ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“Βιολογικά δίκτυα στη μελέτη ασθενειών πολυπαραγοντικής αιτιολογίας”

Εκπόνηση: Ιωάννης Χαρατσιάρης, Αρ. Μητρώου: 01996

Επιβλέπων: Καθηγητής Παντελεήμων Μπάγκος

Τριμελής Εξεταστική Επιτροπή:
Παντελεήμων Μπάγκος, Καθηγητής, Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Γεωργία Μπράλιου, Αναπληρώτρια Καθηγήτρια, Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Παναγιώτα Κοντού, Επίκουρη Καθηγήτρια, Τμήμα Μαθηματικών

ΛΑΜΙΑ, Οκτώβριος 2025

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1) , που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί χωρίς να τα περικλείω σε εισαγωγικά και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.

2. Δέχομαι ότι η αυτολεξεί παράθεση χωρίς εισαγωγικά, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ο – Η Δηλ. Ημερομηνία: 21/10/2025

Ιωάννης Χαρατσιάρης

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Περίληψη

Η ραγδαία ανάπτυξη των τεχνικών υψηλής απόδοσης (high-throughput) στη Βιολογία Συστημάτων έχει οδηγήσει στη συσσώρευση τεράστιου όγκου δεδομένων, καθιστώντας την ανάλυση δικτύων απαραίτητο εργαλείο για τη μελέτη πολυπαραγοντικών ασθενειών. Ωστόσο, τα διμερή δίκτυα (bipartite networks) γονιδίων-ασθενειών που προκύπτουν, καθώς και οι μονομερείς προβολές τους (projections), χαρακτηρίζονται συχνά από υψηλή πυκνότητα και θόρυβο, δυσχεραίνοντας την εξαγωγή βιολογικά σημαντικών συμπερασμάτων. Η παρούσα πτυχιακή εργασία εστιάζει στη συγκριτική αξιολόγηση και εφαρμογή αλγορίθμων εξαγωγής «κορμού» (network backbone extraction), με στόχο τον εντοπισμό των σημαντικότερων συσχετίσεων και την αφαίρεση των τυχαίων συνδέσεων.

Για την ανάλυση, δημιουργήθηκε ένα διμερές δίκτυο συσχέτισης γονιδίων-ασθενειών αντλώντας δεδομένα από τις βάσεις GWAS Catalog και OMIM. Εφαρμόστηκαν και συγκρίθηκαν διαφορετικές μέθοδοι φιλτραρίσματος, οι οποίες κατηγοριοποιούνται σε στατιστικές (π.χ. BiCM, SDSM, Polya Filter, Marginal Likelihood), δομικές (π.χ. MST, Edge Betweenness) και υβριδικές. Η αξιολόγηση πραγματοποιήθηκε βάσει τοπολογικών δεικτών, όπως η αρθρωτότητα (modularity), η διάμετρος και ο συντελεστής συσσωμάτωσης.

Τα αποτελέσματα ανέδειξαν σημαντικές διαφοροποιήσεις στη λειτουργία των αλγορίθμων: μέθοδοι όπως το Modularity Filter διατήρησαν πυκνές δομές κοινοτήτων, ιδανικές για τη μελέτη συν-νοσηροτήτων, ενώ μέθοδοι όπως το Polya Filter και το MST παρήγαγαν εξαιρετικά αραιούς, δενδρικούς σκελετούς, αναδεικνύοντας τις ιεραρχικές σχέσεις. Συμπερασματικά, η εργασία καταδεικνύει ότι δεν υφίσταται μία βέλτιστη μέθοδος για όλα τα σενάρια, αλλά η επιλογή του αλγορίθμου πρέπει να προσαρμόζεται στο εκάστοτε βιολογικό ερώτημα, είτε αυτό αφορά την εύρεση διακριτών λειτουργικών ομάδων είτε τον εντοπισμό των κεντρικών «λεωφόρων» ροής της βιολογικής πληροφορίας.

Λέξεις-κλειδιά: Εξαγωγή Κορμού Δικτύου (Network Backbone Extraction), Διμερή Δίκτυα (Bipartite Networks / Graphs), Βιολογία Συστημάτων, Πολυπαραγοντικές Ασθένειες, Ανάλυση Δικτύων, High-throughput δεδομένα.

Abstract

The rapid advancement of high-throughput techniques in Systems Biology has led to the accumulation of vast amounts of data, establishing network analysis as a crucial tool for studying multifactorial diseases. However, the resulting gene-disease bipartite networks and their one-mode projections are often characterized by high density and noise, obscuring the identification of biologically meaningful patterns. This thesis focuses on the comparative evaluation and implementation of network backbone extraction algorithms, aiming to identify the most significant associations while filtering out spurious connections.

For this analysis, a bipartite gene-disease association network was constructed using data retrieved from the GWAS Catalog and OMIM databases. Distinct filtering methods were applied and compared, categorized into statistical (e.g., BiCM, SDSM, Polya Filter, MarginalLikelihood), structural (e.g., MST, Edge Betweenness), and hybrid approaches. The evaluation was conducted using topological metrics, including modularity, network diameter, and clustering coefficient.

The results revealed significant divergences in algorithmic performance: methods such as the Modularity Filter preserved dense community structures, ideal for studying comorbidities, whereas the Polya Filter and MST produced highly sparse, tree-like skeletons, highlighting hierarchical relationships. In conclusion, this study demonstrates that there is no "one-size-fits-all" method; rather, the choice of the backbone extraction algorithm must be tailored to the specific biological question, whether it aims to detect distinct functional modules or to identify the core pathways of biological information flow.

Keywords: Network Backbone Extraction, Bipartite Networks / Graphs, Systems Biology, Complex Diseases, Network Analysis, High-throughput Data.

Ευχαριστίες

Ολοκληρώνοντας την παρούσα πτυχιακή εργασία, κλείνει ένα σημαντικό κεφάλαιο της ζωής μου, αυτό της προπτυχιακής φοίτησης. Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη στον επιβλέποντα καθηγητή μου, Παντελεήμωνα Μπάγκο, για την πολύτιμη καθοδήγηση, την εξαιρετική συνεργασία και, κυρίως, για την εμπιστοσύνη που έδειξε στις ικανότητές μου. Χάρη σε εκείνον, είχα την ευκαιρία να ασχοληθώ με ένα αντικείμενο που του ενδιαφέροντος μου και να αποκτήσω γνώσεις που θα με συνοδεύουν στη μελλοντική μου πορεία.

Επιπλέον, θέλω να ευχαριστήσω την καθηγήτριά μου, Παναγιώτα Κοντού, για την υποστήριξη και τις πολύτιμες συμβουλές της κατά τη διάρκεια αυτής της διαδικασίας.

Τέλος, νιώθω ιδιαίτερη ευγνωμοσύνη για την αμέριστη στήριξη και την αδιάκοπη παρουσία της οικογένειάς μου και των φίλων μου σε κάθε βήμα αυτής της διαδρομής.

Πίνακας Περιεχομένων

Κεφάλαιο 1

Εισαγωγή

Το παρόν κεφάλαιο παρουσιάζει τη μεθοδολογία που εφαρμόστηκε για τη συλλογή, τον καθαρισμό και την προετοιμασία των δεδομένων, τα οποία αποτελούν κρίσιμα σημεία για την επιτυχή εκπόνηση της εργασίας. Η αποτελεσματική διαχείριση των δεδομένων είναι θεμελιώδης για την εξαγωγή ακριβών και αξιόπιστων συμπερασμάτων. Η ποιότητα των δεδομένων επηρεάζει άμεσα την αξιοπιστία των αναλύσεων, καθιστώντας την προετοιμασία τους ένα από τα πιο σημαντικά βήματα στη διαδικασία της έρευνας. Στο πλαίσιο της ανάλυσης δεδομένων, η αξιοπιστία των αποτελεσμάτων εξαρτάται άμεσα από τη διαφάνεια των πηγών και την τεκμηρίωση των διαδικασιών επεξεργασίας. Ως εκ τούτου, το κεφάλαιο αυτό στοχεύει στην πλήρη τεκμηρίωση των βημάτων επεξεργασίας των δεδομένων, διασφαλίζοντας την αναπαραγωγιμότητα της μεθόδου που ακολουθήθηκε.

1.1 Πηγές Δεδομένων

Η επιλογή των κατάλληλων πηγών δεδομένων είναι καθοριστικής σημασίας για την εγκυρότητα και την πληρότητα της παρούσας ανάλυσης, δεδομένης της πολυπλοκότητας και του μεγάλου όγκου των γενετικών πληροφοριών. Αξίζει να σημειωθεί πως όλες οι μέθοδοι που κατασκευάστηκαν και χρησιμοποιούνται στην παρούσα εργασία βρίσκουν εφαρμογή σε όλες τις κατηγορίες διμερών δικτύων, ωστόσο επιλέξαμε για την εφαρμογή τους ένα σύνολο δεδομένων που αφορά γονίδια και ασθένειες ή φαινότυποι που έχουν συσχετιστεί με αυτά, προκειμένου να επικυρώσουμε την χρήση των μεθόδων. Για τη δημιουργία του διμερούς δικτύου συσχέτισης γονιδίων-ασθενειών (gene-disease), επιλέξαμε να αντλήσουμε δεδομένα από τις βάσεις GWAS Catalog και OMIM (Online Mendelian Inheritance in Man) [1,2]. Η επιλογή αυτή πραγματοποιήθηκε, διότι οι δύο βάσεις καλύπτουν με συμπληρωματικό τρόπο όλο το φάσμα των γνωστών γενετικών σχέσεων που διέπουν τις ανθρώπινες ασθένειες και έχουν μελετηθεί ανά έτη από αποτελώντας αντικείμενο ενεργού ενδιαφέροντος για τους ερευνητές.

Η συλλογή δεδομένων από τις GWAS Catalog και OMIM επιλέχθηκε ειδικά για να παρέχει μια ολιστική και ισορροπημένη αναπαράσταση της γενετικής αιτιολογίας των ασθενειών. Η GWAS Catalog εστιάζει στις πολυπαραγοντικές ασθένειες (όπως για παράδειγμα ο διαβήτης ή οι καρδιοπάθειες), όπου η συσχέτιση είναι στατιστική και προκύπτει από πολλαπλές κοινές γενετικές παραλλαγές (SNPs). Αντιθέτως, η OMIM καλύπτει τις μονογονιδιακές διαταραχές, όπου η σχέση γονιδίου-ασθένειας είναι αιτιολογική και ισχυρή. Ο συνδυασμός αυτών των δύο πηγών διασφαλίζει ότι το δίκτυο που θα δημιουργηθεί περιλαμβάνει τόσο τις σπάνιες, ισχυρές σχέσεις όσο και τις κοινές, πολύπλοκες σχέσεις που αποτελούν τον στόχο μελέτης της παρούσας εργασίας.

1.1.1 Η βάση δεδομένων NHGRI-EBI’s GWAS Catalog

Η GWAS Catalog (Genome-Wide Association Studies) αποτελεί μια διεθνή βάση δεδομένων που συντηρείται από κοινού από το National Human Genome Research Institute (NHGRI) των Ηνωμένων Πολιτειών και το European Bioinformatics Institute (EBI) του Ευρωπαϊκού Εργαστηρίου Μοριακής

Βιολογίας (EMBL). Η βάση αυτή περιλαμβάνει αποτελέσματα από μελέτες συσχέτισης σε ολόκληρο το γονιδίωμα, οι οποίες εξετάζουν τη σχέση μεταξύ γενετικών παραλλαγών, κυρίως SNPs (Single Nucleotide Polymorphisms), και φαινοτυπικών χαρακτηριστικών ή ασθενειών [1]. Η GWAS Catalog είναι προσβάσιμη μέσω της ιστοσελίδας https://www.ebi.ac.uk/gwas/ και δημιουργήθηκε με στόχο να διευκολύνει την ερμηνεία των αποτελεσμάτων μεγάλων γενετικών μελετών, να προωθήσει τη διαφάνεια των δεδομένων και να παρέχει μια ενιαία πλατφόρμα ενοποίησης και σύγκρισης ερευνητικών ευρημάτων [1]. Η βάση περιλαμβάνει καταχωρήσεις που προέρχονται από δημοσιευμένες, υψηλής ποιότητας επιστημονικές μελέτες, οι οποίες πληρούν αυστηρά στατιστικά και μεθοδολογικά κριτήρια (όπως p-value < 5×10^{-8}) [1]. Χρησιμοποιείται ευρέως για τη χαρτογράφηση συσχετίσεων μεταξύ γενετικών παραλλαγών και βιολογικών ή κλινικών χαρακτηριστικών, παρέχοντας πολύτιμο υπόβαθρο για τη μελέτη της γενετικής προδιάθεσης σε ασθένειες και την κατανόηση των μοριακών μηχανισμών που τις διέπουν [1]. Τα δεδομένα που χρησιμοποιήθηκαν αντλήθηκαν από το αρχείο gwas_catalog.csv (12 Απριλίου 2024), διασφαλίζοντας τη χρήση ενημερωμένων και αξιόπιστων πληροφοριών.

1.1.2 Η βάση δεδομένων NCBI’s OMIM (Online Mendelian Inheritance in Man)

Η Online Mendelian Inheritance in Man (OMIM) αποτελεί μια επίσης διαδικτυακή βάση δεδομένων που παρέχει πληροφορίες για τα γονιδιακά νοσήματα και τις γενετικές διαταραχέςστον άνθρωπο. Η OMIM περιλαμβάνει αναλυτικές καταχωρήσεις που συνδέουν γονίδια, γενετικές παραλλαγές και κληρονομικές ασθένειες, καθώς και τα φαινοτυπικά χαρακτηριστικά που επηρεάζονται από γενετικά αίτια. Η βάση αυτή προσφέρει λεπτομερή δεδομένα σχετικά με τις συσχετίσεις μεταξύ γενετικών μεταλλάξεων και φαινοτύπων, καθιστώντας την ένα πολύτιμο εργαλείο για ερευνητικά δεδομένα [2]. Χρησιμοποιείται ευρέως για την κατανόηση της κληρονομικότητας, των μηχανισμών ανάπτυξης και της παθογένεσης γενετικών διαταραχών, αλλά και για την υποστήριξη της διάγνωσης και της θεραπευτικής προσέγγισης ασθενειών με γενετική βάση [2].

Αυτό που διαφοροποιεί την OMIM και την καθιστά την πιο έγκυρη πηγή για την επιλογή δεδομένων είναι η φιλοσοφία της ως βάση δεδομένων που βασίζεται στην εξειδικευμένη επιμέλεια (expert curation) και στη σύνθεση της γνώσης, δηλαδή δεν φιλοξενεί μόνο πρωτογενή δεδομένα αλληλούχισης ή πειραμάτων, αλλά συνθέτει και συνοψίζει τις πιο σημαντικές και αξιόπιστες πληροφορίες, βασισμένη σε ενδελεχή ανασκόπηση της peer-reviewed βιοϊατρικής βιβλιογραφίας από ειδικούς [2]. Συνεπώς, παρέχει μια έγκυρη και αξιόπιστη περίληψη της τρέχουσας επιστημονικής συναίνεσης για τις γονιδιακές διαταραχές.

