ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τεχνολογίες Γραφημάτων και Εφαρμογές

Ιωάννης Μαυροδήμος - it22068@hua.gr
Μεϊντάνης Χρήστος - it22069@hua.gr

Περιεχόμενα

1. Υπολογισμός πίνακα Z

1.1 Ανάλυση αλγορίθμου serial_algorithm.py

Ακολουθεί η ανάλυση του αλγορίθμου που υπολογίζει τον πίνακα Z βάσει του δοθέντος πίνακα MatrixY και των παραμέτρων r, c, iterations και tolerance:

1. Αρχικοποίηση:
    - Δημιουργείται ο πίνακας Zprime με τυχαίες τιμές μεγέθους n×r, όπου n είναι ο αριθμός των γραμμών του MatrixY και r είναι ο αριθμός των στηλών του Z.
    - Ορίζεται η μεταβλητή t στο 1.
2. Επανάληψη:
    - Εκτελείται επαναληπτικά η διαδικασία μέχρι να επιτευχθεί σύγκλιση ή μέχρι να φτάσει τον αριθμό των επαναλήψεων iterations.
    - Εντός κάθε επανάληψης, δημιουργείται ένας νέος πίνακας Z, ο οποίος αρχικοποιείται με τιμές ίσες με τον Zprime.
    - Γίνεται διπλός βρόγχος επάνω στους δείκτες i και j του MatrixY.
        - Ενημερώνεται κάθε γραμμή του πίνακα Z με βάση τη συνάρτηση ενημέρωσης που περιλαμβάνει τη σταθερά cc και τις τιμές των MatrixY, Z[i] και Z[j].
3. Συνθήκη Τερματισμού:
    - Ελέγχεται η Frobenius νόρμα της διαφοράς μεταξύ του Z και του Zprime.
    - Αν η νόρμα είναι μικρότερη από το tolerance, ο αλγόριθμος σταματά.
4. Ενημέρωση:
    - Αν δεν έχει συγκλίνει, η Zprime ενημερώνεται με τον Z και η μεταβλητή t αυξάνεται κατά 1.

Ουσιαστικά, ο αλγόριθμος εκτελεί επαναληπτικά βήματα ενημέρωσης του πίνακα Z, με στόχο τη σύγκλιση προς έναν πίνακα Z που βελτιστοποιεί κάποιο κριτήριο που σχετίζεται με τον πίνακα MatrixYκαι τις παραμέτρους rr και cc. Οι τιμές iterations και tolerance χρησιμοποιούνται για τον έλεγχο του αριθμού των επαναλήψεων και τη σύγκλιση του αλγορίθμου.

1.2 Ανάλυση αλγορίθμου algorithm_networkx.py

Ο αλγόριθμος χρησιμοποιεί το πακέτο networkx σε Python. Ο στόχος του αλγορίθμου είναι να ενσωματώσει γράφους σε χώρους χαμηλής διάστασης, διατηρώντας τις δομικές ιδιότητες του γράφου.

Ακολουθεί η ανάλυση του κώδικα σε 4 βήματα:

1. Οι παράμετροι εισόδου είναι:
    - G: Ο γράφος που θέλουμε να ενσωματώσουμε.
    - r: Το πλήθος των χαρακτηριστικών για κάθε κόμβο (τα οποία ονομάζονται Z).
    - c: Σταθερά πολλαπλασιασμού για την ενημέρωση των ενσωματώσεων.
    - iterations: Ο μέγιστος αριθμός επαναλήψεων.
    - tolerance: Κατώτατο όριο για τη σύγκλιση του αλγορίθμου.
2. Αρχικοποίηση του μεγέθους του γράφου και του πίνακα Z_prime (τυχαίες τιμές).
3. Επαναληπτικά, επαναλαμβάνεται για έναν αριθμό επαναλήψεων (ή μέχρι να συγκλίνει ο αλγόριθμος) το εξής:

    a. Αντιγραφή του πίνακα Z_prime ως Z.

    b. Διάσχιση όλων των ζευγαριών κόμβων (i, j) και ενημέρωση του πίνακα Z[i] χρησιμοποιώντας τον κανόνα ενημέρωσης που περιέχεται στον εσωτερικό βρόγχο.

    c. Έλεγχος για σύγκλιση: Αν η Frobenius νόρμα της διαφοράς μεταξύ Z και Z_prime είναι μικρότερη από το καθορισμένο όριο τότε ο αλγόριθμος τερματίζει.

    d. Ανανέωση του βήματος.

