J(i,j) + \sum_{i,j \in c_j} J(i,j)$$

και,

$$Q'_d = \sum_{c \in C'} I(c) = \sum_{c \in C \setminus c_i, c_j} [I(c)] + I(c_i \cup c_j) \Leftrightarrow$$

$$Q'_d = \sum_{c \in C \setminus c_i, c_j} [I(c)] + \sum_{i,j \in c_i \cup c_j} J(i,j)$$

Από τα παραπάνω είναι φανερό ότι:

$$\Delta Q = Q'_d - Q_d = 2 \sum_{i \in c_i, j \in c_j} J(i,j) \Leftrightarrow$$

$$\Delta Q = 2 \sum_{i \in c_i, j \in c_j} (1-\gamma)\overline{D_V(i,j)} - \gamma D_V(i,j)$$

Επιπλέον, όταν ενώνονται δύο κοινότητες c_i, c_j, έστω στην c_j, τότε για κάθε άλλη κοινότητα c_k η τιμή ΔQ_{jk} μεταβέλεται ως εξίς:

$$\Delta Q'_{jk} = 2 \sum_{i \in c_i \cup c_j, j \in c_k} J(i,j) = 2 \sum_{i \in c_i, j \in c_k} J(i,j) + 2 \sum_{i \in c_j, j \in c_k} J(i,j) = \Delta Q_{ik} + \Delta Q_{jk}$$

Τώρα είμαστε σε θέση να ορίσουμε τον αλγόριθμο:

Διατηρούμε έναν πίνακα ΔQ που θα περιέχει για κάθε δύο κοινότητες c_i, c_j τη τιμή ΔQ_{ij}. Για την αρχική περίπτωση όπου κάθε κόμβος v έχει δική του κοινότητα $c = \{v\}$:

$$\Delta Q_{ik} = 2 \cdot [(1-\gamma)\overline{D_V(c_i, c_j)} - \gamma D_V(c_i, c_j)] \tag{1}$$

1. Υπολογίζουμε τις απαραίτητες ποσότητες και τις αποθηκεύουμε $(D_V(i,j),\ d_k(i),\ m_G(k),\ Pr[i,j,k],\ \overline{D_V(i,j)}, diam(G))$.

2. Αρχικοποιούμε κάθε κόμβο σε δική του κοινότητα.

3. Αρχικοποιούμε τον πίνακα ΔQ σύμφωνα με την εξίσωση 1.

4. Βρίσκουμε το μέγιστο στοιχείο του πίνακα ΔQ. Αυτό το στοιχείο μας δίνει ποιές κοινότητες θα δώσουν τη μέγιστη διαφορά στην Q_d αν ενωθούν.

5. Αν το $max\Delta Q_{ij}$ που βρήκαμε είναι θετικό, ενώνουμε τις κοινότητες c_i, c_j στην c_j, ανανεώνουμε τα στοιχεία του πίνακα ΔQ ως εξίς:

   $$\Delta Q_{jk} = \Delta Q_{kj} = \Delta Q_{ik} + \Delta Q_{jk}, \quad \Delta Q_{ik} = \Delta Q_{ki} = -1$$

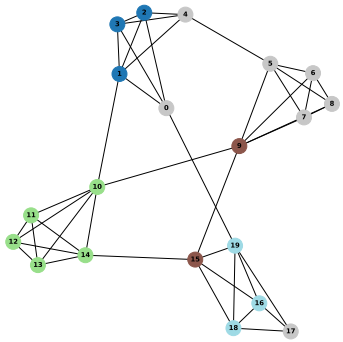
   και επιστρέφουμε στο βήμα 4.

6. Αν το $max\Delta Q_{ij}$ που βρήκαμε είναι αρνητικό, τότε τερματίζουμε τον αλγόριθμο και επιστρέφουμε τις κοινότητες όπως είναι σε αυτό το βήμα.