Επιπλέον, η OMIM υποστηρίζει τη συνένωση δεδομένων (data integration) από διαφορετικές πηγές, επιτρέποντας τη διασύνδεση πληροφοριών μεταξύ βάσεων δεδομένων όπως το NCBI Gene, το UniProt, το Ensembl και άλλες σημαντικές βιοπληροφορικές πλατφόρμες. Η βάση δεδομένων αναπτύσσεται και συντηρείται από το Johns Hopkins University (Ηνωμένες Πολιτείες Αμερικής) και είναι διαθέσιμη στην ηλεκτρονική διεύθυνση https://omim.org. Ανανεώνεται τακτικά με νέες πληροφορίες που αφορούν γενετικές ανακαλύψεις, νέες ασθένειες και μεταλλάξεις που σχετίζονται με τη γενετική, εξασφαλίζοντας έτσι την επικαιρότητα και την επιστημονική εγκυρότητα των δεδομένων της [2]. Για τις ανάγκες της εργασίας χρησιμοποιήθηκε το αρχείο genemap2.txt το οποίο λήφθηκε από τον ιστότοπο ftp://ftp.ncbi.nih.gov/gene/DATA/mim2gene_medgen (12 Απριλίου 2024) [3].

1.2 Καθαρισμός και Προετοιμασία Δεδομένων

Ο καθαρισμός των δεδομένων πραγματοποιήθηκε με δύο αρχεία σε γλώσσα προγραμματισμού Python, ένα για κάθε πηγή δεδομένων, με σκοπό την τυποποίηση και την εξαγωγή των απαραίτητων πληροφοριών, αξιοποιώντας και τροποποιώντας υπάρχων κώδικα ο οποίος είναι διαθέσιμος στην επίσημη τεκμηρίωση του αρχείου genemap2.txt. Η γλώσσα Python είναι ευρέως γνωστή στον χώρο της ανάλυσης δεδομένων λόγω της μεγάλης ευελιξίας που παρέχει στην σύνταξη κώδικα, στην εύκολη ανάγνωση, την ισχυρή κοινότητα και τις βιβλιοθήκες που παρέχει. Είναι διαθέσιμη προς λήψη στον ιστότοπο https://www.python.org.Το προεπεξεργασμένο αρχείο GWAS catalog αποτελούταν από δεδομένα όπως χρωμόσωμα, θέση έναρξης γονιδιώματος, θέση λήξης γονιδιώματος, κυτταρική τοπολογία, υπολογισμένη κυτταρική τοπολογία, αριθμός MIM, μεταξύ άλλων. Για το αρχείο GWAS αρχικά πραγματοποιήθηκε καθαρισμός των ονομάτων των στηλών του αρχείου και στη συνέχεια δημιουργήθηκε μια ενιαία στήλη για τα γονίδια, επιλέγοντας την πιο έγκυρη πληροφορία από διαφορετικές αρχικές στήλες. Γραμμές που δεν περιείχαν έγκυρο γονίδιο η κενές τιμές (missing values) απορρίφθηκαν. Πολλαπλά γονίδια που ενδέχεται να εμφανίζονταν σε μία καταχώριση, διαχωρίστηκαν σε ξεχωριστές εγγραφές, εξασφαλίζοντας ένα μοναδικό γονίδιο ανά γραμμή, ώστε κάθε γονίδιο να δημιουργεί ζεύγος με την ασθένεια η φαινότυπο που του αντιστοιχεί. Καθαρίστηκαν τυχόν περιττά κενά ή ειδικοί χαρακτήρες που θα μπορούσαν να αλλοιώσουν το όνομα του γονιδίου και απορρίφθηκαν καταχωρίσεις που αποτελούνταν μόνο από αριθμούς. Το τελικό σύνολο δεδομένων περιορίστηκε στις απαραίτητες πληροφορίες (ζεύγη ασθένειας ή φαινοτυπικού χαρακτηριστικού και γονιδίου), αφαιρέθηκαν οι διπλότυπες εγγραφές για να διασφαλιστεί η μοναδικότητα των ζευγών και αποθηκεύτηκε σε μορφή πίνακα. Για τον καθαρισμό των δεδομένων των δύο αρχείων που λήφθηκαν, χρησιμοποιήθηκε ως κώδικας αναφοράς, το αρχείο που βρίσκεται στον σύνδεσμο https://github.com/OMIM-org/genemap2-parser όπως αναγράφεται και στην τεκμηρίωση του αρχείου gen3map2.txt.

Το αρχικό αρχείο από την βάση OMIM περιείχε δεδομένα όπως ασθένεια η χαρακτηριστικό φαινότυπου, περιοχή γονιδίου, αναφερόμενα γονίδια, χαρτογραφημένα γονίδια, και πολυμορφισμούς (SNPs). Για την επεξεργασία των δεδομένων αρχικά αγνοήθηκαν οι γραμμές σχολίων και έγινε έλεγχος για την πληρότητα και εγκυρότητα των βασικών πεδίων που περιέχουν πληροφορίες για γονίδια και φαινοτύπους [3]. Από τις αρχικές καταχωρίσεις, απομονώθηκαν και καθαρίστηκαν τα ονόματα των γονιδίων και των φαινοτύπων. Αυτό περιλάμβανε την αφαίρεση περιττών πληροφοριών όπως αναγνωριστικά κωδικών και ειδικών συμβόλων, ώστε να διατηρηθεί αυτούσιο το όνομα της ασθένειας ή του γονιδίου. Οι στήλες που τα δεδομένα τους δεν χρειάζοταν για την ανάλυση απορρίφθηκαν επίσης. Κάθε γονίδιο συσχετίστηκε με τον αντίστοιχο καθαρισμένο φαινότυπο. Διασφαλίστηκε ότι κάθε ζεύγος φαινοτύπου-γονιδίου είναι μοναδικό, απορρίπτοντας τυχόν επαναλήψεις, και το τελικό αποτέλεσμα εξήχθη σε μορφή πίνακα. Σε εγγραφές όπου δεν υπήρχε εγγραφή για το γονίδιο επιλέχθηκε η περιοχή γονιδίου στο data set.

Οι διαδικασίες καθαρισμού και προετοιμασίας δεδομένων από GWAS Catalog και OMIM οδήγησαν στη δημιουργία δύο καθαρών και δομημένων συνόλων δεδομένων (output_cleaned.csv και output_omim.txt). Αυτά τα αρχεία συγχωνεύτηκαν σε ένα ενιαίο αρχείο δεδομένων, έτοιμο προς ανάλυση, διασφαλίζοντας την ποιότητα και την αξιοπιστία των δεδομένων που θα χρησιμοποιηθούν [3].

Κεφάλαιο 2

2. Εισαγωγή στα διμερή δίκτυα

Η σύγχρονη βιολογία συστημάτων και η βιοπληροφορική έχουν μετατοπίσει την εστίαση από τη μελέτη μεμονωμένων βιολογικών παραγόντων στη διερεύνηση της δυναμικής σύνθετων συστημάτων σε μεγάλη κλίμακα. Οι ραγδαίες εξελίξεις στις τεχνολογίες υψηλής απόδοσης (high-throughput), όπως η αλληλούχιση γονιδιώματος και η ανάλυση γενετικών διαταραχών, οδήγησαν στην παραγωγή τεράστιου όγκου δεδομένων OMICS. Αυτό αποκάλυψε έναν πρωτοφανή αριθμό νέων σχέσεων μεταξύ των βιολογικών οντοτήτων, καθιστώντας αναγκαία την προσέγγιση σε επίπεδο συστήματος. Για την αποτελεσματική μοντελοποίηση και ανάλυση αυτών των περίπλοκων αλληλεπιδράσεων σε δίκτυα γονιδίων ή συσχετίσεις γονιδίων-ασθενειών, η θεωρία γραφημάτων αναδεικνύεται ως ένα κεντρικό εργαλείο, όπου οι βιολογικές οντότητες αναπαρίστανται ως κόμβοι και οι σχέσεις τους ως ακμές [4].

Μια ιδιαίτερα χρήσιμη κατηγορία για την αναπαράσταση αυτών των δεδομένων είναι τα διμερή γραφήματα (bipartite graphs), γνωστά και ως δίκτυα δύο καταστάσεων (2-mode networks). Σε αυτά τα γραφήματα, οι κόμβοι χωρίζονται σε δύο διακριτά, μη επικαλυπτόμενα σύνολα (π.χ., γονίδια U και ασθένειες V), με τις ακμές να συνδέουν κόμβους μόνο μεταξύ των δύο συνόλων. Αυτή η δομή είναι ιδανική για τη μοντελοποίηση σχέσεων όπου εμπλέκονται δύο διαφορετικοί τύποι οντοτήτων, καθώς αντανακλά με ακρίβεια την εγγενή δομή των συσχετίσεων γονιδίων-ασθενειών [4]. Η κατανόηση των τοπολογικών ιδιοτήτων αυτών των διμερών δικτύων, σε συνδυασμό με τη χρήση μεθόδων όπως η προβολή (projection) και μέθοδοι εξαγωγής σκελετού διμερών δικτύων, είναι κρίσιμη για την αποκάλυψη νέων πληροφοριών για τις λειτουργίες του δικτύου. Στο πλαίσιο αυτό, η παρούσα εργασία διερευνά τον ρόλο και τη μεθοδολογία ανάλυσης των διμερών δικτύων στη βιολογία συστημάτων. Αρχικά, αναλύονται οι βασικές μαθηματικές έννοιες και οι τοπολογικές ιδιότητες των διμερών γραφημάτων. Στη συνέχεια, παρουσιάζονται και συγκρίνονται οι διάφορες στρατηγικές προβολής που επιτρέπουν τη μετατροπή του διμερούς δικτύου σε μονομερές, ώστε να εφαρμοστούν τυποποιημένα μέτρα κεντρικότητας.

2.1 Ορισμός και Αναπαράσταση Διμερών Δικτύων

Ορισμός 2.1

Ένα δίκτυο που αποτελείται από δύο σύνολα κόμβων, U και V, για να χαρακτηρίζεται **διμερές**, πρέπει κάθε ακμή συνδέει έναν κόμβο του U με έναν κόμβο του V, αλλά χωρίς να υπάρχουν συνδέσεις μεταξύ κόμβων του ίδιου συνόλου και ορίζεται ως ένα γράφημα $B=(U,V,E)$, όπου:

- U και V είναι δύο μη κενά, διακριτά σύνολα κορυφών (κόμβων), γνωστά ως μέρη του γραφήματος.
- E είναι το σύνολο των ακμών, όπου κάθε ακμή $e \in E$ συνδέει αποκλειστικά έναν κόμβο από το U με έναν κόμβο από το V, δηλαδή $E \subseteq U \times V$. Δεν επιτρέπονται ακμές μεταξύ κόμβων του ίδιου συνόλου [4].

Ορισμός 2.2

Με τον όρο διμερές ορίζουμε ένα δίκτυο η γράφημα αν και μόνο αν ισχύει η ιδιότητα Διμερότητας (Bipartiteness), δηλαδή δεδομένου 2 συνόλων κόμβων, όπου οι συνδέσεις εμφανίζονται μόνο μεταξύ κόμβων που ανήκουν σε διαφορετικά σύνολα και όχι εντός του ίδιου συνόλου [4].

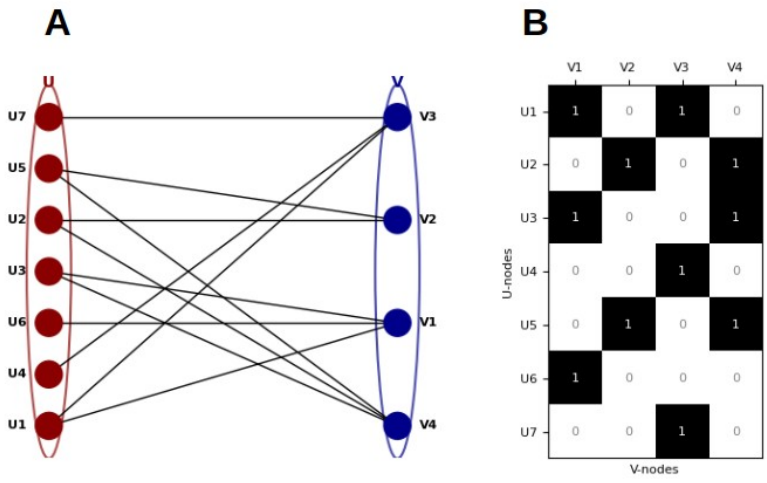
Ορισμός 2.3

Ένα ισορροπημένο διμερές γράφημα είναι ένα διμερές γράφημα στο οποίο τα δύο σύνολα κορυφών έχουν ίσο πλήθος στοιχείων, δηλαδή $|U| = |V|$. Αν όλοι οι κόμβοι του ίδιου μέρους έχουν τον ίδιο βαθμό (degree), τότε το γράφημα ονομάζεται δικανονικό (biregular) [4].

Ένα διμερές γράφημα μπορεί να αναπαρασταθεί με μία ορθογώνια μήτρα γειτνίασης B, διαστάσεων $|U| \times |V|$ [4]. Μαθηματικά εκφράζεται:

$$B_{ij} = \begin{cases} 1, & \alpha\nu \ \ (u_i, s_j) \in E(G) \\ 0, & \delta\iota\alpha\phi\rho\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$$

Τα διμερή γραφήματα μπορούν να είναι σταθμισμένα, όταν οι ακμές αντιπροσωπεύουν διαφορετικά επίπεδα συχνότητας αλληλεπίδρασης, όπως συμβαίνει στα δίκτυα αλληλεπιδράσεων γονιδίων-ασθενειών. Σε αυτά τα δίκτυα, οι ακμές μπορεί πολλές φορές να μην είναι δυαδικές (0 ή 1), αλλά να φέρουν μεγαλύτερες τιμές που υποδεικνύουν τη συχνότητα ή την ένταση των αλληλεπιδράσεων μεταξύ των κόμβων.



Εικόνα 2.1 Απεικόνιση διμερούς γραφήματος και του αντίστοιχου πίνακα γειτνίασης. (Α) Απεικόνιση ενός διμερούς γραφήματος. (Β) Ο πίνακας γειτνίασης του διμερούς γραφήματος.