4. Επιστροφή του τελικού Z.

Συνοψίζοντας, ο αλγόριθμος εκπαιδεύει τις ενσωματώσεις των κόμβων (πίνακα Z) λαμβάνοντας υπόψη τις συνδέσεις μεταξύ των κόμβων στον γράφο, με στόχο τη διατήρηση της δομής του γράφου σε έναν χώρο χαμηλής διάστασης

1.3 Επεξήγηση παραδειγμάτων 1 και 2

Στα παραδείγματα 1 και 2, δημιουργούνται δύο γράφοι χρησιμοποιώντας το πακέτο NetworkX, και στη συνέχεια, εφαρμόζεται ο αλγόριθμος compute_Z_networkx σε κάθε έναν από αυτούς τους γράφους για την εύρεση του πίνακα Z.

Ο πρώτος γράφος (G1) είναι ένας μη κατευθυνόμενος γράφος με τρεις κόμβους και δύο ακμές με βάρη. Ο δεύτερος γράφος (G2) είναι ένας κατευθυνόμενος γράφος με τέσσερις κόμβους και τρεις ακμές με βάρη.

Κατόπιν, εφαρμόζεται ο αλγόριθμος compute_Z_networkx σε κάθε γράφο, χρησιμοποιώντας τυχαίες τιμές για τις παραμέτρους r και c. Οι πίνακες Z1 και Z2 εκτυπώνται στο τέλος.

Παρατηρείστε ότι για την εμφάνιση των γράφων χρησιμοποιείται η βιβλιοθήκη Matplotlib, ενώ ο αλγόριθμος compute_Z_networkx είναι αυτός που παρουσιάστηκε προηγουμένως.

1.4 Επεξήγηση παραδειγμάτων 3 και 4 (Χρήση λίγο πιο σύνθετων γράφων)

Στα παραδείγματα 2 και 3 αναφέρονται δύο επιπλέον γράφοι (G3 και G4) και εφαρμόζεται ο αλγόριθμος compute_Z_networkx σε κάθε έναν από αυτούς τους γράφους.

1. Παράδειγμα 3 - Μη Κατευθυνόμενος Γράφος:
    - Δημιουργία ενός μη κατευθυνόμενου γράφου με τέσσερις κόμβους και τέσσερις ακμές με επισημασμένα βάρη.
    - Οι ακμές επισημαίνονται με διάφορα βάρη.
    - Εμφάνιση του γράφου χρησιμοποιώντας Matplotlib.
2. Παράδειγμα 4 - Κατευθυνόμενος Γράφος:
    - Δημιουργία ενός κατευθυνόμενου γράφου με πέντε κόμβους και πέντε ακμές με επισημασμένα βάρη.
    - Οι ακμές επισημαίνονται με διάφορα βάρη.
    - Εμφάνιση του γράφου με χρήση Matplotlib.
3. Εφαρμογή του Αλγορίθμου:
    - Καθορισμός τυχαίων τιμών για τις παραμέτρους r και c.
    - Εφαρμογή του αλγορίθμου compute_Z_networkx σε κάθε γράφο.
4. Εκτύπωση Αποτελεσμάτων:
    - Εκτύπωση των πινάκων Z που προκύπτουν από την εφαρμογή του αλγορίθμου για κάθε γράφο.

Όπως και προηγουμένως, η εμφάνιση των γράφων γίνεται με Matplotlib, και οι πίνακες Z3 και Z4 εκτυπώνται στο τέλος.

1.5 Εφαρμογή αλγορίθμου compute_Z_networkx σε πραγματικα δεδομένα

Επιλέξαμε να εξετάσουμε τον αλγόριθμο σε 3 dataset πραγματικών δεδομένων (networks).

Στον κώδικα αυτό,

- Καθορίζουμε τη διαδρομή του αρχείου με την μεταβλητή file_path.
- Χρησιμοποιούμε τη συνάρτηση nx.read_edgelist() της NetworkX για να διαβάσουμε το αρχείο δεδομένων και να δημιουργούμε ένα γράφο από αυτό.
- Ορίζουμε τις παραμέτρους r, c, iterations και tolerance για τη χρήση τους στη συνάρτηση compute_Z_networkx.
- Καλούμε τη συνάρτηση compute_Z_networkx με τον γράφο G και τις ορισμένες παραμέτρους r, c, iterations, και tolerance.
- Εκτυπώνουμε το αποτέλεσμα που επιστρέφει η συνάρτηση compute_Z_networkx.