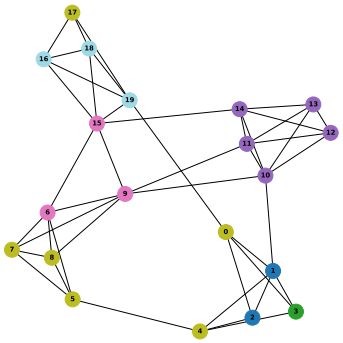
Σύμφωνα με την έρευνα των Clauset-Newman-Moore [1], από το βήμα όπου το $max\Delta Q$ γίνει για πρώτη φορά αρνητικό και μετέπειτα, θα συνεχήσει να είναι αρνητικό. Δηλαδή η Q_d θα μειώνεται σταδιακά. Επομένως μόλις φτάσουμε σε αυτό το στάδιο μπορούμε να τερματίσουμε τον αλγόριθμο, αφού βρήκαμε το μέγιστο.

Αυτός ο αλγόριθμος είναι ήδη σημαντικά πιο γρήγορος από τον προηγούμενο, αφού το εσωτερικό της λούπας πλέον απλοποιείται στην εύρεση μέγιστου στοιχείου ενός πίνακα. Τα υπόλοιπα είναι μόνο προσθέσεις, αναθέσεις πινάκων και έλεγχοι, όλες διεργασίες σταθερού χρόνου. Επίσης στον αλγόριθμο αυτόν δεν χρειάζεται να υπολογιστεί η Q_d αναλυτικά, αλλά μόνο τα αρχικά ΔQ που είναι σημαντικά απλούστερα.

Μια άμεση βελτίωση που θα μπορούσε να γίνει [1] είναι να τοποθετούμε τα ΔQ_{ij} σε μια σωρό μεγίστων, ώστε να αποφευχθεί το $O(|V|^2)$ της εύρεσης μεγίστου στον πίνακα σε $O(log|V|)$. Όμως αυτό δε το υλοποίησα στο πλαίσιο της παρούσης εργασίας.

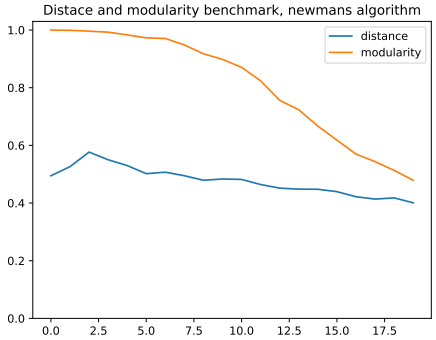


(α′) Αποτελέσματα αλγορίθμου Newman στο 2ο βήμα. ($\gamma = 0.15$)

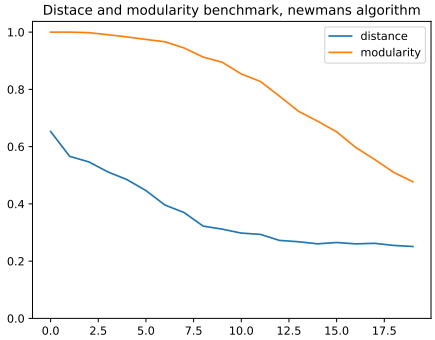


(β′) Αποτελέσματα αλγορίθμου Newman στο 4ο βήμα. ($\gamma = 0.15$)

Σχήμα 5



(α′) $\gamma = 0.01$



(β′) $\gamma = 0.02$

Σχήμα 6: Μέση απόδοση με χρήση Jaccard similarity, στη distance χρησιμοποιήθηκε αλγόριθμος Newman.

3.2.1 Απόδοση

Στα σχήματα 5, 6 φαίνονται τα αποτελέσματα του συγκεκριμένου αλγόριθμου με τις ίδιες τεχνικές όπως προηγουμένως. Ο αλγόριθμος Newman φαίνεται να έχει ίσως χειρότερα (ή αντίστοιχα) αποτελέσματα από/με τον προηγούμενο, το οποίο είναι αναμενόμενο αφού ενώνει κάθε φορά ολόκληρες κοινότητες αντί για μεμονομένους κόμβους και έτσι μια λάθος ένωση στην αρχή, διαδίδεται και πολλαπλασιάζεται.