2.2 Βασικές Ιδιότητες Διμερών Δικτύων

Στην παρούσα ενότητα, παρουσιάζονται ορισμένες θεμελιώδεις ιδιότητες των διμερών γραφημάτων, οι οποίες αποδεικνύουν την δομή τους. Αυτές οι ιδιότητες έχουν σημαντικές εφαρμογές στην κατανόηση της συμπεριφοράς των διμερών γραφημάτων και στην αποτελεσματική εφαρμογή τους σε διάφορες περιοχές, προκειμένου να γίνουν περισσότερο κατανοητές οι μεθοδολογίες των επόμενων κεφαλαίων. Ανάλογα με τη δομική πτυχή που επιδιώκουμε να αναλύσουμε όπως η συνέπεια των συνδέσεων, τα φυσικά όρια των δυνατών συνδέσεων κάθε κόμβου, την πυκνότητα, η το μέγιστο μέγεθος του δικτύου, έχουν οριστεί οι αντίστοιχες μαθηματικές ιδιότητες.

Σε ένα διμερές γράφημα, το συνολικό πλήθος των ακμών που συνδέονται με τους κόμβους της μιας πλευράς είναι ίσο με το πλήθος των ακμών που συνδέονται με τους κόμβους της άλλης. Πιο αναλυτικά, $B(U, V, E)$ όπου $|U| = n_1$, $|V| = n_2$ και $|E| = m|E|$, ισχύουν οι ακόλουθες ιδιότητες [5].

Ιδιότητα 1

Για κάθε διμερές μη κατευθυνόμενο γράφημα το άθροισμα των βαθμών του ενός συνόλου U είναι ίσο με το αντίστοιχο άθροισμα του V και ίσο με το συνολικό πλήθος ακμών [5].

$$\sum_{i=1}^{|n_1|} deg(u_i) = \sum_{j=1}^{|n_2|} deg(v_j) = |\mathrm{E}|$$

Ιδιότητα 2

Ο βαθμός του κάθε κόμβου u_i, v_i του κάθε συνόλου U, V είναι τουλάχιστον 1 και το πολύ ίσο με το μέγεθος n του αντίστοιχου άλλου συνόλου [5].

$$1 \le deg(u_i) \le n_2 \text{ , } \forall u_i \in U \text{, όπου } n_2 \text{το μέγεθος του } V$$

$$1 \le deg(v_j) \le n_1 \text{ , } \forall v_j \in V \text{, όπου } n_1 \text{ το μέγεθος του } U$$

Ιδιότητα 3

Ο μέγιστος αριθμός ακμών σε έναν πλήρη διμερές γράφο δίνεται από τον τύπο $|E|max = n_1 \cdot n_2$ [5].

Ιδιότητα 4

Ένας γράφος θεωρείται αραιός (sparse) αν ο αριθμός των ακμών είναι της τάξης $O(|V|)$και πυκνός (dense) αν ο αριθμός των ακμών είναι της τάξης $O(|V|^2)$ [5].

Ιδιότητα 5

Αν $|U| = m$ και $|V| = n$, τότε το μέγιστο δυνατό πλήθος ακμών είναι $m \times n$ [5].

2.3 Βασικά Μέτρα Διμερών Δικτύων

Ανάλογα με το αν θέλουμε να μετρήσουμε την προσβασιμότητα ενός κόμβου, τον ρόλο του ως μεσάζοντα ή την επιρροή του βάσει των συνδέσεών του, έχουν οριστεί τα αντίστοιχα μέτρα κεντρικότητας, το καθένα από τα οποία αναδεικνύει διαφορετικές πτυχές της δομής του δικτύου [6].Η ανάλυση κεντρικότητας βοηθά στην αναγνώριση των πιο σημαντικών κόμβων στο δίκτυο, χρησιμοποιώντας διαφορετικές μεθόδους, για να προσδιοριστούν τα κόμβοι που έχουν μεγαλύτερη επίδραση στο σύστημα. Πιο συγκεκριμένα ορίζονται οι ακόλουθες μετρικές:

Degree Centrality: Μετράται από το συνολικό πλήθος των άμεσων συνδέσεων (δεσμών/ακμών) που έχει ο κόμβος με τους άλλους κόμβους στο δίκτυο [6].

$$C_d(N_i) = \sum_{j=1}^{n} X_{ij}$$

όπου X_{ij} το στοιχείο του πίνακα γειτνίασης (Adjacency Matrix). Είναι 1 αν υπάρχει άμεσος δεσμός μεταξύ του $N_{¿}$i και του N_j, και 0 αν δεν υπάρχει.

Closeness Centrality: Χρησιμοποιείται όταν το ενδιαφέρον μας εστιάζει στον εντοπισμό των κόμβων που μπορούν να προσεγγίσουν γρήγορα όλους τους υπόλοιπους στο δίκτυο. Το μέτρο αυτό είναι χρήσιμο σε περιπτώσεις όπου η ταχύτητα διάδοσης πληροφορίας είναι κρίσιμη, καθώς κόμβοι με υψηλή τιμή closeness θεωρούνται "κοντά" σε όλους τους άλλους μέσω σύντομων διαδρομών [6].

$$C_c(N_i) = \frac{1}{\sum_{j=a}^{n} d(N_i, N_j)}$$

με $\sum_{j=a}^{n} d(N_i, N_j)$ να είναι το συνολικό άθροισμα των αποστάσεων (μήκος συντομότερης διαδρομής) από τον κόμβο N_i προς όλους τους άλλους κόμβους N_j του δικτύου [6].

Betweenness Centrality: Εφαρμόζεται όταν είναι χρήσιμη η εύρεση κόμβων που λειτουργούν ως "γέφυρες" μεταξύ διαφορετικών περιοχών του δικτύου. Οι κόμβοι αυτοί είναι σημαντικοί για την κατανόηση της ροής πληροφορίας ή πόρων, καθώς συμμετέχουν συχνά στις συντομότερες διαδρομές μεταξύ άλλων κόμβων.

$$C_b(N_i) = \sum_{k<j} \frac{G_{jk}(N_i)}{G_{jk}}$$

όπου $G_{jk}(N_i)$ ο αριθμός των συντομότερων διαδρομών μεταξύ N_j και N_k που διέρχονται από τον N_i και G_{jk} ο συνολικός αριθμός των συντομότερων διαδρομών μεταξύ του N_j και του N_k [6].

Eigenvector Centrality: Χρησιμοποιείται όταν δεν είναι αρκετή η γνώση του αριθμού των συνδέσεων ενός κόμβου, αλλά και στην ταυτοποίησή τους, ποιοι είναι οι γείτονές του. Το μέτρο αυτό

εκτιμά την επιρροή ενός κόμβου, με βάση την κεντρικότητα των συνδεδεμένων κόμβων [6]. Έτσι, ένας κόμβος που συνδέεται με άλλους σημαντικούς κόμβους αποκτά και ο ίδιος υψηλή βαθμολογία.

Μαθηματικά εκφράζεται:

$$\lambda x = Ax$$

Όπου:

- λ είναι η ιδιοτιμή (Eigenvalue).
- A είναι ο πίνακας γειτνίασης (Adjacency Matrix) του γραφήματος.
- x είναι το ιδιοδιάνυσμα (Eigenvector) που περιέχει τις βαθμολογίες (score) κεντρικότητας κάθε κόμβου [7].

Καθένα από τα παραπάνω μέτρα προσφέρει διαφορετική πληροφόρηση για τη σημασία των κόμβων μέσα στο δίκτυο και επιλέγεται ανάλογα με τη φύση του προβλήματος και το είδος της ανάλυσης που επιθυμούμε να πραγματοποιήσουμε.

Κεφάλαιο 3

3.1 Εισαγωγή στις Μεθόδους Προβολής Διμερών Δικτύων

Σε πολλές περιπτώσεις, για την καλύτερη ανάλυση και κατανόηση της δομής του δικτύου, απαιτείται η μετατροπή του διμερούς δικτύου σε μονομερές (one-mode) δίκτυο, όπου οι κόμβοι ανήκουν σε ένα μόνο σύνολο και οι συνδέσεις δημιουργούνται μεταξύ των κόμβων του ίδιου συνόλου βάσει των σχέσεων που προκύπτουν από το διμερές δίκτυο. Η διαδικασία αυτή ονομάζεται προβολή (projection) [8].

Ειδικότερα επιτρέπουν την ανακάλυψη σχέσεων και μοτίβων μεταξύ διαφορετικών τύπων κόμβων, όπως γονίδια και ασθένειες, μετατρέποντας τις σχέσεις τους σε μονομερές δίκτυο (π.χ. γονίδια-γονίδια ή ασθένειες-ασθένειες). Αυτή η διαδικασία διευκολύνει την ανάλυση κοινών χαρακτηριστικών ή συμπεριφορών, όπως η αναγνώριση γονιδίων που σχετίζονται με κοινές ασθένειες. Αυτή η μετατροπή επιτρέπει την αποτελεσματικότερη χρήση εργαλείων που έχουν σχεδιαστεί για ομογενή δίκτυα, ενισχύοντας την κατανόηση των βιολογικών μηχανισμών. Επιπλέον, η μείωση της πολυπλοκότητας διευκολύνει την οπτικοποίηση και την ερμηνεία των δεδομένων, επιτρέποντας στους ερευνητές να επικεντρωθούν στα πιο σημαντικά ζεύγη (π.χ. τα πιο συσχετισμένα γονίδια) για περαιτέρω πειραματική επικύρωση [4,8].

Οι μέθοδοι προβολής διακρίνονται κυρίως σε δύο κατηγορίες, την απλή (unweighted) προβολή, όπου οι νέες συνδέσεις στο μονομερές δίκτυο είτε υπάρχουν είτε όχι, και τη σταθμισμένη (weighted) προβολή, όπου το βάρος της νέας σύνδεσης υπολογίζεται με βάση τον αριθμό των κοινών γειτόνων στο αρχικό διμερές δίκτυο. Ωστόσο, είναι σημαντικό να σημειωθεί ότι, ενώ η προβολή διευκολύνει την ανάλυση, συχνά οδηγεί σε απώλεια πληροφορίας σχετικά με τη λεπτομερή δομή του αρχικού διμερούς δικτύου, καθώς η πληροφορία των ενδιάμεσων κόμβων χάνεται [8,9].

3.2 Απλή Μονομερής Προβολή (Simple One-Mode Projection)

Τα Διμερή Δίκτυα (Bipartite Networks) είναι ιδιαίτερα συνηθισμένα, καθώς μοντελοποιούν τη συνάφεια μεταξύ δύο διακριτών συνόλων κόμβων U και V. Ωστόσο, για να εφαρμοστούν τεχνικές που αναπτύχθηκαν για μονομερή δίκτυα (όπως η εύρεση κοινοτήτων, ο υπολογισμός κεντρικότητας ή η εξαγωγή του κορμού ενός δικτύου), είναι συχνά απαραίτητο να μετασχηματιστούν τα διμερή δίκτυα σε μονομερή. Αυτός ο μετασχηματισμός επιτυγχάνεται μέσω της διαδικασίας της μονομερούς προβολής (one-mode projection). Η απλούστερη και πιο θεμελιώδης μορφή αυτής της διαδικασίας είναι η απλή μονομερής προβολή, η οποία επιτρέπει τη μελέτη των σχέσεων εντός ενός συνόλου (π.χ., μόνο μεταξύ των κόμβων U), με βάση τις κοινές συνδέσεις που μοιράζονται στο αρχικό διμερές δίκτυο. Η σωστή κατανόηση και εφαρμογή της προβολής είναι κρίσιμη, καθώς επηρεάζει άμεσα την εσωτερική δομή και τις ιδιότητες του δικτύου που προκύπτει [8].

Αυτή η μέθοδος κατασκευάζει ένα μονομερές δίκτυο όπου δύο κόμβοι από το ίδιο σύνολο συνδέονται αν έχουν κοινές συνδέσεις στο αρχικό διμερές δίκτυο. Δεδομένου ενός διμερούς δικτύου όπως ορίστηκε προηγουμένως η μονομερής προβολή μπορεί να οριστεί ως ένας γράφος $G=(U,U,E')$, όπου $X \subseteq U$ ή $X \subseteq V$, και η ύπαρξη ακμής μεταξύ δύο κόμβων του X καθορίζεται από τη συνεκφορά τους με κόμβους του αντίθετου συνόλου [8]. Υπάρχουν δύο εκδοχές:

i) Προβολή επί των κόμβων U: Δύο κόμβοι στο U συνδέονται αν έχουν κοινό γείτονα στο V.

ii) Προβολή επί των κόμβων V: Δύο κόμβοι στο V συνδέονται αν έχουν κοινό γείτονα στο U.

3.2.1 Ορισμός της Απλής Μονομερούς Προβολής

Ορισμός 3.1

Έστω ένα διμερές γράφημα $G=(U,V,E)$, όπου U και V είναι δύο διακριτά σύνολα κόμβων, στη περίπτωσή μας το ένα σύνολο αποτελούν τα γονίδια και το άλλο οι ασθένειες. Το σύνολο των ακμών είναι E, όπου κάθε ακμή συνδέει έναν κόμβο από το U με έναν κόμβο από το V δηλαδή $E \subseteq U \times V$.

Η προβολή επί των κόμβων U παράγει ένα μονομερές γράφημα $G=(U,U,E)$, όπου δύο κόμβοι $u_i, u_j \in U$ συνδέονται αν υπάρχει τουλάχιστον ένας κοινός γείτονας $v \in V$ τέτοιος ώστε: $(u_i, v) \in E$ και $(u_j, v) \in E$ [8]. Αντίστοιχα, κατασκευάζεται η προβολή επί των κόμβων V.

Στο μονομερές δίκτυο που προκύπτει, η ύπαρξη μιας ακμής μεταξύ δύο κόμβων U (π.χ., δύο γονίδια) ερμηνεύεται ως η κοινή συμμετοχή τους σε κάποιο γεγονός, χαρακτηριστικό ή ιδιότητα (π.χ., συνδέονται και τα δύο με την ίδια ασθένεια του συνόλου V). Η απλή προβολή παράγει ένα μη σταθμισμένο (unweighted) δίκτυο. Ο αριθμός των κοινών γειτόνων αγνοείται στην απλή προβολή, κάτι που αποτελεί τον κύριο περιορισμό της.

Ορισμός 3.2

Έστω B ο πίνακας γειτνίασης ενός διμερούς γραφήματος. Η μονομερής προβολή ενός διμερούς δικτύου εκφράζεται ως το γινόμενο του πίνακα γειτνίασης του αρχικού διμερούς δικτύου με τον ανάστροφό πίνακα αυτού [8].

Μαθηματικά δίνεται από τον ακόλουθο τύπο: $P_U = BB^T$, για την προβολή ως προς το σύνολο κόμβων U, ενώ $P_V = B^T B$ για την προβολή ως προς το σύνολο κόμβων V, δεδομένου ότι ο πίνακας γειτνίασης B έχει ως γραμμές τους κόμβους του συνόλου U και στήλες κόμβους του συνόλου V [8]. Για να προκύψει η τελική απλή (μη σταθμισμένη) προβολή, ο πίνακας P πρέπει να δυαδικοποιηθεί (δηλαδή, όλα τα μη μηδενικά στοιχεία του, εκτός της διαγωνίου, να αντικατασταθούν με 1).