Αρχικά επιλέξαμε να εξετάσουμε το dataset (*corporate_club_memberships.py*) όπου περιέχει πληροφορίες μελών εταιρικών στελεχών σε κοινωνικούς οργανισμούς όπως λέσχες και συμβούλια. Τα αριστερά nodes αντιπροσωπεύουν πρόσωπα και τα δεξιά nodes αντιπροσωπεύουν κοινωνικούς οργανισμούς. Το edge μεταξύ ενός ατόμου και μιας κοινωνικής οργάνωσης δείχνει ότι το άτομο είναι μέλος.

```
PS C:\Users\user> & "C:/Program Files/Python310/python.exe" "c:/Users/user/Downloads/Νέος φάκελος/corporate_club_memberships.py"
[[0.00573575 0.08073408 0.79367675]
 [0.06853802 0.92937733 0.23451705]
 [0.82855746 0.39020962 0.52530934]
 [0.84264783 0.33940554 0.81820615]
 [0.48692656 0.87860654 0.53623279]
 [0.96569582 0.33369977 0.96889422]
 [0.49937293 0.43606773 0.98838649]
 [0.35630208 0.70954905 0.34756024]
 [0.21308356 0.7959183  0.97810975]
 [0.05798245 0.61345045 0.80887409]
 [0.38091823 0.80632563 0.43890289]
 [0.52862963 0.37509297 0.00388947]
 [0.296945   0.76154763 0.8094451 ]
 [0.28431765 0.77519942 0.7698005 ]
 [0.2492747  0.95892634 0.05236329]
 [0.71647832 0.74407633 0.66404876]
 [0.09074512 0.98156853 0.82832047]
 [0.16556442 0.69218165 0.15879601]
 [0.65856498 0.9225997  0.68531957]
 [0.71856426 0.71299726 0.66025698]
 [0.72072224 0.31961701 0.60665684]
 [0.95531394 0.45944541 0.21624254]
 [0.90176988 0.49009453 0.15949128]
 [0.90918888 0.66439578 0.20707019]
 [0.40983019 0.44934658 0.80443991]]
Ο αλγόριθμος ολοκληρώθηκε με επιτυχία.
```

Το δευτερο dataset (*zebras.py*) που επιλέξαμε περιέχει αλληλεπιδράσεις μεταξύ 28 ζέβρων του Grévy (Equus grevyi) στην Κένυα. Ο ένας κόμβος αντιπροσωπεύει μια ζέβρα και ο άλλος μεταξύ δύο ζέβρες δείχνει ότι υπήρξε μια αλληλεπίδραση μεταξύ τους κατά τη διάρκεια της μελέτης.

```
PS C:\Users\user> & "C:/Program Files/Python310/python.exe" "c:/Users/user/Downloads/Νέος φάκελος/zebras.py"
[[0.29076585 0.93790825 0.38492331]
 [0.20089066 0.37928927 0.76740866]
 [0.54202467 0.88703852 0.500696  ]
 [0.16175043 0.7398084  0.76038075]
 [0.49385932 0.82614485 0.9644984 ]
 [0.78677312 0.18728612 0.00980977]
 [0.47967712 0.72363631 0.40117308]
 [0.68776968 0.52971423 0.6072981 ]
 [0.50835318 0.00232897 0.17544069]
 [0.26839641 0.58393269 0.42798646]
 [0.59385262 0.04788149 0.41470536]
 [0.75128992 0.06251614 0.35473211]
 [0.29313611 0.57627875 0.17482719]
 [0.81658535 0.72852129 0.83244834]
 [0.54435951 0.52109218 0.4412399 ]
 [0.00121217 0.09031385 0.83160577]
 [0.56606861 0.14479281 0.75051821]
 [0.74763249 0.44431553 0.52647628]
 [0.17917512 0.30358004 0.52619171]
 [0.26139029 0.00596561 0.00532304]
 [0.11117408 0.4180623  0.18566279]
 [0.33682361 0.84600323 0.50876365]
 [0.02868236 0.48497213 0.23487287]
 [0.31608854 0.9254425  0.77633917]
 [0.36598166 0.49727398 0.94635666]
 [0.54061197 0.6939833  0.16234083]
 [0.18386486 0.01190216 0.7749525 ]]
Ο αλγόριθμος ολοκληρώθηκε με επιτυχία.
```