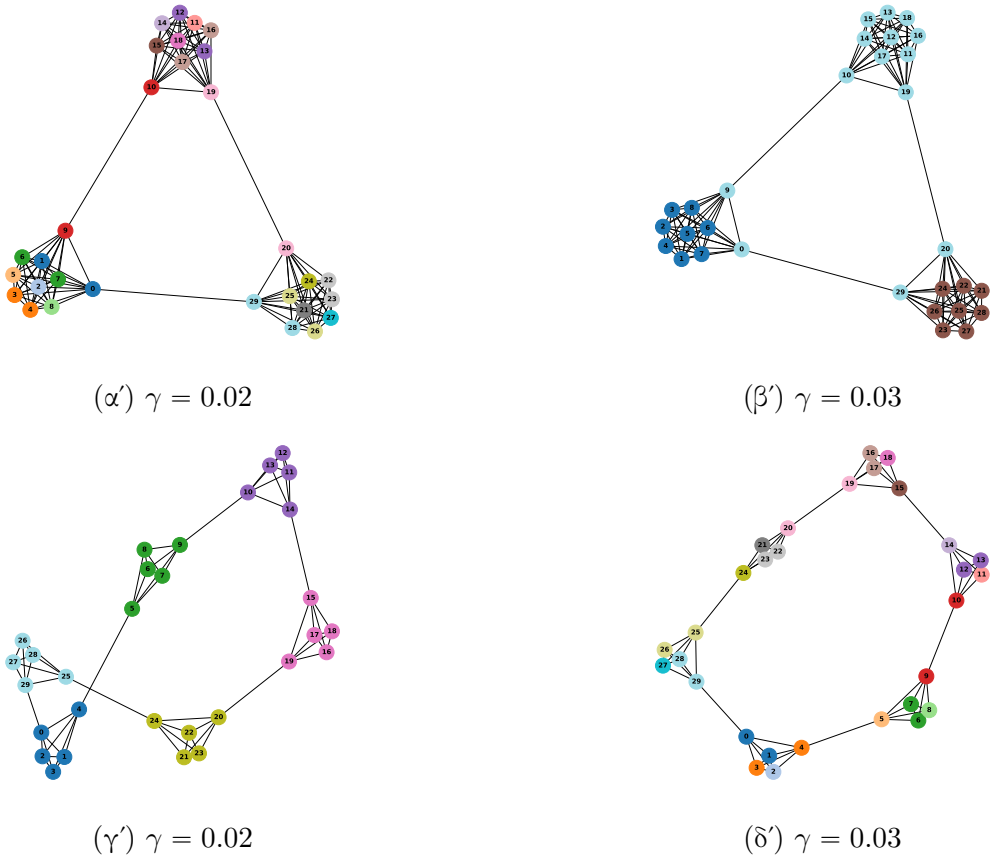
Το πλεονέκτιμά του είναι η ταχύτητα.

3.3 Χρονική πολυπλοκότητα και σύγκριση

Για τους δύο αλγορίθμους το πρώτο στάδιο είναι παρόμοιο (υπολογισμός απαραίτητων ποσοτήτων), δεν αναφέρω την χρονική πολυπλοκότητα του, μιας και δεν βρήκα τις χρονικές πολυπλοκότητες των συναρτήσεων της βιβλιοθήκης *networkx* που χρησιμοποίησα. Συγκεκριμένα στον αλγόριθμο *Newman* δε χρειάζονται όλες οι ποσότητες όποτε το στάδιο αυτό είναι πιο γρήγορο.