Παρατήρηση 3.1

Έστω ένας συμμετρικός πίνακας μονομερούς προβολής P, ισχύει ότι κάθε στοιχείο της κύριας διαγωνίου P_{ii}, ισούται με τον βαθμού του κόμβου u_i στο αρχικό διμερές δίκτυο. Επομένως ισχύει ότι:

$$P_{ii} = degree(node_{ij})$$ [8].

3.2.2 Υπολογιστική Υλοποίηση και Δομική Ανάλυση των Δικτύων Προβολής

Η μονομερής προβολή του διμερούς δικτύου ασθενειών υλοποιήθηκε προγραμματιστικά σε γλώσσα Python και με τις δύο μεθόδους που περιγράφηκαν παραπάνω (Ορισμοί 3.1 και 3.2). Η πρώτη μέθοδος βασίστηκε στη λογική του κοινού γείτονα, με χρήση άπληστης αναζήτησης (Brute Force Search) για την εύρεση κοινών γειτόνων χρησιμοποιώντας δομές δεδομένων λεξικών, ενώ η δεύτερη μέθοδος υλοποιήθηκε μέσω γραμμικού πολλαπλασιασμού των πινάκων γειτνίασης του δικτύου, αξιοποιώντας αραιούς πίνακες (sparse matrices) [8]. Παρά τη διαφορετική προσέγγιση των δύο μεθόδων, η έξοδος τους ήταν ταυτόσημη όσον αφορά τις συνδέσεις που προέκυψαν στο μονομερές δίκτυο, επιβεβαιώνοντας τη συνέπεια των αποτελεσμάτων. Ωστόσο, η μέθοδος του πολλαπλασιασμού πινάκων αποδείχθηκε σημαντικά πιο γρήγορη σε σχέση με τη μέθοδο των κοινών γειτόνων με χρήση λεξικών, με συνολικό χρόνο εκτέλεσης 0,05 και 0,21 δευτερόλεπτα αντίστοιχα. Αυτό οφείλεται στην αποτελεσματική διαχείριση των πράξεων στους αραιούς πίνακες, οι οποίοι αποθηκεύουν μόνο τα μη μηδενικά στοιχεία, εξοικονομώντας υπολογιστικό κόστος και χρόνο, όπως επιβεβαιώνεται από τα αποτελέσματα. Το αρχικό διμερές δίκτυο περιελάμβανε 3854 κόμβους γονιδίων και 1268 κόμβους ασθενειών (συνολικά 5124) ενώ οι ακμές που ενώνουν τα δύο σύνολα είναι 7158.

i) Απλή μονομερής προβολή ως προς τις ασθένειες

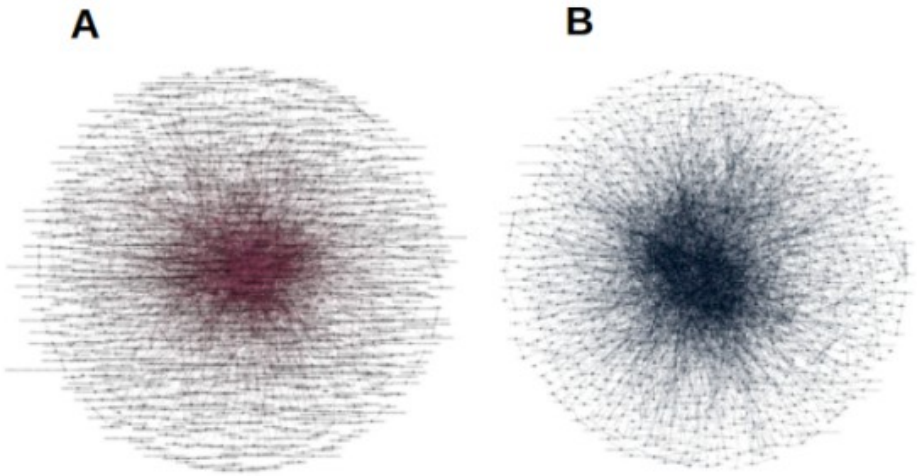
Το μονομερές δίκτυο που προέκυψε από την προβολή επί των ασθενειών με 743 κόμβοι στο σύνολο δεδομένων μας και 5.694 ακμές, δηλαδή σχέσεις μεταξύ του ίδιου συνόλου, χαρακτηρίζεται από δομή με σαφή ομαδοποίηση (βλ. Εικόνα 3.1 Α, Πίνακες 3.1, 3.2). Η χαμηλή πυκνότητα (0.01) δείχνει ότι, οι περισσότερες ασθένειες του συνόλου δεδομένων δεν συνδέονται μεταξύ τους μέσω κοινού γονιδιακού υπόβαθρου, γεγονός που υποδηλώνει υψηλό βαθμό εξειδίκευσης. Ωστόσο, η παρουσία αρκετών ακμών υποδηλώνει ότι υπάρχει ένας αριθμός ζευγών ασθενειών που μοιράζονται σημαντικές γενετικές συσχετίσεις. Ο μέσος βαθμός του δικτύου ανέρχεται σε 11 (βλ. Πίνακας 3.2), που σημαίνει ότι, κατά μέσο όρο, κάθε ασθένεια συνδέεται με περίπου έντεκα άλλες ασθένειες μέσω κοινών γονιδίων. Η ελάχιστη τιμή βαθμού είναι 1, κάτι που δείχνει την ύπαρξη ασθενειών με σχεδόν μοναδικό γενετικό συσχετισμό (μονογονιδιακές), ενώ η μέγιστη τιμή 162 φανερώνει ότι ορισμένες ασθένειες αποτελούν κεντρικούς κόμβους (hubs), συνδεόμενες με μεγάλο αριθμό άλλων ασθενειών. Αυτές οι υψηλού βαθμού ασθένειες χαρακτηρίζονται από πολυγονιδιακές επιδράσεις.

Παρά την αραιότητα του δικτύου, ο μέσος όρος clustering φτάνει το 0.72, τιμή αρκετά υψηλή. Αυτό σημαίνει ότι οι ασθένειες τείνουν να σχηματίζουν τοπικές, τριγωνικές δομές, όπου ομάδες ασθενειών εμφανίζουν αλληλεπικαλυπτόμενες γενετικές συσχετίσεις. Η μορφολογία αυτή υποδηλώνει την ύπαρξη ασθενειών ομαδοποιημένων γύρω από κοινές λειτουργικές διεργασίες ή βιολογικές οδούς (pathways). Το μέσο συντομότερο μονοπάτι, το οποίο υπολογίστηκε στο μεγαλύτερο συνεκτικό συστατικό, έχει τιμή 3.15, δείχνοντας ότι οποιεσδήποτε δύο ασθένειες συνδέονται κατά μέσο όρο με τρία περίπου βήματα. Η διάμετρος του δικτύου είναι 9, γεγονός που υποδηλώνει ότι η μέγιστη απόσταση μεταξύ δύο ασθενειών στο μεγαλύτερο συνεκτικό υποδίκτυο είναι σχετικά μικρή. Αυτό σημαίνει ότι οι γενετικές συσχετίσεις δεν είναι τυχαία κατανεμημένες, αλλά διαμορφώνουν ένα συνεκτικό πυρήνα στον οποίο οι ασθένειες συγκεντρώνονται γύρω από κοινά βιολογικά χαρακτηριστικά.

Συνολικά, το δίκτυο ασθενειών που προκύπτει από τη μονομερή προβολή του διμερούς γράφου παρουσιάζει ισχυρή τοπική οργάνωση, έντονη παρουσία κόμβων υψηλού βαθμού και σχετικά μικρές αποστάσεις μεταξύ των ασθενειών. Τα χαρακτηριστικά αυτά υποδηλώνουν ότι το γενετικό υπόβαθρο πολλών παθολογιών συμπλέκεται σε δομημένες υποκοινότητες, οι οποίες μπορεί να αντανακλούν κοινές λειτουργικές κατηγορίες, παθοβιολογικούς μηχανισμούς ή μοριακές αλληλεπιδράσεις.(βλ. Πίνακα 3.2).

ii) Απλή μονομερής προβολή ως προς τα γονίδια

Αντιθέτως, το δίκτυο που προέκυψε από την προβολή επί των γονιδίων (3.855 κόμβοι, 140.826 ακμές, Εικόνα 3.1 Β) παρουσιάζει μια εντελώς διαφορετική, ομοιογενή και συνεκτική δομή. Ο μέσος βαθμός (79 κόμβοι) και ο μεγάλος αριθμός ακμών υποδηλώνουν εξαιρετικά υψηλή συνδεσιμότητα, ενώ ο εξαιρετικά υψηλός συντελεστής συσσωμάτωσης (0.852) δείχνει έντονο σχηματισμό τριγώνων και ισχυρή τοπική διασύνδεση. Το πιο χαρακτηριστικό εύρημα είναι η πολύ χαμηλή αρθρωτότητα (0.037), η οποία υποδηλώνει την απουσία σαφώς διακριτών κοινοτήτων και ομοιογενή κατανομή των σχέσεων, σε αντίθεση με το δίκτυο των ασθενειών. Τέλος, η ύπαρξη μίας ενιαίας συνδεδεμένης συνιστώσας και το πολύ χαμηλό μέσο μήκος διαδρομής (2.883) επιβεβαιώνουν την εξαιρετική συνοχή του δικτύου, εξασφαλίζοντας ταχύτατη διάδοση πληροφορίας. Συγκριτικά, η προβολή ασθενειών σχηματίζει ένα δίκτυο με σαφή ομαδοποίηση (υψηλή Modularity), ενώ η προβολή γονιδίων σχηματίζει ένα δίκτυο που είναι εξαιρετικά πυκνό και ομοιογενές (χαμηλή Modularity), υποδηλώνοντας ότι τα εμπλεκόμενα γονίδια αποτελούν ένα ενιαίο, αλληλεπιδραστικό σύστημα.



Εικόνα 3.1 Οπτική αναπαράσταση των γραφημάτων προβολής με την χρήση του λογισμικού Gephi. (Α) Γραφική Αναπαράσταση της One Mode Projection ως προς τις ασθένειες του πραγματικού συνόλου δεδομένων.(Β) Γραφική Αναπαράσταση της One Mode Projection ως προς τα γονίδια πραγματικού συνόλου δεδομένων και στατιστικά του δικτύου.

Πίνακας 3.1 Συγκριτική παρουσίαση των βασικών στατιστικών μεγεθών (αριθμός κόμβων, αριθμός ακμών και πυκνότητα) των δύο μονομερών δικτύων που προέκυψαν από την απλή προβολή του διμερούς δικτύου γονιδίων-ασθενειών.

Δίκτυο	Αριθμός Κόμβων	Αριθμός Ακμών	Πυκνότητα
Disease Projected Network	743	5694	0.01
Gene projected Network	3855	140827	0.02

Πίνακας 3.2 Συγκριτική παρουσίαση των μετρικών βαθμού και τοπολογίας (clustering, μέσο συντομότερο μονοπάτι και διάμετρος) των δύο δικτύων προβολής (γονίδια και ασθένειες).

Δίκτυο	Μέσος Βαθμός	Ελάχ. Βαθμός	Μέγ. Βαθμός	Μέσος όρος Clustering	Μέσο Συντομότερο Μονοπάτι	Διάμετρος
Disease Projected Network	11	1	162	0.72	3.15	9
Gene projected Network	78	1	652	0.85	3.15	9

3.3 Σταθμισμένη Προβολή (Weighted Projection)

Στο πλαίσιο των δμερών δικτύων γονιδίων ασθενειών, η απλή σταθμισμένη προβολή (που βασίζεται στον αριθμό των κοινών ασθενειών που επηρεάζουν δύο γονίδια) θεωρείται ποσοτικά μεροληπτική και λιγότερο πληροφοριακή επειδή αδυνατεί να αποτυπώσει την πραγματική λειτουργική σημασία της συνέκφρασης [9]. Συγκεκριμένα, η μέθοδος αυτή δεν ενσωματώνει την εξατομίκευση, αφού αντιμετωπίζει κάθε κοινή ασθένεια ισότιμα, αγνοώντας ότι η πρώτη κοινή ασθένεια σε μια νέα συσχέτιση μπορεί να υποδεικνύει μια ισχυρότερη νέα σύνδεση από μία εκατοστή συν-έκφραση σε συνθήκες που είναι ήδη γνωστές [9]. Στην απλή προβολή, δύο κόμβοι συνδέονται αν έχουν τουλάχιστον έναν κοινό γείτονα στο διμερές δίκτυο. Όμως, αυτό δεν διακρίνει ισχυρές από αδύναμες συνδέσεις [9]. Επιπλέον, πάσχει από έλλειψη διόρθωσης βαθμού, καθώς δεν αντισταθμίζει το γεγονός ότι μία ασθένεια που ενεργοποιείται από πολλά γονίδια (υψηλός βαθμός κόμβου Y) δίνει μικρότερο πληροφοριακό βάρος στην αλληλεπίδραση μεταξύ δύο συγκεκριμένων γονιδίων X σε σχέση με ένα εξειδικευμένο πείραμα που ενεργοποιεί λίγα γονίδια.

Η βαρυμένη προβολή των μονομερών δικτύων αναπτύχθηκε ως βελτίωση της απλής μονομερούς προβολής, προκειμένου να ενσωματώνει πληροφορία για την ισχύ των συνδέσεων μεταξύ των κόμβων. Για να μειωθεί η απώλεια πληροφορίας, οι ακμές στη μονομερή προβολή μπορούν να αποκτήσουν βάρη που υποδεικνύουν τη δύναμη της σχέσης μεταξύ δύο κόμβων. Στη σταθμισμένη προβολή, η ύπαρξη ακμής καθορίζεται όπως στην απλή προβολή, αλλά επιπλέον ανατίθεται βάρος $w(u_i, u_j)$ στην ακμή που ενώνει τους κόμβους [9].

3.3.1 Μέτρα Ομοιότητας και Ανάθεσης Βάρους

Δεδομένου ενός διμερούς γραφήματος $G(U, V, E)$ που περιέχει πλειάδες (u_i, v_j, w_{ij}) όπου w_{ij} είναι το βάρος μεταξύ των κόμβων του U και V, το βάρος μπορεί να ανατεθεί αξιοποιώντας ανάλογα την μελέτη που πραγματοποιείται στο γράφημα απο τις ακόλουθες μεθόδους :

Number of Common Neighbors: Ο πιο διαδεδομένος και απλος συντελεστής. Το βάρος της ακμής είναι ίσο με τον αριθμό των κοινών κόμβων στο διμερές δίκτυο με βάρος : $w_{ij} = |U(i) \cap V(j)|$

με $U(i)$ο αριθμός των στοιχείων του κόμβου i και $V(j)$ ο αριθμός των στοιχείων του κόμβου j [9].