Το τρίτο dataset (*google_data.py*) περέχει το δίκτυο υπερσυνδέσμων από σελίδες εντός των ιστότοπων της Google, π.χ. στο google.com

```
PS C:\Users\user> & "C:/Program Files/Python310/python.exe" "c:/Users/user/Downloads/Νέος φάκελος/google_data.py"
[[0.38160888 0.28542504 0.06817061]
 [0.23054762 0.92448915 0.24069585]
 [0.4233381  0.94797321 0.45629225]
 ...
 [0.20590839 0.43178919 0.94788599]
 [0.55452539 0.74339557 0.6380027 ]
 [0.34935696 0.11036359 0.8395111 ]]
Ο αλγόριθμος ολοκληρώθηκε με επιτυχία.
```

2. Εφαρμογές

2.1 Σε ποιά προβλήματα είναι χρήσιμη η τεχνική αυτή

Η τεχνική που περιγράφεται, η εκτίμηση του μητρώου επικοινωνίας σε δίκτυα (Stochastic Block Model - SBM), είναι χρήσιμη σε διάφορα προβλήματα στον τομέα της ανάλυσης δικτύων. Ανάμεσα σε αυτά τα προβλήματα περιλαμβάνονται:

1. Εντοπισμός Κοινοτήτων (Community Detection): Η SBM μπορεί να χρησιμοποιηθεί για τον εντοπισμό κοινοτήτων σε ένα δίκτυο. Κοινότητες αντιστοιχούν σε ομάδες κόμβων που έχουν στενούς δεσμούς μεταξύ τους.

2. Πρόβλημα Κατηγοριοποίησης (Clustering Problem): Η τεχνική SBM μπορεί να εφαρμοστεί για την ομαδοποίηση (κατηγοριοποίηση) κόμβων σε ένα δίκτυο βάσει των πιθανοτήτων επικοινωνίας.

3. Πρόβλημα Κατηγοριοποίησης Συνόλων Δεδομένων (Community Discovery in Multi-layer Networks): Η εκδοχή της SBM για πολυστρωματικά δίκτυα μπορεί να χρησιμοποιηθεί για την ανίχνευση κοινοτήτων σε δίκτυα που αποτελούνται από πολλαπλά επίπεδα.

4. Προβλήματα Εντοπισμού Ανωμαλιών (Anomaly Detection Problems): Μπορεί να χρησιμοποιηθεί για τον εντοπισμό ανωμαλιών σε δίκτυα μέσω της σύγκρισης των πραγματικών πιθανοτήτων επικοινωνίας με τις εκτιμώμενες.

5. Προβλήματα Πρόβλεψης Διασποράς Πληροφορίας (Information Spread Prediction Problems): Η ανίχνευση κοινοτήτων μπορεί να βοηθήσει στην πρόβλεψη του πώς η πληροφορία εξαπλώνεται σε ένα δίκτυο.

6. Προβλήματα Εντοπισμού Κλειδων Δικτύου (Network Key Player Identification): Η SBM μπορεί να χρησιμοποιηθεί για τον εντοπισμό των κύριων κόμβων (key players) σε ένα δίκτυο.

Αυτά είναι μερικά από τα πολλά προβλήματα που μπορεί να επιλύσει η τεχνική SBM στον τομέα της ανάλυσης δικτύων. Η εφαρμογή εξαρτάται από τη φύση των δεδομένων και του προβλήματος που θέλετε να αντιμετωπίσετε.

2.2 Ο αλγόριθμος Louvain

Ο αλγόριθμος **Louvain** είναι ένας αλγόριθμος κοινωνικής δικτύωσης που χρησιμοποιείται για τον εντοπισμό κοινοτήτων σε γράφους. Ο στόχος του αλγορίθμου είναι να βρει τη δομή της κοινωνικής οργάνωσης ενός δικτύου, διαχωρίζοντας τους κόμβους σε διάφορες κοινότητες ή ομάδες με βάση την αλληλεπίδραση τους.