Ούτως ή άλλως, όμως, το αργό στάδιο είναι το δεύτερο, αυτό της βελτιστοποίησης της δομής κοινοτήτων.

- Στον πρώτο αλγόριθμο τρέχουμε για κάθε κόμβο και κάθε γείτονα του, την δεύτερη φάση του υπολογισμού της Q_d (βλ. 2.2), η οποία είναι $O(|V|^2)$. Επομένως έχουμε πολυπλοκότητα χειρότερης περίπτωσης: $O(|V|^4)$. Αυτή η διαδικασία όμως γίνεται όσες φορές χρειάζεται



(α′) $\gamma = 0.02$ (β′) $\gamma = 0.03$

(γ′) $\gamma = 0.02$ (δ′) $\gamma = 0.03$

Σχήμα 7: Αποτελέσματα γράφων για διαφορετικές τιμές του γ με τον αλγόριθμο Newman.

ώστε να μην βελτιώνεται άλλο η Q_d. Στον κώδικα έχει δωθεί ένα άνω φράγμα επαναλύψεων (max iterations = $maxit$), έτσι η πολυπλοκότητα χειρότερης περίπτωσης είναι συνολικά $O(maxit \cdot |V|^4)$, όπου $|V|$ το πλήθος των κόμβων.

- Στον αλγόριθμο *Newman* το εσωτερικό της λούπας, όπως ήδη αναφέρθηκε, είναι $O(|V|^2)$ για την εύρεση του μέγιστου στοιχείου του πίνακα ΔQ (το οποίο μπορεί και να βελτιωθεί με σωρό μεγίστων). Η λούπα επαναλαμβάνεται το πολύ $|V| - 1$ φορές, και έτσι έχουμε συνολοκή πολυπλοκότητα χειρότερης περίπτωσης: $O(|V|^3)$.

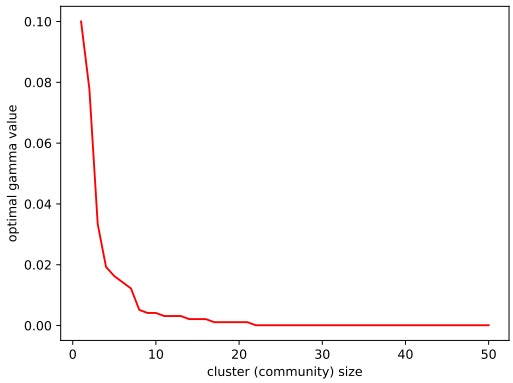
3.4 Επιρροή της τιμής γ

Στο σχήμα 7 φαίνοται τα αποτελέσματα δύο γράφων για διαφορετικές τιμές του γ. Βλέπουμε ότι ο γράφος στο 7α′ για $\gamma = 0.02$ δεν έχει καλά αποτελέσματα, ενώ στο 7β′ για $\gamma = 0.003$ έχει πολύ καλύτερα. Το αντίθετο ισχύει για το γράφο των 7γ′, 7δ′, όπου για $\gamma = 0.02$ έχει άρτια αποτελέσματα.

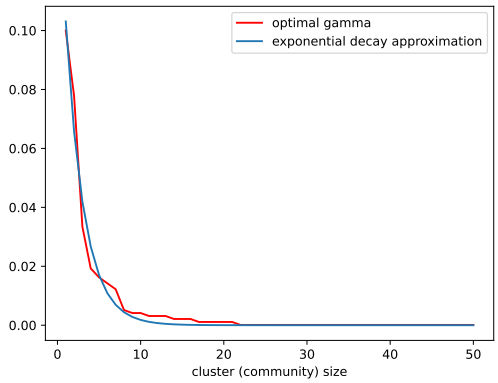
Έιναι ξεκάθαρη η επιρροή του γ όμως η βέλτιστη τιμή του είναι διαφορετική σε κάθε γράφο. Έτσι παρουσιάζεται το ερώτημα: *μπορούμε να εκτιμήσουμε τη βέλτιστη τιμή του γ;*