Resource Allocation Index: Μετρά την ομοιότητα μεταξύ κόμβων βασισμένη στην κατανομή πόρων και την πιθανότητα να μπορεί ένας κόμβος να συνδεθεί με έναν άλλο μέσω κοινών γειτόνων [9].

$$Score(u, v) = \sum \frac{1}{|\Gamma(z)|}$$

Jaccard similarity: Μετρά την ομοιότητα μεταξύ δύο πεπερασμένων συνόλων (A και B). Ορίζεται ως το πηλίκο του αριθμού των κοινών γειτόνων μεταξύ των κόμβων i και j, προς τον ο αριθμό των μοναδικών γειτόνων μεταξύ των κόμβων i και j, δηλαδή το μέγεθος της ένωσης των δύο συνόλων [10]. Μαθηματικά εκφράζεται:

$$w_{[ij]}=\frac{\{|U(i)\cap V(j)|\}}{\{|U(i)\cup V(j)|\}}$$

Salton's Cosine similarity (Συνάφεια Συνημιτόνου): Αποτελεί μια μέτρηση ομοιότητας μεταξύ δύο μη-μηδενικών διανυσμάτων σε έναν πολυδιάστατο χώρο.

$$Cosine\,Similarity(A,B)=\frac{A\cdot B}{\|A\|\cdot\|B\|}$$

Έστω ότι έχουμε δύο διανύσματα A και B στον n-διάστατο χώρο (δηλαδή κάθε διανύσμα έχει n στοιχεία) όπως στην περιπτωσή μας, οι διαστάσεις είναι $1\,x$ *Αριθμός Μοναδικών Ασθενειών* για τα γονίδια [11].Τα διανύσματα A και B έχουν την εξής μορφή:

$$A=[A_1,A_2,\ldots,A_n],\;\; B=[B_1,B_2,\ldots,B_n]$$

Το εσωτερικό γινόμενο (dot product) μεταξύ των διανυσμάτων A και B υπολογίζεται ως το άθροισμα των γινομένων των συντεταγμένων τους:

$$A\cdot B=\sum_{i=1}^{n} A_i\cdot B_i=A_1\cdot B_1+A_2\cdot B_2+\cdots+A_n\cdot B_n$$

Adamic-Adar Index: Ο δείκτης Adamic-Adar μετρά την ομοιότητα μεταξύ δύο κόμβων λαμβάνοντας υπόψη τον αριθμό των κοινούς γειτόνων τους, δίνοντας μεγαλύτερη έμφαση σε γείτονες που είναι λιγότερο συνδεδεμένοι, δηλαδή τους πιο σπάνιους [12].

$$score_{[AA](u,v)}=\sum_{[z\in\Gamma(u)\cap\Gamma(v)]1}\frac{1}{\log(|\Gamma(z)|)}$$

$\Gamma(u)$, $\Gamma(v)$ αναπαριστούν τα σύνολα γειτόνων των κόμβων u και v αντίστοιχα, $\Gamma(u)\cap\Gamma(v)$ η τομή των γειτόνων των κόμβων u και v δηλαδή το σύνολο των κοινών γειτόνων τους [12]. Ο λογάριθμος του βαθμού ενός γειτονικού κόμβου z χρησιμοποείται ως δείκτης σπανιότητας του z. Η έκφραση $\frac{1}{\log(|\Gamma(z)|)}$ διασφαλίζει ότι:

- Εάν ο βαθμός Γ(z) είναι μεγάλος (κοινός γείτονας), η συμβολή είναι μικρή.
- Εάν ο βαθμός Γ(z) είναι μικρός (σπάνιος γείτονας), η συμβολή είναι μεγάλη [12].

Pearson Correlation Coefficient: Μετρά τη γραμμική σχέση μεταξύ δύο μεταβλητών [13].

$$\rho(X,Y)=\frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}$$, όπου:

- $\overline{x}, \overline{y}$: Μέσοι όροι των X και Y.
- x_i, y_i: Τιμές παρατηρήσεων.
- n: Πλήθος παρατηρήσεων.

Spearman: Μετρά τη σχέση μεταξύ δύο μεταβλητών χρησιμοποιώντας τις κατατάξεις τους (μη γραμμική συσχέτιση). Ο συντελεστής συσχέτισης Spearman's rank correlation coefficient (ρ) είναι ένα μη παραμετρικό μέτρο που ποσοτικοποιεί τη μονοτονική σχέση μεταξύ δύο μεταβλητών. Χρησιμοποιείται όταν τα δεδομένα είναι τουλάχιστον ταξινομικά (ordinal) και δεν απαιτεί καμία υπόθεση σχετικά με την κατανομή των δεδομένων. Χρησιμοποιείται κυρίως όταν τα δεδομένα δεν είναι κανονικά κατανεμημένα και θέλουμε να μετρήσουμε μη γραμμικές μονοτονικές σχέσεις [13].

$$\rho = 1 - \frac{6\sum d_i^2}{n\left(n^2-1\right)}$$

Η τιμή του κυμαίνεται από -1 έως 1. Όπου:

- d_i: Είναι η διαφορά μεταξύ των τάξεων (ranks) των αντίστοιχων ζευγών παρατηρήσεων για τις δύο μεταβλητές. Αν R_{x_i} είναι η κατάταξη της i-οστής παρατήρησης της μεταβλητής X και $R_{¿}$ η κατάταξη της i-οστής παρατήρησης της μεταβλητής Y, τότε $d_i = R_{x_i} - R_{y_i}$.
- $\sum d_i^2$: Είναι το άθροισμα των τετραγώνων αυτών των διαφορών στις κατατάξεις για όλες τις παρατηρήσεις.
- n : Είναι το πλήθος των παρατηρήσεων (ή το μέγεθος του δείγματος), δηλαδή ο αριθμός των ζευγών δεδομένων.

Αν $\rho = 1$ υπάρχει τέλεια θετική συσχέτιση (οι τάξεις των δύο μεταβλητών είναι ίδιες) και $\rho = -1$ τέλεια αρνητική συσχέτιση (οι τάξεις είναι αντιστρόφως ίδιες), ενώ ρ=0 καμία σχέση μεταξύ των δύο μεταβλητών [13].

Kendall: Αντίστοιχο του Spearman, αλλά χρησιμοποιεί μια διαφορετική μέθοδο για να υπολογίσει την αντιστοιχία μεταξύ των κατατάξεων. Ο συντελεστής συσχέτισης Kendall's Tau (τ) είναι ένα μέτρο συμφωνίας μεταξύ δύο κατατάξεων [13]. Υπολογίζεται ως:

$$\tau_{tau} = \frac{2\left(C-D\right)}{n\left(n-1\right)} \quad \text{, όπου :}$$

- C: ΟαΑριθμός των ζευγών ίδιας διάταξης.
- D: Ο αριθμός των ζευγών διαφορετικής διάταξης.
- n : Το μέγεθος του δείγματος [13].

Odds Ratio: Το odds ratio (λόγος πιθανοτήτων) είναι ένα μέτρο που χρησιμοποιείται στη στατιστική, κυρίως στην επιδημιολογία και στην κλινική έρευνα, για να ποσοτικοποιήσει τη σχέση μεταξύ μιας έκθεσης (exposure) και ενός αποτελέσματος (outcome) [14].

Ο μαθηματικός τύπος για το odds ratio (OR) ορίζεται ως: OR = $\frac{a*d}{b*c}$. Ο τύπος αυτός βασίζεται σε έναν πίνακα $2\,x\,2$, όπου:

a: ο αριθμός των ατόμων που είναι εκτεθειμένοι στον παράγοντα κινδύνου και έχουν την ασθένεια.

b: ο αριθμός των ατόμων που είναι εκτεθειμένοι στον παράγοντα κινδύνου και δεν έχουν την ασθένεια.

c: ο αριθμός των ατόμων που δεν είναι εκτεθειμένοι στον παράγοντα κινδύνου και έχουν την ασθένεια.

d: ο αριθμός των ατόμων που δεν είναι εκτεθειμένοι στον παράγοντα κινδύνου και δεν έχουν την ασθένεια [14].

Η βαρυμένη προβολή εν γένει μπορεί να χρησιμοποιήσει ποικίλους συντελεστές, οι οποίοι υπολογίζονται μεταξύ των ακμών, και να ανατεθεί ως βάρος σε μία ακμή ανάλογα και του ερευνητικού ερωτήματος που τίθεται στο εκάστοτε πρόβλημα. Για κάθε μετρική που ορίστηκε υπολογίζεται το μέτρο για κάθε ζεύγος στο δίκτυο και το νέο δίκτυο που προκύπτει έχει τις ίδιες ακμές με το αρχικό, όμως με την πληροφορία του βάρους που υπολογίστηκε για κάθε ακμή. Για την ανάλυσή μας χρησιμοποιήθηκε ως προτεύων το μέτρο του αριθμού κοινών γειτόνων καθώς στην βιβλιογραφία, οι περισσότερες μέθοδοι εξαγωγής κορμού λειτουργούν με βαρυμένα δίκτυα που χρησιμοποιούν ως βάρος αυτό το μέτρο.

3.4 Υπεργραφήματα (Hypergraphs)

Τα υπεργραφήματα αποτελούν μια γενίκευση των συμβατικών γραφημάτων, με τη θεμελιώδη διαφορά ότι είναι ικανά να αναπαριστούν αλληλεπιδράσεις ανώτερης τάξης, οι οποίες συχνά απαντώνται σε πολύπλοκα συστήματα. Αυτό επιτρέπει μια πιο εκφραστική μορφή κωδικοποίησης για σχέσεις ανώτερης τάξης. Σε αντίθεση με τα απλά γραφήματα, όπου οι ακμές συνδέουν μόνο δύο κόμβους, οι υπερακμές (hyperedges) στα υπεργραφήματα μπορούν να συνδέουν περισσότερους από δύο κόμβους. Αυτή η ικανότητα τα καθιστά κατάλληλα για τη μοντελοποίηση φαινομένων σε πολλά σενάρια όπως κοινωνικές αλληλεπιδράσεις, βιολογικές διεργασίες ή συνεργασίες επιστημόνων σε άρθρα. Όταν αυτές οι πολύπλοκες σχέσεις συμπιέζονται αναγκαστικά σε σχέσεις ανά ζεύγη, οδηγούμαστε σε αναπόφευκτη απώλεια πληροφορίας. Αυτή η απώλεια πληροφορίας μπορεί να είναι κρίσιμη, καθώς οι ομαδικές αλληλεπιδράσεις συχνά εξηγούν τη δομή και την τοπική πυκνότητα των παρατηρούμενων δικτύων [15].

Αξιοποιώντας αυτήν την ικανότητα αναπαράστασης, τα υπεργραφήματα αποτελούν το θεμελιώδες πλαίσιο για την αντιμετώπιση κεντρικών προβλημάτων μηχανικής μάθησης. Η ανίχνευση κοινοτήτων (clustering), που στοχεύει στην εύρεση κρυφών ομάδων παρόμοιων κόμβων, καθώς και η ταξινόμηση (classification) και η ενσωμάτωση (embedding), αποτελούν πλέον εφαρμόσιμες εργασίες. Η ποιότητα των αποτελεσμάτων ενισχύεται σημαντικά όταν, πέραν των δομικών ιδιοτήτων του υπεργραφήματος, ενσωματώνεται πρόσθετη πληροφορία, όπως τα χαρακτηριστικά των κόμβων (covariate information). Η επίλυση αυτών των προβλημάτων απαιτεί την ανάπτυξη εξειδικευμένων μαθηματικών μοντέλων και την αντίστοιχη αλγοριθμική υλοποίηση για την εξαγωγή συμπερασμάτων (inference) σε σύνθετα δεδομένα.

Ένα διμερές δίκτυο μπορεί να αναπαρασταθεί ως υπεργράφημα (hypergraph), όπου οι κόμβοι του ενός συνόλου αντιστοιχούν σε υπερακμές που συνδέουν τους κόμβους του άλλου συνόλου. Έπειτα, εφαρμόζονται τεχνικές υπεργραφημάτων για την εξαγωγή μιας μονομερούς αναπαράστασης.

Ορισμός 3.3

Ένα υπεργράφημα H είναι ένα ζεύγος (V, E):

- V είναι ένα σύνολο κόμβων.
- $E = [E1, E2, ..., Em]$ είναι ένα σύνολο συνόλων με $Ei \subseteq V$ για όλα τα $1 \leq i \leq m$.

Για τους σκοπούς της ανακατασκευής, υποθέτει ότι οι υπερακμές είναι διακριτές, δηλαδή $\forall 1 \leq i, j \leq m, E_i \neq E_j$

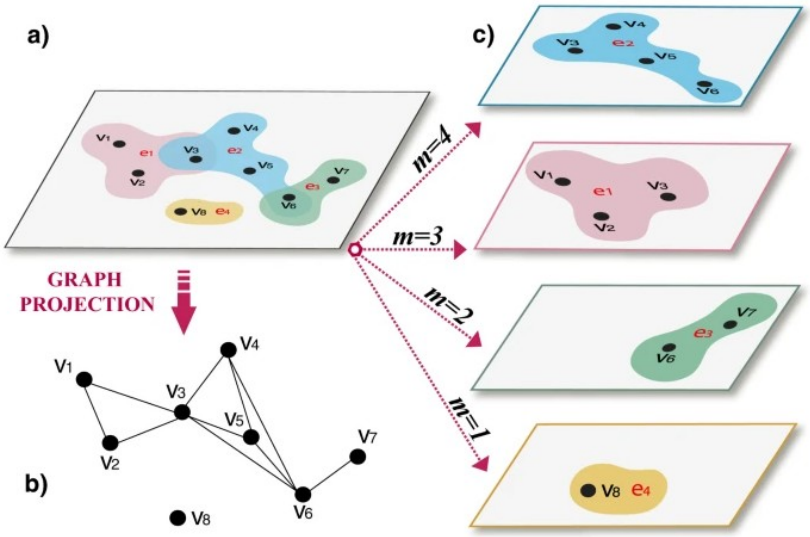
3.4.1 Υλοποίηση Υπεργραφημάτων

Τα υπεργραφήματα κατασκευάστηκαν μέσω μιας διαδικασίας που περιλαμβάνει την ανάγνωση δεδομένων, την κατασκευή ενός γράφου και τη δημιουργία κοινοτήτων γονιδίων. Στην αρχή, τα δεδομένα διαβάστηκαν από ένα αρχείο, το οποίο δημιουργήθηκε ώστε να περιέχει υπερακμές (σύνολα κόμβων που συνδέονται μεταξύ τους). Χρησιμοποιώντας την βιβλιοθήκη NetworkX, δημιουργήθηκε ένας γράφος, όπου κάθε γραμμή του αρχείου αντιστοιχούσε σε ακμές μεταξύ ενός