Ο αλγόριθμος Louvain λειτουργεί με τον εξής τρόπο:

1. Αρχικά, ο κάθε κόμβος θεωρείται ως μία ξεχωριστή κοινότητα.
2. Στη συνέχεια, επιχειρεί να βελτιστοποιήσει ένα κριτήριο, το οποίο λέγεται "Louvain Modularity", που μετρά το πόσο καλά χωρίζονται οι κόμβοι σε κοινότητες.
3. Κατά τη διάρκεια κάθε επανάληψης, οι κόμβοι συγχωνεύονται σε "σούπερ-κόμβους" με σκοπό την αύξηση του modularity.
4. Η διαδικασία επαναλαμβάνεται μέχρις ότου δεν είναι δυνατή περαιτέρω βελτιστοποίηση του κριτηρίου modularity.

Ο αλγόριθμος Louvain είναι αποδοτικός και ευέλικτος, επιτρέποντας την εντοπισμό κοινοτήτων σε μεγάλους γράφους. Είναι ευρέως χρησιμοποιούμενος σε διάφορους τομείς, όπως η κοινωνική δικτύωση, η βιολογία, η πληροφορική και άλλοι τομείς που απαιτούν την ανάλυση δικτύων.

2.3 Partitioned_graph_algorithm

Ο **partitioned_graph_algorithm** αναφέρεται στη χρήση του αλγορίθμου γραφικής διαίρεσης Louvain για τον εντοπισμό κοινοτήτων σε ένα γράφο, καθώς και τον υπολογισμό του πίνακα Z με χρήση του αλγορίθμου compute_Z_networkx.

Ας αναλύσουμε τον κώδικα βήμα προς βήμα:

1. Φόρτωση του γράφου:
    - Ο γράφος φορτώνεται από ένα αρχείο edge list (file_path) χρησιμοποιώντας τη συνάρτηση nx.read_edgelist.
2. Εφαρμογή του Louvain για γραφική διαίρεση:
    - Ο αλγόριθμος Louvain εφαρμόζεται στον γράφο με τη χρήση της συνάρτησης louvain_graph_partitioning, η οποία επιστρέφει ένα λεξικό (partition) που αντιστοιχεί τους κόμβους σε κοινότητες.
3. Υπολογισμός του πίνακα Z:
    - Ο πίνακας Z υπολογίζεται με τη χρήση της συνάρτησης compute_Z_networkx.
    - Οι παράμετροι rr και cc καθορίζουν τον αριθμό των στηλών του πίνακα Z και μια σταθερά στην ενημέρωση των τιμών του.
4. Εκτύπωση των κοινοτήτων:
    - Οι κοινότητες εκτυπώνονται με βάση το λεξικό partition.
5. Σχεδίαση του γράφου με βάση τις κοινότητες:
    - Ο γράφος σχεδιάζεται με χρωματισμό των κόμβων ανά κοινότητα, χρησιμοποιώντας τη συνάρτηση plot_partitioned_graph.

Το αποτέλεσμα είναι η οπτικοποίηση του γράφου με τη χρήση του Louvain για την εύρεση κοινοτήτων, καθώς και η εκτύπωση των κοινοτήτων και ο πίνακας Z που υπολογίστηκε χρησιμοποιήσαμε το dataset από το πραγματικό γεγονός με τις Ζέβρες.