Μια παρατήρηση που κάνουμε εξ αρχής είναι ότι η βέλτιστη τιμή του γ φαίνεται να μικραίνει όσο μεγαλώνει το μέγεθος των κοινοτήτων. Έτσι δοκιμάστηκε να βρεθεί η σχέση τους, εάν αυτή υπάρχει: Δημιουργήθηκαν τυχαίοι γράφοι με την τεχνική που παρουσιάστηκε και προηγουμένως για διαφορετικά μεγέθη των κοινοτήτων (πλήρεις υπογράφοι). Για κάθε κάθε τέτοιο γράφο δοκιμάστηκαν πολλές τιμές γ και βρέθηκε η βέλτιστη από αυτές. Βρέθηκε ξεκάθαρη εξάρτηση της βέλτιστης τιμής γ και του μεγέθους κοινότητας.

Στο σχήμα 8α′ απεικονίζεται η εξάρτηση αυτή για μεγέθη κοινοτήτων απο 1 μέχρι 50 κόμβους. Παρατηρούμε εκθετική μείωση, οπότε δοκιμάστηκε να προσαρμωστεί η καμπύλη $ce^{-\lambda x}$ (δηλαδή να



(α′) Συσχέτηση βέλτιστου γ ως προς μέγεθος κοινοτήτων



(β′) Εκτίμηση της συσχέτησης με $ce^{-\lambda x}$, $c = 0.1616558536042243, \lambda = 0.4500547630497226$.

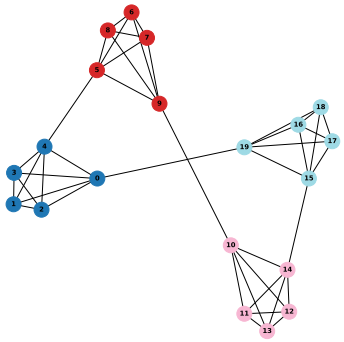
Σχήμα 8

βρεθούν σταθερές c, λ) στα δεδομένα. Χρησιμοποιήθηκε η συνάρτηση $scipy.optimise.curve\_fit$.

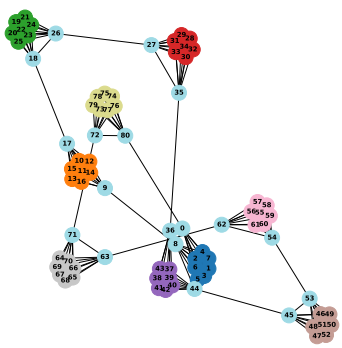
Το αποτέλεσμα φαίνεται στο σχήμα 8β′ για $c = 0.1616558536042243, \lambda = 0.4500547630497226$.

Έτσι δημιουργήθηκε μια παραλαγή του αλγορίθμου $Newman$, όπου αυτή τη φορά το γ δε δίνεται ως είσοδος, αλλά υπολογίζεται με την σχέση $ce^{-\lambda x}$, όπου x το αναμενόμενο μέγεθος κοινοτήτων.

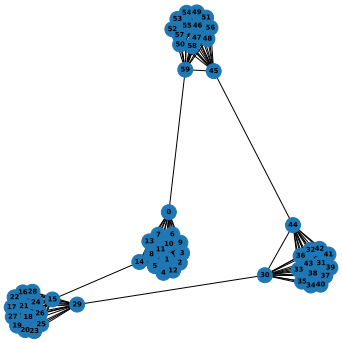
Αποτελέσματα της μεθόδου αυτής φαίνονται στο σχήμα 9. Στα παραδείγματα 9α′, 9β′ κατάφερε με ακρίβεια να αναγνωρίσει τις κοινότητες, όμως στο 9γ′ απέτυχε. Ίσως με περισσότερα δεδομένα να μπορούσε να εκτιμηθεί καλύτερα η καμπύλη της συσχέτησης του βέλτιστου γ και να είχαμε καλύτερα αποτελέσματα.