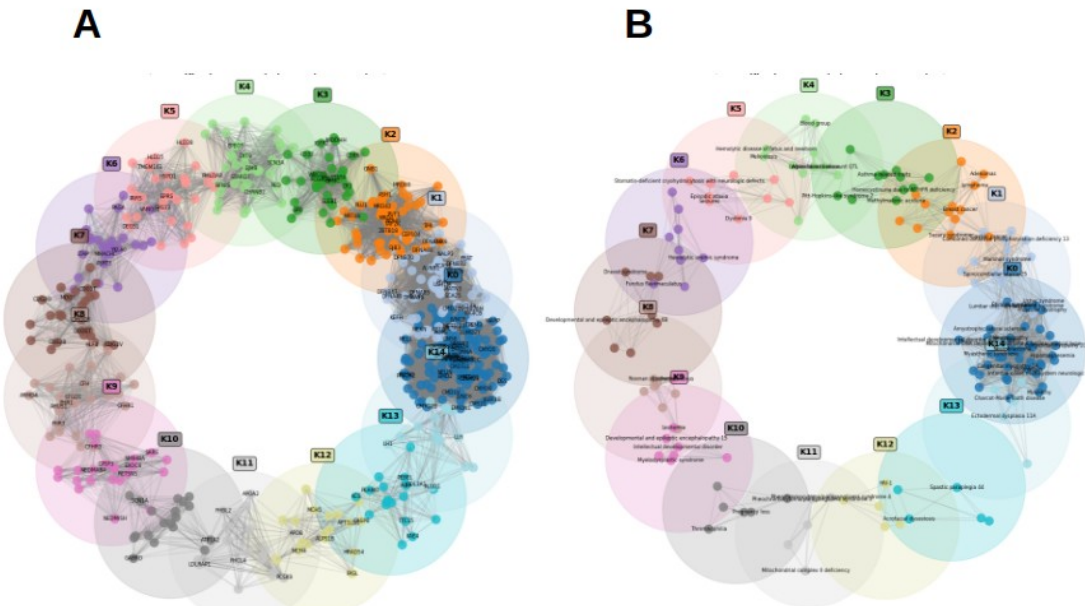
κόμβου με τους γείτονες του. Στη συνέχεια, με τη βοήθεια της συνάρτησης nx.connected_components(), εντοπίστηκαν οι συνδεδεμένες κοινότητες στον γράφο, οι οποίες αντιστοιχούν σε ομάδες γονιδίων που συνδέονται μέσω των υπερακμών.

Στο δεύτερο βήμα, δημιουργήθηκε μια κυκλική διάταξη για τις κοινότητες, χρησιμοποιώντας την matplotlib για την οπτικοποίηση. Κάθε κοινότητα τοποθετήθηκε γύρω από ένα κεντρικό σημείο σε κυκλικό διάταγμα, και τα γονίδια της κοινότητας τοποθετήθηκαν τυχαία γύρω από το κέντρο της. Χρησιμοποιώντας την create_community_layout(), δημιουργήθηκε ο γράφος με τις ακμές μεταξύ των γονιδίων εντός κάθε κοινότητας, και ορίστηκαν τυχαίες θέσεις για τους κόμβους γύρω από το κέντρο της κοινότητας. Τέλος, η οπτικοποίηση του υπεργράφου έγινε με την συνάρτηση visualize_communities_circular(), η οποία εμφάνισε τις κοινότητες με διαφορετικά χρώματα και τοποθέτησε ετικέτες για τα γονίδια, δημιουργώντας μια καθαρή και κατανοητή αναπαράσταση της δομής των κοινοτήτων γονιδίων.

Το πρόγραμμα εφαρμόστηκε και για τα δύο one mode projection δίκτυα ενώ εφαρμόστηκε ιεράρχιση των συνιστωσών σε φθίνουσα σειρά με βάση των αριθμό κόμβων που συμμετέχουν και διατηρήθηκαν οι 15 μεγαλύτερες. Η μεγαλύτερη συνιστώσα του υπεργραφήματος γονιδίων περιείχε 141 γονίδια, ενώ μεγαλύτερη συνιστώσα στο δίκτυο ασθενειών περιείχε 40 ασθένειες (βλ. Εικόνα 3.2).



Εικόνα 3.2 Η εικόνα παρουσιάζει τρεις αναπαραστάσεις ενός υπεργράφου με σύνολο κόμβων $V = v_1, .., v_8\}$ και σύνολο υπερακμών $E = e_1, e_2, e_3, e_4¿\}$, όπου $e_1 = v_1, v_2, v_3$, $e_2 = v_3, v_4, v_5, v_6$, $e_3 = v_6, v_7$ και $e_4 = v_8$. (a) Ο αρχικός υπεργράφος, όπου οι υπερακμές (χρωματισμένες περιοχές) συνδέουν διαφορετικούς αριθμούς κόμβων. Αυτή η αρχική δομή μπορεί να αναπαρασταθεί είτε μέσω της προβολής (b) του γράφου. (c) Η αποσύνθεση που διαχωρίζει τον υπεργράφο σε επίπεδα m-ομοιόμορφων υπεργράφων (χρωματικά κωδικοποιημένα), όπου m αντιστοιχεί στο μέγεθος (πληθικότητα) της υπερακμής [16].



Εικόνα 3.3 Αναπαράσταση Υπεργραφήματος των δικτύων one mode. (A) Αναπαράσταση του φιλτραρισμένου υπεργραφήματος των γονιδίων, αναδεικνύοντας τις 10 μεγαλύτερες κοινότητες $(K_1 - K_{10})$ που προέκυψαν από τον αλγόριθμο Louvain. (B) Αναπαράσταση του φιλτραρισμένου υπεργραφήματος των ασθενειών, αναδεικνύοντας τις 10 μεγαλύτερες κοινότητες $(K_1 - K_{10})$ που προέκυψαν από τον αλγόριθμο Louvain.

Κεφάλαιο 4

4.1 Εξαγωγή Σκελετού Διμερών Δικτύων

Οι ιδιότητες που χαρακτηρίζουν τα διμερή δίκτυα όπως περιγράφτηκαν σε συνδιασμό με την υψηλή δημοτικότητα που απέκτησαν, τα τελευταία 20 έτη, τα καθίστούν απαραίτητα για την ανάλυση και οπτικοποίηση πολύπλοκων συστημάτων. Ωστόσο επειδή τα προβλήματα που αναπαριστούν πολλές φορές δεν είναι καλώς ορισμένα αλλά και λόγω του αυξανόμενου μεγέθους τους απαιτούνται τεχνικές εξαγωγής ραχοκοκαλιάς του δικτύου (backbone extraction methods) που στοχεύουν στη μείωση του μεγέθους τους διατηρώντας παράλληλα τα πιο κρίσιμα χαρακτηριστικά.

Η εξαγωγή του σκελετού των δικτύων έχει πολλές εφαρμογές, όπως η ανάλυση κοινωνικών, βιολογικών, γενετικών, οικονομικών, εμπορικών, πολιτικών και μεταφορικών δικτύων μεταξύ άλλων. Πλεονεκτήματα της εξαγωγής κορμού αφορούν την μείωση του όγκου δεδομένων και του απαιτούμενου αποθηκευτικού χώρου, καθιστώντας ταυτόχρονα τους αλγόριθμους γραφημάτων και τα ερωτήματα πολύ ταχύτερα. Παράλληλα, διευκολύνει την υποστήριξη διαδραστικής ανάλυσης για τον χρήστη, ενώ το κυριότερο όφελος είναι η εξάλειψη του θορύβου, επιτρέποντας στους ερευνητές να επικεντρωθούν στα κρίσιμα και ουσιώδη χαρακτηριστικά του δικτύου.. Μία ευρέως γνωστή και απλούστερη μέθοδος αποτελεί η μέθοδος απλής κατωφλίωσης **(Global Threshold Filter)** όπου οι τιμές με βάρος μεγαλύτερο η μικρότερο του επιλεγμένου κατωφλίου απορρίπτονται από το δίκτυο [17], [18], [19].

Παρόλο που η μελέτη στην παρούσα εργασία επικεντρώνεται στην εξαγωγή σκελετού διμερών δικτύων ειναι χρήσιμο να σημειωθεί πως οι μέθοδοι αυτές βρίσκουν σημαντικές εφαρμογές στην ομαδοποίηση (clustering), ταξινόμηση (classification), ανίχνευση κοινοτήτων (community detection), ανίχνευση ανωμαλιών (outlier detection), οπτικοποίηση μεγάλων δικτύων καθώς και εντοπισμό θορύβου σε δίκτυα. Στην περίπτωση της οπτικοποίησης μεγάλων δικτύων, οι μέθοδοι προσφέρουν ένα αποτελεσματικό εργαλείο για την οπτική αναπαράσταση σύνθετων δικτύων, καθιστώντας τα ευκολότερα στην κατανόηση και ανάλυση. Τέλος, οι τεχνικές που θα αναλυθούν μπορούν να χρησιμοποιηθούν για τον εντοπισμό θορύβου σε δίκτυα, απομονώνοντας αχρείαστες ή λανθασμένες πληροφορίες που μπορεί να επηρεάσουν την ανάλυση των δεδομένων. Όλες αυτές οι εφαρμογές καθιστούν τη μέθοδο πολύτιμη σε διάφορους τομείς της επιστήμης των δεδομένων και της ανάλυσης δικτύων [17].

Ανάλογα την μέθοδο διακρίνονται τρεις κατηγορίες:

- Στατιστικές μέθοδοι (Statistical Methods)**:** Χρησιμοποιούν ελέγχους υποθέσεων για να εντοπίσουν σημαντικές ακμές και διαγράφουν ακμές που χαρακτηρίζονται ως θόρυβος [17].
- Δομικές μέθοδοι (Structural Methods) **:** Εστιάζουν στα τοπολογικά χαρακτηριστικά του δικτύου ενώ διατηρούν ακμές βάσει δομικών κριτηρίων, όπως η κεντρικότητα ή η συνδεσιμότητα [17].
- Υβριδικές μέθοδοι (Hybrid Methods) **:** Συνδυάζουν στοιχεία στατιστικών και δομικών μεθόδων [17].

4.2 Ιεραρχική Εξαγωγή σκελετού του δικτύου

Η χρήση ενός ιεραρχικού backbone στη μελέτη πολυπαραγοντικών ασθενειών και των σχετικών γονιδίων θα είχε νόημα για διάφορους λόγους. Αρχικά, θα μπορούσε να ανιχνεύσει νέες, μη καταγεγραμμένες συσχετίσεις, εντοπίζοντας έμμεσες αλλά σημαντικές σχέσεις μεταξύ ασθενειών και γονιδίων που δεν έχουν ακόμα καταγραφεί. Η μέθοδος αναγνωρίζει ιεραρχικές σχέσεις σε ένα δίκτυο , εξετάζοντας αν δύο κόμβοι συνεμφανίζονται πολύ περισσότερο από το αναμενόμενο και αν υπάρχει ασυμμετρία στις πιθανότητες εμφάνισής τους. Έτσι, μπορούμε κατασκευάζεται μια ιεραρχία όπου οι πιο γενικές έννοιες βρίσκονται ψηλότερα και οι πιο εξειδικευμένες χαμηλότερα [20]. Για παράδειγμα, αν ένα γονίδιο A σχετίζεται με μια ασθένεια X και ένα γονίδιο B συνδέεται με το A, τότε είναι πιθανό το B να σχετίζεται και με την X. Επιπλέον, η μέθοδος θα μπορούσε να συμβάλει στην οργάνωση της γνώσης σε ιεραρχικά επίπεδα, δημιουργώντας μια πιο κατανοητή χαρτογράφηση των ασθενειών και των γονιδίων. Ασθένειες που μοιράζονται κοινά γονίδια μπορεί να οργανωθούν σε φυσικά clusters, αποκαλύπτοντας λειτουργικές σχέσεις.

Οι σχέσεις μεταξύ γονιδίων και ασθενειών είναι ιεραρχικές. Κάποια γονίδια εμπλέκονται σε γενικότερες μεταβολικές διεργασίες που οδηγούν σε παθολογία, ενώ άλλα έχουν πιο εξειδικευμένες επιδράσεις. Για παράδειγμα, έστω ότι το γονίδιο Z είναι ένας βασικός ογκοκατασταλτικός παράγοντας που επηρεάζει πολλούς τύπους καρκίνου, ενώ το γονίδιο Y σχετίζεται περισσότερο με καρκίνο του μαστού και των ωοθηκών. Η βασική ιδέα της μεθόδου είναι πως μέσω της επαγωγής μπορούμε να αποδείξουμε ότι το γονίδιο $Z \rightarrow Y$, το Y δηλαδή αποτελεί υποκατηγορία του Z ως προς τον μηχανισμό λειτουργίας τους [20].

Δεδομένου οτι διαθέτουμε δύο κόμβους u και v, η μέθοδος εξάγει έναν κορμό, διερευνώντας αν ισχύει η ακόλουθη επαγωγή $u \rightarrow v$ δηλαδή ότι ο κόμβος v είναι υποκατηγορία του u [20]. Για να απαντηθεί το ερώτημα αυτό πρέπει να ισχύουν τα ακόλουθα δύο κριτήρια :

- **Σημαντική συν-εμφάνιση** $N(u,v)$

  $N(u,v)$ είναι ο αριθμός των αντικειμένων που έχουν και τις δύο ετικέτες u και v. Αν αυτή η συνεμφάνιση είναι πολύ μεγαλύτερη από την αναμενόμενη $EN(u,v)$, δηλαδή τη μέση τιμή σε ένα τυχαίο δίκτυο όπου οι βαθμοί των κόμβων διατηρούνται, τότε υπάρχει κάποια ισχυρή σύνδεση μεταξύ τους [20].

- **Ασυμμετρία στις πιθανότητες συσχέτισης**

  $P(u|v)=\frac{N(u,v)}{N(v)}$ είναι η πιθανότητα να έχει ένα αντικείμενο την ετικέτα u, δεδομένου ότι έχει την ετικέτα v, και $P(v|u)=\frac{N(u,\mathrm{v})}{N(u)}$ η πιθανότητα να έχει ένα αντικείμενο την ετικέτα v, δεδομένου ότι έχει την ετικέτα u [20].

Αν η τιμή $P(u|v)$ είναι πολύ μεγαλύτερη από την τιμή $P(v|u)$, σημαίνει ότι όταν εμφανίζεται ο κόμβος v, σχεδόν πάντα εμφανίζεται και ο κόμβος u, αλλά το αντίστροφο δεν ισχύει τόσο συχνά. Αυτό συνεπάγεται ότι u είναι μια γενικότερη κατηγορία και v είναι υποκατηγορία της [20].