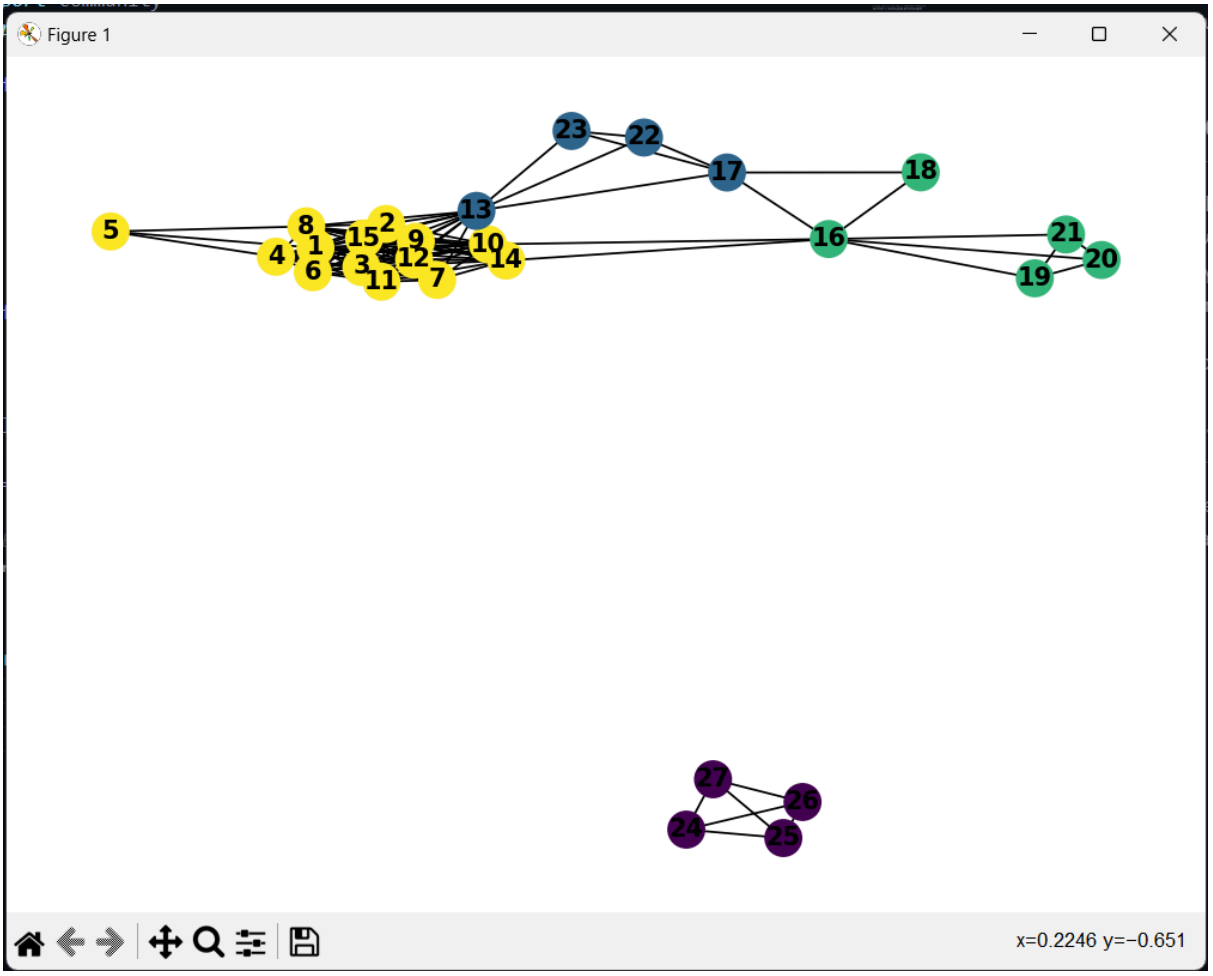
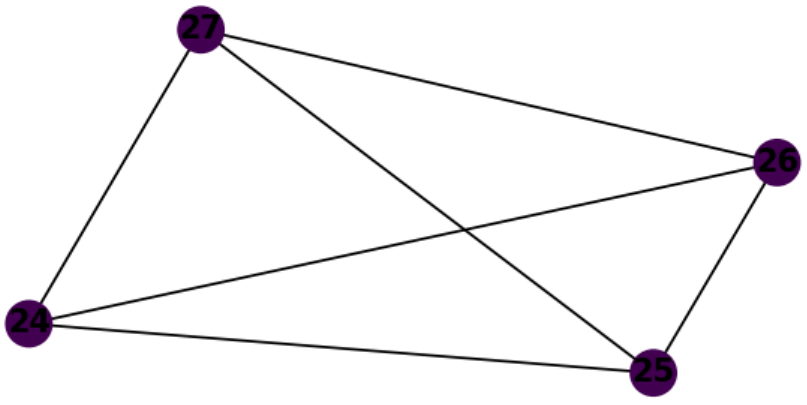
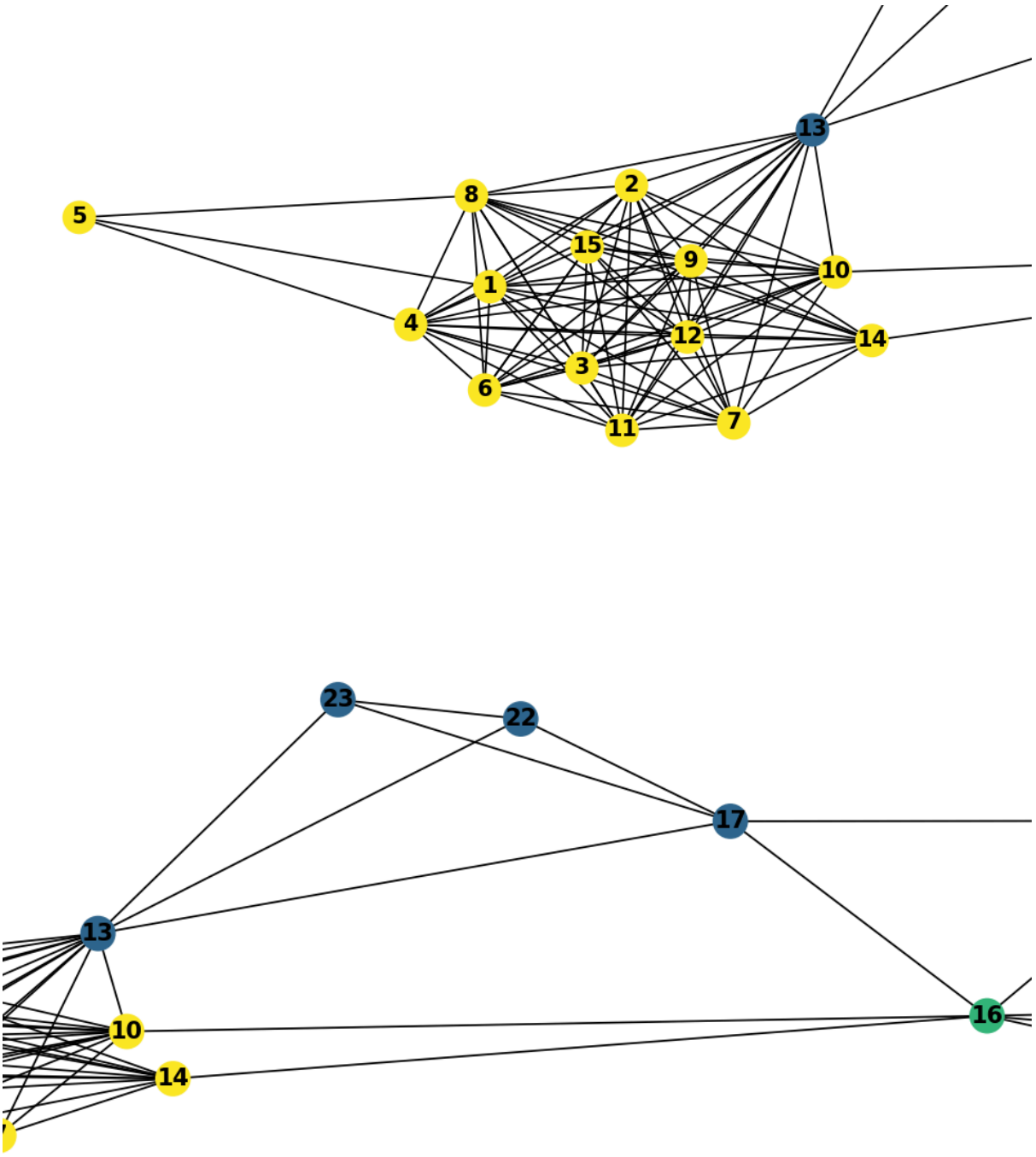
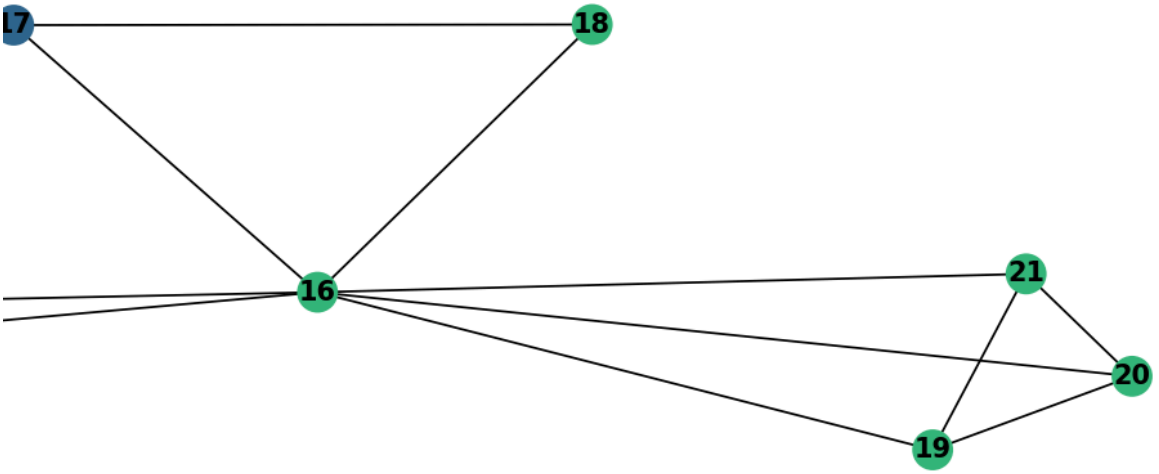
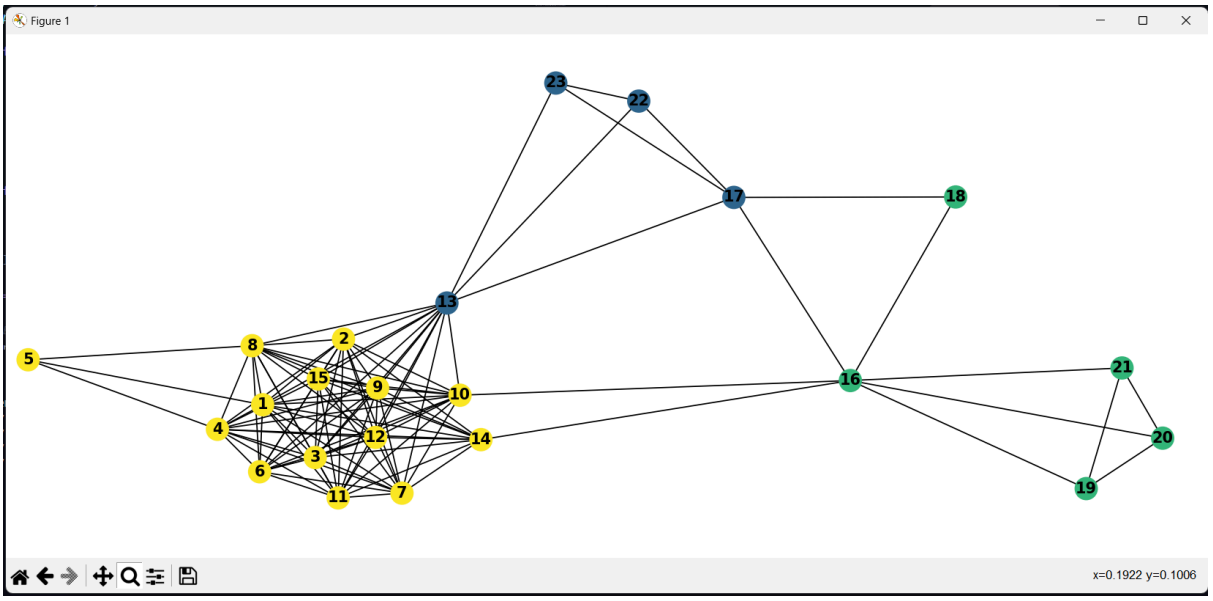


Figure 1
23
22
17
18
13
5
8
2
15
9
10
16
21
4
1
12
14
20
6
3
11
7
19
27
26
24
25
x=0.2246 y=−0.651


27
26
24
25


13
2
8
5
15
9
10
1
4
12
14
3
6
11
7
23
22
17
13
16
10
14






```
PS C:\Users\user> & "C:/Program Files/Python310/python.exe" "c:/Users/user/Downloads/Νέος φάκελος/test2_ex2_graphs.py"
Community 4: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '14', '15']
Community 2: ['13', '17', '22', '23']
Community 3: ['16', '18', '19', '20', '21']
Community 1: ['24', '25', '26', '27']
```