(α′) 4 κοινότητες από 5 κόμβους.



(β′) 9 κοινότητες από 9 κόμβους βους



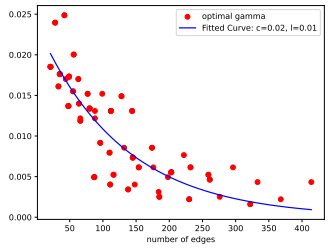
(γ′) 4 κοινότητες από 15 κόμβους.

Σχήμα 9: Αλγόριθμος $Newman$ με εκτίμηση γ από το μέγεθος των κοινοτήτων

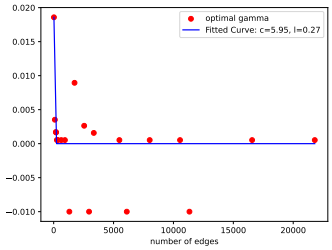
Όμως το πρόβλημα έτσι δεν λύνεται: θα θέλαμε ο αλγόριθμος να μπορεί να αναγνωρίσει τις κοινότητες χωρίς να χρειάζεται να γνωρίζουμε ήδη το μέγεθός τους.

Μια δεύτερη παρατήρηση ήταν ότι, ενώ το βέλτιστο γ μπορεί να αλλάξει για γράφο με ίδιο πλήθος κόμβων (παράδειγμα στο σχήμα 12β′), φαίνεται να μένει σταθερή αν σταθεροποιήσουμε και το πόσες κοινότητες θα έχει. Έτσι υποθέτω ότι ίσως υπάρχει συσχέτηση με το πλήθος *ακμών* του γράφου, το οποίο εξ′ άλλου γνωρίζουμε εκ των προτέρων.

Οπότε για διάφορους γράφους απεικόνησα το βέλτιστο γ ως προς το πλήθος ακμών και φάνηκε πάλι να υπάρχει συσχέτηση εκθετικής μείωσης.



(α′)



(β′)

Στο σχήμα 10α′ φαίνονται αυτά τα δεδομένα όπως και η εκτίμηση τους με την ίδια τεχνική όπως προηγουμένως.

Στο σχήμα 10β′, για να αντιπροσωπεύει καλύτερα ένα πραγματικό δίκτυο, τοποθετείται 'θόρυβος' στο γράφο, με τη μορφή ακμών που διαγράφονται από το εσωτερικό των κοινοτήτων και προστίθονται στο εξωτερικό τους. Έτσι μοιάζουν περισσότερο με τους γράφους του σχήματος 3α′ παρά αυτούς του σχήματος 9β′. Δοκιμάστηκαν, εν τέλη, γράφοι με μεγέθη από 12 κόμβους μέχρι 1500, κοινότητες από 3 μέχρι 50 σε πλήθος και από 4 μέχρι 30 σε μέγεθος. Σε κάθε γράφο οι ακμές που τοποθετούνταν τυχαία από το εσωτερικό των κοινοτήτων στο εξωτερικό τους ήταν $|E|/3$ σε πλήθος. Βλέπουμε (όπως ήταν και αναμενόμενο) μεγαλύτερη διασπορά στους τυχαίους γράφους.

Έτσι δημιουργήθηκε μία συνάρτηση $newman\_greedy\_distance\_auto(G)$ η οποία πλέον δέχεται ως όρισμα *μόνο* τον γράφο G. Υπολογίζει την αναμενώμενη τιμή του γ σύμφωνα με την εξίσωση $ce^{-\lambda|E|}$, όπου $|E|$ το πλήθος ακμών του G, $c = 5.954172813613528$, $\lambda = 0.2747453041873875$, και εκτελεί τον αλγόριθμο $Newman$ για αυτό το γ.

Λόγω της διασποράς στο σχήματα δεν περιμένουμε να έχει πάντα το καλύτερο αποτέλεσμα.