Το αρχικό δίκτυο B έχει δύο τύπους κόμβων γονίδια και ασθένειες. Για να μελετήσουμε μόνο τις σχέσεις μεταξύ ετικετών του ενός συνόλου, το πρώτο βήμα είναι να προβάλλουμε το δίκτυο σε ένα μονομερές δίκτυο γονιδίων G, όπου οι κόμβοι είναι μόνο οι ετικέτες γονιδίων, οι συνδέσεις μεταξύ ετικετών δείχνουν πόσο συχνά εμφανίζονται μαζί στα ίδια αντικείμενα. Δηλαδή, δημιουργούμε ένα νέο δίκτυο όπου κάθε γονίδιο συνδέεται με άλλα γονίδια αν εμφανίζονται μαζί στις ίδιες ασθένειες.

Αφού έχουμε το νέο δίκτυο G, θέλουμε να διατηρήσουμε μόνο τις σημαντικές συνδέσεις μεταξύ των ετικετών. Για να επιτευχθεί αυτό, χρησιμοποιούμε το z-score της παρατηρούμενης συνεμφάνισης $N(u,v)$, δηλαδή πόσο διαφέρει η πραγματική συνεμφάνιση από αυτήν που θα περιμέναμε σε ένα τυχαίο δίκτυο. Ο υπολογισμός βασίζεται στο μοντέλο Bipartite Configuration Model (BiCM), όπου διατηρούνται οι βαθμοί των κόμβων. Η πιθανότητα συνεμφάνισης ακολουθεί μια υπεργεωμετρική κατανομή (hypergeometric distribution) [20].

Για να υπολογιστεί η αναμενόμενη συνεμφάνιση ορίζουμε, $N(u)$ το πλήθος αντικειμένων που έχουν την ετικέτα *u* και $N(v)$ το πλήθος αντικειμένων που έχουν την ετικέτα v και *O* τον συνολικό αριθμό αντικειμένων.

Η αναμενόμενη συνεμφάνιση m αφορά τον αναμενόμενο αριθμό αντικειμένων που έχουν και τις δύο ετικέτες u , v δίνεται από τον τύπο:

$$m = \frac{N(u)\cdot N(v)}{[O]}$$

Η διασπορά της (variance) σ^2 ορίζεται:

$$\sigma^2 = \frac{N(u)\cdot N(v)}{[O]}\cdot\frac{[O]- N(u)}{[O]}\cdot\frac{[O] - N(v)}{[O]-1}$$

όπου σ είναι η τυπική απόκλιση (standard deviation), δηλαδή πόσο μπορεί να αποκλίνει αυτός ο αριθμός λόγω τυχαιότητας. Αν η πραγματική συνεμφάνιση $N(u,v)$ είναι πολύ μεγαλύτερη από την αναμενόμενη *m*, τότε η σύνδεση μεταξύ των δύο ετικετών είναι σημαντική. Έπειτα αφαιρούμε τις συνδέσεις για τις οποίες το z-score δεν ξεπερνάει ένα όριο (z-threshold), δηλαδή δεν είναι στατιστικά σημαντικές [20]..

Από το μονομερές δικτύο G, αφαιρούνται οι ασήμαντες συνδέσεις, δημιουργώντας το δίκτυο G_{pruned}. Σε αυτό το σημείο, μετράται εαν μία ετικέτα u είναι ανώτερη ιεραρχικά από μια άλλη ετικέτα v, δηλαδή αν ισχύει η σχέση $u \rightarrow v$.

Για τον σκοπό αυτό γίνεται χρήση ενός δείκτη ιεραρχικής ισχύος (hierarchical strength) ο οποίος ορίζεται ως:

$$\left(\left(\alpha_{u->v} = \frac{min(k_u, k_v)}{k_{max}}\right)\cdot\left[\frac{N(u,v)}{N(v)} - \frac{N(u,v)}{N(u)}\right]\right)$$

με k_u να είναι ο βαθμός της ετικέτας u στο περικομμένο δίκτυο G_{pruned}, k_v ο βαθμός της ετικέτας v στο περικομμένο δίκτυο Gpruned, k_{max} ο μέγιστος βαθμός που παρατηρείται σε οποιαδήποτε ετικέτα στο Gpruned [20].

Τελικό στάδιο αυτής της μεθόδου αποτελεί η εύρεση της κατεύθυνσης της ιεραρχικής σχέσης. Η ιεραρχική σχέση $u \to v$ (δηλαδή, ο κόμβος u ανήκει σε μία γενικότερη κατηγορία από τον κόμβο v, ή αλλιώς ο v είναι υποκατηγορία του u) θεωρείται ότι υπάρχει αν η συνθήκη $\alpha_{u \to v} > 0$ είναι αληθής. Από το G_{pruned} διατηρούνται μόνο οι κατευθυνόμενες ακμές (u, v) για τις οποίες: $\alpha_{u \to v} \geq \alpha_{th}$ όπου α_{th} είναι ένα προκαθορισμένο όριο. Το προκύπτον δίκτυο είναι ένα κατευθυνόμενο ακυκλικό γράφημα (DAG) [20].

4.3 Bipartite Configuration model

Το Bipartite Configuration Model (BiCM), ή Μοντέλο Διαμόρφωσης Διμερούς Γραφήματος, είναι ένα στατιστικό μηδενικό μοντέλο για δυαδικά διμερή δίκτυα, δηλαδή χωρίς βάρος στο δίκτυο εισόδου. Σε ένα στατιστικό πλαίσιο, ένα μηδενικό μοντέλο είναι ένα μοντέλο αναφοράς που χρησιμοποιείται, για να αξιολογηθεί εάν τα παρατηρούμενα χαρακτηριστικά ενός δικτύου είναι στατιστικά σημαντικά ή αν μπορούν να αποδοθούν σε τυχαία ενδεχόμενα. Πιο συγκεκριμένα η μέθοδος παράγει τυχαία διμερή δίκτυα διατηρώντας τα βαθμούς των κόμβων (degrees of nodes) του αρχικού, παρατηρούμενου διμερούς δικτύου. Δημιουργεί μια βάση τυχαιότητας επιτρέποντας να απαντηθεί το ερώτημα αν παρατηρούμενη σύνδεση μεταξύ του κόμβου U και του κόμβου V είναι πιο ισχυρή από ό,τι θα αναμενόταν τυχαία, δεδομένων των συνολικών συνδέσεων που έχουν ο U και ο V. Παράλληλα βοηθάει στον εντοπισμό των στατιστικά σημαντικών συνδέσεων, αφαιρώντας τον θόρυβο που οφείλεται μόνο στους υψηλούς βαθμούς των κόμβων. Αυτό οδηγεί στην εξαγωγή του backbone του δικτύου [21].

Οι βαθμοί των κόμβων του παρατηρούμενου δικτύου ορίζονται ως:

Βαθμός κόμβου r στο σύνολο R : $$k_r = \sum_{C=1}^{N_C} M_{rc}$$

Βαθμός κόμβου cστο σύνολο C: $$h_c = \sum_{r=1}^{N_r} M_{rc}$$

Κατασκευάζονται μονοδιάστατα διανύσματα με τις ακολουθίες βαθμών $d(M)$, $u(M)$ των κόμβων των δύο συνόλων και γίνεται χρήση της συνάρτησης Hamiltonian.

$$H(M|\alpha, \beta) = \alpha \cdot k_r + \beta \cdot h_c,$$

όπου α, β είναι οι πολλαπλασιαστές Lagrange που αντιστοιχούν στους περιορισμούς των βαθμών για τα δύο σύνολα κόμβων αντίστοιχα [21]. Παράλληλα δίνει μια αναλυτική έκφραση για την πιθανότητα κάθε δικτύου $P(M)$ στο σύνολο, με βάση τις συγκεκριμένες ακολουθίες βαθμών με τον τύπο :

$$P(M|x, y) = \prod_{cp} \left(p_{cp}{}^{m_{cp}}\right) \cdot \left(1 - \left(p_{cp}{}^{1-m_{cp}}\right)\right)$$

Η πιθανότητα ύπαρξης σύνδεσης μεταξύ του κόμβου r και του κόμβου c στο BiCM, δίνεται από τον τύπο: $$p_{rc} = \frac{x_r y_c}{1 + x_r y_c}$$

όπου x_r και y_c είναι οι πολλαπλασιαστές Lagrange που αντιστοιχούν στους κόμβους r και c, αντίστοιχα. Αυτοί οι πολλαπλασιαστές υπολογίζονται έτσι ώστε οι αναμενόμενοι βαθμοί του μοντέλου να ταιριάζουν με τους παρατηρούμενους βαθμούς του δικτύου [21].

Οι πολλαπλασιαστές x_r και y_c βρίσκονται λύνοντας το ακόλουθο σύστημα μη γραμμικών εξισώσεων:

- $k_r = \sum_{C=1}^{N_C} \frac{x_r y_c}{1 + x_r y_c}$ για κάθε $r \in 1, \ldots, N_r$
- $h_c = \sum_{r=1}^{N_r} \frac{x_r y_c}{1 + x_r y_c}$ για κάθε $c \in 1, \ldots, N_C$

Αφού υπολογιστούν οι πιθανότητες p_{rc} για κάθε ζεύγος κόμβων (r, c), μπορούμε να αξιολογήσουμε τη στατιστική σημαντικότητα των παρατηρούμενων συνδέσεων. Η τιμή p_{rc} αντιπροσωπεύει την πιθανότητα να υπάρχει μια σύνδεση μεταξύ r και c τυχαία, δεδομένων των συνολικών βαθμών τους.

Για να προσδιοριστεί εάν μια παρατηρούμενη σύνδεση M_{rc} είναι στατιστικά σημαντική, μπορούμε να χρησιμοποιήσουμε έναν έλεγχο υποθέσεων. Συνήθως, εξετάζουμε εάν η παρατηρούμενη σύνδεση είναι ισχυρότερη από αυτή που θα αναμενόταν τυχαία. Για παράδειγμα, μια σύνδεση M_{rc} θεωρείται στατιστικά σημαντική εάν: $M_{rc} > p_{rc} + Z_{\frac{a}{2}} \sqrt[2]{2\ p_{rc}\ (1 - p_{rc})}$ [21].

Η εφαρμογή του BiCM επιτρέπει την εξαγωγή του στατιστικά σημαντικού backbone του διμερούς δικτύου. Αυτό σημαίνει ότι διατηρούμε μόνο τις συνδέσεις που είναι στατιστικά πιο ισχυρές από ό,τι θα αναμενόταν τυχαία. Το τελικό, δίκτυο περιέχει μόνο τις πιο ουσιαστικές και μη τυχαίες αλληλεπιδράσεις, παρέχοντας μια πιο ακριβή εικόνα των υποκείμενων δομών [21].

4.4 Stochastic Degree Sequence Model

Ένα επίσης ισχυρό στατιστικό μοντέλο μηδενικής υπόθεσης που χρησιμοποιείται στην εξαγωγή κορμού διμερών δικτύων ειναι το Stochastic Degree Sequence Model (SDSM) [22], [23]. Η βασική ιδέα του μοντέλου είναι ότι εισάγοντας ένα διμερές δίκτυο (π.χ., γονίδια συνδεδεμένα με ασθένειες) προσπαθεί να κατασκευάσει ένα τυχαίο μοντέλο που να διατηρεί ορισμένες από τις βασικές του ιδιότητες, συνήθως τους βαθμούς των κόμβων όπως το BiCM. Στόχος του δεν είναι να αναπαράγει το ακριβές παρατηρούμενο δίκτυο, αλλά να δημιουργήσει πολλούς πιθανούς δυαδικούς πίνακες B που θα μπορούσαν να προκύψουν αν οι συνδέσεις γίνονταν με βάση τους βαθμούς των κόμβων, αλλά τυχαία [22]. Αυτό επιτρέπει να διακρίνουμε ποιες συνδέσεις στο πραγματικό δίκτυο είναι στατιστικά σημαντικές (δηλαδή, εμφανίζονται πολύ συχνότερα ή σπανιότερα από το αναμενόμενο βάσει της τυχαιότητας που ορίζει το μοντέλο). Αυτές οι στατιστικά σημαντικές συνδέσεις αποτελούν το backbone του δικτύου.

Το SDSM θεωρεί ως πιθανούς όλους τους δυαδικούς πίνακες $m \times n$ (όπου m είναι ο αριθμός των κόμβων της μιας ομάδας, π.χ. γονίδια, και n ο αριθμός των κόμβων της άλλης ομάδας, π.χ. ασθένειες). Δημιουργεί αυτούς τους πίνακες συμοληρώνοντας κάθε κελί B_{ik} (κάθε πιθανή σύνδεση μεταξύ του κόμβου i της πρώτης ομάδας και του κόμβου k της δεύτερης) με ένα 0 ή ένα 1. Η απόφαση για το αν

θα είναι 0 ή 1 βασίζεται στην έκβαση μιας ανεξάρτητης δοκιμής Bernoulli με μια συγκεκριμένη πιθανότητα p_{ik} Το κρίσιμο σημείο είναι ο τρόπος επιλογής του p_{ik}. Επιλέγεται έτσι ώστε να προσεγγίζει την πιθανότητα $Pr(B_{ik}=1)$ που θα είχε μια σύνδεση αν οι βαθμοί των κόμβων διατηρούνταν κατά μέσο όρο. Δηλαδή, δεν απαιτείται οι περιορισμοί στους βαθμούς να ικανοποιούνται σε κάθε μεμονωμένο τυχαίο δίκτυο, αλλά ο μέσος όρος των βαθμών σε όλους τους πιθανούς κόσμους να ταιριάζει με τους παρατηρούμενους [23].

Η μέθοδος ξεκινά με την επιλογή μεθόδου για τον υπολογισμό των πιθανοτήτων p_{ik}**.** Αυτό είναι το πιο κρίσιμο βήμα που διαφοροποιεί τις διάφορες παραλλαγές του SDSM. Με p_{ik} να είναι η πιθανότητα να υπάρχει σύνδεση μεταξύ του κόμβου i και του κόμβου k σε έναν τυχαία παραγόμενο πίνακα. Ο υπολογισμός των πιθανοτήτων πραγματοποιείται με τρεις τρόπους:

- Αριθμητική Μέθοδος RCF/Chung-Lu

  $pik_{*} = \frac{r_i \times c_k}{f}$, με r_i ο βαθμός (αριθμός συνδέσεων) του κόμβου i της πρώτης ομάδας, c_k ο βαθμός (αριθμός συνδέσεων) του κόμβου k του δεύτερου συνόλου, f ο συνολικός αριθμός των ακμών στο αρχικό δίκτυο (ή το άθροισμα όλων των r_i ή c_k). Αυτή η τιμή πρέπει να περιοριστεί στο διάστημα [0,1] [23].