Αποτελέσματα της μεθόδου αυτής βρίσκονται στο σχήμα 11 για διάφορα παραδείγματα.

3.5 Ανικανότητες των τεχνικών αξιολόγησης

Για την αξιολόγηση του Q_d ως μετρηκή για αναγνώρηση κοινοτήτων χρησιμοποιήσαμε μόνο γράφους των οποίων οι κοινότητες είχαν ίδιο μέγεθος, και των οποίων οι κοινότητες είναι στενά συνδεδεμένες και πολύ αποσυνδεδεμένες μεταξύ τους. Τέτοια δίκτυα δύσκολα παρατηρούνται στον πραγματικό κόσμο.

Υπάρχουν πιο ρεαλιστικές τεχνικές αξιολόγησης [2] οι οποίες δεν ήταν στα πλαίσια της παρούσης εργασίας.

Η μετρηκή Q_d (distance quiality) δεν φαίνεται να λειτουργεί καλά για γράφους των οποίων οι κοινότητες δεν έχουν ίδια μεγέθη. Ένα παράδειγμα φαίνεται στο σχήμα 12α′.

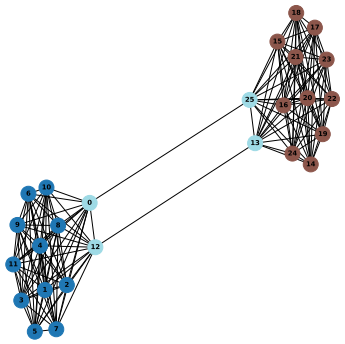
Είναι πιθανό, τέλος, επειδή τα δεδομένα συσχέτησης του βέλτιστου γ με το πλήθος ακμών παράχθηκαν με τέτοιους γράφους, η συσχέτηση αυτή να μην ανταποκρίνεται στην πραγματικότητα.

4 Πραγματικός γράφος

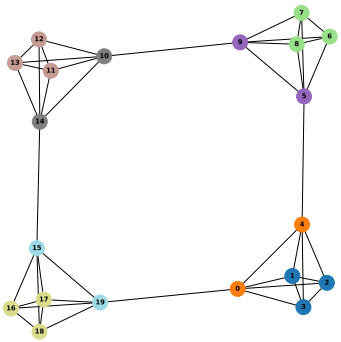
Θε ελέγξουμε, επιτέλους, την απόδοση της Q_d σε έναν πραγματικό δίκτυο. Χρησιμοποιήθηκε το δίκτυο email-Eu-core network απο το Stanford Network Analysis Project, το οποίο αντιπροσωπεύει επικοινωνία δύο μελών ενός Ευρωπαϊκού ινστιτούτου έρευνας μέσω e-mail. Μας δίνονται επίσης οι πραγματικές κοινότητες (42 τμήματα στα οποία ανήκουν).

Στο σχήμα ;; αποτυπώνεται το δίκτυο και οι σημαντικότερες κοινότητες του χρωματίζονται.

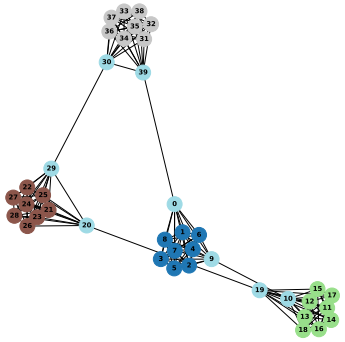
Είναι κατευθυνόμενο, όμως όλα τα παραπάνω λειτουργούν και για κατευθυνόμενους γράφους (;;;;).



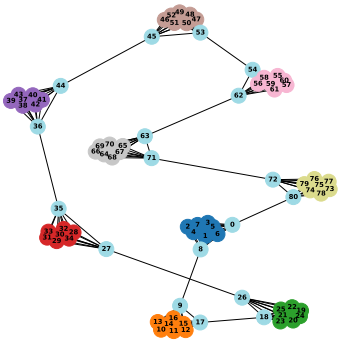
(α′) 2 κοινότητες 13 κόμβων



(β′) 4 κοινότητες 5 κόμβων

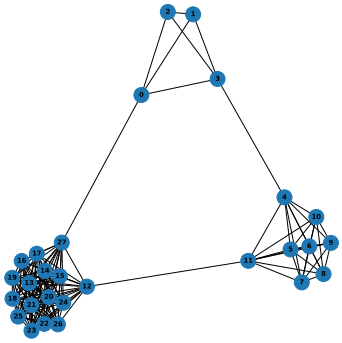


(γ′) 4 κοινότητες 10 κόμβων

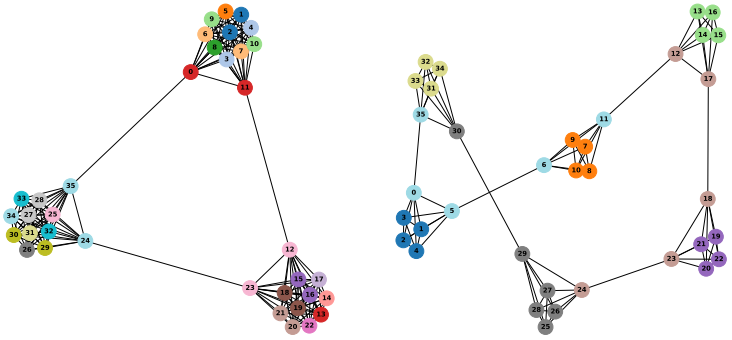


(δ′) 9 κοινότητες 9 κόμβων

Σχήμα 11: Αποτελέσματα τελικής μεθόδου, όπου το γ εκτιμάται από το πλήθος ακμών, για διάφορους γράφους.



(α′) Γράφος με κοινότητες διαφορετικών μεγεθών.



(β′) Οι δύο γράφοι έχουν το ίδιο μέγεθος (πλήθος κόμβων), όμως με την ίδια τιμή $\gamma = 0.01$ στον πρώτο δεν αναγνωρίζονται οι κοινότητες ενώ στον δεύτερο γίνεται μια καλή προσπάθεια. Δεν υπάρχει επομένως συσχέτηση μόνο με το μέγεθος του γράφου, αλλά παίζει ρόλο και το πλήθος κοινοτήτων.

Σχήμα 12

5 Συμπεράσματα

ρανδομ ωαλκ

6 Σημειώσεις περί του κώδικα

Κάποια αρχεία κώδικα *python* είναι προχειρογραμένα διότι χρησιμοποιήθηκαν μόνο για τον έλεγχο και την παραγωγή γραφημάτων. Δε θεώρησα απαραίτητο να τα καθαρογράψω αφού δεν αποτελούν τον πηγαίο κώδικα των υλοποιήσεων των συναρτήσεων και αλγορίθμων που παρουσιάζονται στην εργασία (και λόγω χρόνου). Θα μπορούσα ίσως να μην τα συμπεριλάβω στην υποβολή καθόλου, και να δώσω τα γραφήματα στην αναφορά, όμως θεώρησα χρήσιμο να μπορείτε να τα τρέξετε και να λάβεται τα ίδια γραφήματα. Δεν θα παραχθούν όλα τα γραφήματα που χρησιμοποίησα γιατί κάποια παράχθηκαν σε παλίες κλήσεις, οι οποίες έχουν διαγραφτεί. Θα παραχθούν και πολλά εξτρά.

Στον φάκελο υπάρχει ένα *README.md* αρχείο, στο οποίο αναγράφεται πλήρης αναφορά του περιεχομένου των αρχείων *.py* και της χρησιμότητας του καθενός.

Το Github repo της εργασίας: https://github.com/frilip/Network-theory.git.

Αναφορές

[1] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, December 2004.

[2] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008.

[3] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.