- Γενικευμένα Γραμμικά Μοντέλα

  Αυτή η κατηγορία προσεγγίζει την πιθανότητα p_{ik} μέσω της προσαρμογής ενός στατιστικού μοντέλου. Η βασική ιδέα είναι να προβλέψουμε την ύπαρξη μιας σύνδεσης (B_{ik}) χρησιμοποιώντας τους βαθμούς των εμπλεκόμενων κόμβων (r_i και c_k) ως προβλεπτικές μεταβλητές [23]. Συγκεκριμένα έχουν προταθεί δύο κύριες παραλλαγές:

  α) Γραμμικό Μοντέλο

  $B_{ik} = \beta_0 + \beta_1 r_i + \beta_2 c_k + \varepsilon$ B_{ik} η (0 ή 1) που δηλώνει την ύπαρξη σύνδεσης, $\beta_0, \beta_1, \beta_2$ οι συντελεστές που εκτιμώνται (π.χ., με ελάχιστα τετράγωνα - Least Squares). , r_i Ο βαθμός της γραμμής i, c_k ο βαθμός της στήλης k. και ϵ ο όρος σφάλματος [23].

  β) Λογιστική Παλινδρόμηση (Logistic Regression)

  Είναι πιο κατάλληλη μέθοδος για δυαδικά δεδομένα, καθώς μοντελοποιεί άμεσα την πιθανότητα ενός γεγονότος. Χρησιμοποιεί μια συνάρτηση (link function) για να μετατρέψει τη συνάρτηση των μεταβλητών σε πιθανότητα μεταξύ 0 και 1 [23].

$$\ln\left(\frac{p_{ik}}{1-p_{ik}}\right) = \beta_0 + \beta_1 r_i + \beta_2 c_k + \varepsilon$$

Οι συντελεστές β εκτιμώνται συνήθως χρησιμοποιώντας μέγιστη πιθανοφάνεια (maximum likelihood) [23]. Οι μέθοδοι Μέθοδοι Μεγιστοποίησης Εντροπίας (Entropy Maximization Methods) μεταξύ άλλων είναι θεωρητικά πιο αυστηρές [23]. Στοχεύουν στη δημιουργία ενός τυχαίου μοντέλου που έχει μέγιστη εντροπία, διατηρώντας ταυτόχρονα ορισμένους δομικούς περιορισμούς του αρχικού δικτύου. Οι δύο κύριες μέθοδοι που ανήκουν σε αυτή την κατηγορία είναι:

i) Η Polytope Method, η οποία βασίζεται στην εύρεση μιας πιθανότητας κατανομής πάνω στο σύνολο όλων των πιθανών δυαδικών πινάκων που έχουν τους δοθέντες βαθμούς γραμμών και στηλών. Αυτό είναι ένα υπολογιστικά έντονο πρόβλημα, ειδικά για μεγάλα δίκτυα και ο στόχος είναι να βρεθούν οι p_{ik} έτσι ώστε το μοντέλο να είναι όσο το δυνατόν πιο τυχαίο, αλλά οι μέσοι βαθμοί των κόμβων να συμφωνούν ακριβώς με τους παρατηρούμενους βαθμούς [23].

ii)Το Διμερές Μοντέλο Διαμόρφωσης (Bipartite Configuration Model – BiCM), όπου αναλύθηκε εκτενώς ανωτέρω [23].

4.5 Η Μέθοδος Disparity Filter

Ο σκοπός της συγκεκριμένης μεθόδου είναι να εξάγει το multiscale backbone (πολυκλιμακωτός σκελετός) του δικτύου, διατηρώντας ακμές με βάρη που δεν προκύπτουν από τυχαίες κατανομές, αλλά αντανακλούν σημαντικές ετερογένειες λόγω των αρχών οργάνωσης του δικτύου [24]. Η φιλοσοφία της έγκειται στην τοπική σύγκριση, καθώς το κριτήριο εφαρμόζεται σε κάθε κόμβο ξεχωριστά [24]. Οι παρατηρούμενες τιμές συγκρίνονται με το μοντέλο μηδενικής υπόθεσης (null model) και οι ακμές φιλτράρονται στο επίπεδο σημαντικότητας α. Επειδή το null model εφαρμόζεται σε κάθε κόμβο ξεχωριστά, μια ακμή μπορεί να είναι σημαντική από τη σκοπιά του ενός κόμβου, αλλά όχι του άλλου [24], [25]. Η μέθοδος αυτή είναι ιδιαίτερα χρήσιμη για την ανάλυση σταθμισμένων δικτύων, όπου επιδιώκουμε να διατηρήσουμε μόνο τις πιο σημαντικές σχέσεις, απορρίπτοντας εκείνα που προκύπτουν από τυχαίες κατανομές [24].

Η μηδενική υπόθεση υποθέτει ότι τα κανονικοποιημένα βάρη (p_{ij}) των συνδέσεων ενός κόμβου βαθμού k δεν οφείλονται σε οργανωτικές αρχές του δικτύου, αλλά σε τοπική τυχαιότητα [24], [25]. Υπό αυτή την υπόθεση, η συνάρτηση πυκνότητας πιθανότητας $\rho(x)$ για το κανονικοποιημένο βάρος x ακολουθεί μια Beta Κατανομή (συγκεκριμένα Beta(1, k-1), η οποία εξαρτάται από τον βαθμό k του κόμβου Στην πράξη, αυτό αναπαρίσταται με τη διανομή $k-1$ σημείων με ομοιόμορφη πιθανότητα στο διάστημα $[0,1]$, δημιουργώντας k υποδιαστήματα. Τα μήκη αυτών των υποδιαστημάτων αντιπροσωπεύουν τις αναμενόμενες τιμές για τα k κανονικοποιημένα βάρη p_{ij} σύμφωνα με την μηδενική υπόθεση. Η συνάρτηση πυκνότητας πιθανότητας για μια από αυτές τις μεταβλητές να λάβει μια συγκεκριμένη τιμή x σε συνάρτηση με τον βαθμό του k που εξετάζεται δίνεται από:

$$\rho(x)\,dx=(k-1)(1-x)^{k-2}\,dx \text{ [24].}$$

Για κάθε ακμή ενός δεδομένου κόμβου, υπολογίζεται η πιθανότητα α_{ij} το κανονικοποιημένο της βάρος p_{ij} να είναι συμβατό με την μηδενική υπόθεση. Αυτή είναι η p-value. Η α_{ij} είναι η πιθανότητα, αν η μηδενική υπόθεση είναι αληθής, να λάβουμε μια τιμή για τη μεταβλητή υπό εξέταση μεγαλύτερη ή ίση με την παρατηρούμενη. Μαθηματικά αυτό εκφράζεται:

$$a_{ij}=\int_{p_{ij}}^{1}\rho(x)\,dx=\int_{p_{ij}}^{1}(k-1)(1-x)^{k-2}\,dx\ ,\ \text{η ισοδύναμα}\quad \alpha_{ij}=1-(k-1)\int_{0}^{p_{ij}}(1-x)^{k-2}\,dx$$

Χρησιμοποιώντας ένα επίπεδο σημαντικότητας α, οι ακμές για τις οποίες η τιμή p-value α_{ij} είναι μικρότερη από το επίπεδο σημαντικότητας ($a_{ij}<\alpha$) απορρίπτουν την μηδενική υπόθεση. Αυτές οι ακμές θεωρούνται ότι έχουν βάρη που δεν είναι συμβατά με μια τυχαία κατανομή και επομένως αντιπροσωπεύουν σημαντικές ετερογένειες που οφείλονται στις αρχές οργάνωσης του δικτύου. Έτσι

οι ακμές με $\alpha_{ij} < \alpha$ διατηρούνται [24]. Για κάθε ακμή στο αρχικό δίκτυο, ελέγχεται αν ικανοποιεί το κριτήριο διατήρησης ($\alpha_{ij} < \alpha$) για τουλάχιστον έναν από τους δύο κόμβους που συνδέει. Αν η ακμή ικανοποιεί το κριτήριο για έναν ή και τους δύο κόμβους, διατηρείται στον εξαγόμενο κορμό. Οι ακμές που δεν ικανοποιούν το κριτήριο για κανέναν από τους δύο κόμβους απορρίπτονται. Αυτό διασφαλίζει ότι ασθενείς κόμβοι δεν υποτιμούνται [24]. Επαναλαμβάνοντας με διαφορετικά επίπεδα σημαντικότητας μπορούν να φιλτραριστούν σταδιακά περισσότερες ακμές, εστιάζοντας σε πιο σημαντικές συνδέσεις. Μειώνοντας το επίπεδο σημαντικότητας (μικρότερες τιμές α), λαμβάνονται πιο περιορισμένα υποσύνολα του δικτύου [24].

4.6 Η Μέθοδος Marginal Likelihood Filter

Η Marginal Likelihood Filter η αλλιώς Φίλτρο Οριακής Πιθανότητας είναι μια στατιστική μέθοδος για την εξαγωγή κορμών δικτύων που αξιολογεί τη σημαντικότητα μιας ακμής λαμβάνοντας υπόψη το τοπικό πλαίσιο και των δύο κόμβων που συνδέονται στην εξεταζόμενη ακμή, σε αντίθεση με το Disparity Filter, που αξιολογεί την ακμή από την οπτική κάθε κόμβου ξεχωριστά [18]. Αυτή η προσέγγιση αποφεύγει να υποτιμήσει συστηματικά κόμβους χαμηλού βαθμού και τις δομές που αντιπροσωπεύουν [26].

Το μοντέλο αυτό διατηρεί το συνολικό βάρος του δικτύου και τον μέσο βαθμό κάθε κόμβου. Υποθέτει ότι οι ακμές επιλέγονται τυχαία, με τις πιθανότητες να είναι ανάλογες με τους βαθμούς των κόμβων στους οποίους καταλήγουν. Σύμφωνα με το μοντέλο αυτό, όσο μεγαλύτεροι είναι οι βαθμοί δύο κόμβων, τόσο πιο πιθανό είναι να συνδέονται τυχαία. Ως εκ τούτου, για να θεωρηθεί μια σύνδεση σημαντική, πρέπει να έχει μεγαλύτερο βάρος εάν οι δύο κόμβοι έχουν ήδη υψηλούς βαθμούς [26].

Το null model υποθέτει ότι οι μοναδιαίες ακμές του γράφου (μια σταθμισμένη ακμή βάρους w θεωρείται ως w παράλληλες ακμές μοναδιαίου βάρους) αντιστοιχίζονται σε ζεύγη κορυφών μία κάθε φορά και ανεξάρτητα η μία από την άλλη. Για κάθε μοναδιαία ακμή, οι δύο τελικοί κόμβοι επιλέγονται ανεξάρτητα τυχαία με πιθανότητες ανάλογες με τον βαθμό τους. Η πιθανότητα σύνδεσης μεταξύ δύο κορυφών i και j με βαθμούς k_i και k_j αντίστοιχα δίνεται από:

$$p = \frac{k_i * k_j}{2 * T^2}$$

όπου $T = \frac{1}{2}\sum_i k_i,$ είναι ο συνολικός αριθμός μοναδιαίων ακμών (το συνολικό βάρος του γράφου) [26].

Έχοντας ως δεδομένο την μηδενική υπόθεση, η πιθανότητα το βάρος σ_{ij} της ακμής που συνδέει τις κορυφές i και j να είναι m (δηλαδή, m από τις T συνολικές μοναδιαίες ακμές να επιλέξουν τις i και j ως τελικά σημεία) ακολουθεί μια διωνυμική κατανομή [26]. Η κατανομή αυτή μαθηματικά ορίζεται ως :

$$Pr[\sigma_{ij} = m] = m \ \ |k_i, k_j, T| = \left(\frac{T}{m}\right) p^m * (1-p)^{T-m}$$

Υπολογίζεται η τιμή p-value για το παρατηρούμενο βάρος κάθε ακμής. Έστω w_{ij} το παρατηρούμενο βάρος της ακμής που συνδέει τις κορυφές i και j. Η p-value $s_{ij}\ (w_{ij}\ )$ για αυτή την ακμή ορίζεται ως η πιθανότητα να παρατηρηθεί ένα βάρος μεγαλύτερο ή ίσο με το παρατηρούμενο βάρος w_{ij} υπό το null model ως :

$$s_{ij}(w_{ij}) = \sum_{m \geq wij} Pr\left[\sigma_{ij} = m \mid k_i, k_j, T\right] \text{ [26].}$$

Οποιαδήποτε ακμή με p-value $s_{ij}(w_{ij}) > \alpha$, όπου α το επίπεδο σημαντικότητας που έχει οριστεί (π.χ. α = 0,05 στην περίπτωσή μας) απορρίπτεται. Οι ακμές με τιμή p-value $s_{ij}(w_{ij}) \leq \alpha$ διατηρούνται.

Για μεγάλο T , η διωνυμική κατανομή μπορεί να προσεγγιστεί από την κατανομή Poisson (για $pT = O(1)$) ή την κανονική κατανομή (για $pT \gg 1$), για την αποτελεσματικότερη αριθμητική αξιολόγηση της p-value [26].

4.7 Η Μέθοδος Noise Corrected Filter

Η Noise Corrected Filter είναι μια στατιστική μέθοδος εξαγωγής κορμού που βασίζεται σε μια διωνυμική κατανομή, όπως η Marginal Likelihood Filter. Χρησιμοποιεί ένα Bayesian πλαίσιο για να εκτιμήσει την πιθανότητα να παρατηρήσουμε ένα συγκεκριμένο βάρος μεταξύ δύο κόμβων. Υπολογίζει την διακύμανση των ακμών και ιατηρεί μια ακμή μόνο αν το βάρος της είναι τουλάχιστον δ τυπικές αποκλίσεις πάνω από την αναμενόμενη τιμή της [27].

Η μηδενική υπόθεση (null model) υποθέτει ότι κάθε ατομική αλληλεπίδραση που ξεκινά από έναν κόμβο i βρίσκει ως προορισμό έναν κόμβο j με πιθανότητα ίση με το μερίδιο των συνολικών αλληλεπιδράσεων στο δίκτυο που λαμβάνει ο j. Η αναμενόμενη τιμή του βάρους της ακμής μεταξύ των κόμβων i και j δίνεται από τον τύπο:

$$E\left[N_{ij}\right] = \frac{N_i \cdot N_j}{N}$$

όπου N_i είναι το συνολικό εξερχόμενο βάρος του κόμβου i, N_j είναι το συνολικό εισερχόμενο βάρος του κόμβου j, και N είναι το συνολικό βάρος στο δίκτυο [27].

Ορίζεται ένα μέτρο *Lift* το οποίο μετρά πόσο απροσδόκητα υψηλό είναι το παρατηρούμενο βάρος μιας ακμής σε σχέση με το αναμενόμενο βάρος υπό το null model. Υπολογίζεται ως ο λόγος του παρατηρούμενου βάρους (N_{ij}) προς το αναμενόμενο βάρος ($E[N_{ij}]$):

$$L_{ij} = \frac{N_{ij}}{E\left[N_{ij}\right]}$$

Επειδή το *Lift* εμφανίζει ασυμμετρία γύρω από το 1, μετασχηματίζεται σε μια συμμετρική μέτρηση γύρω από το 0, χρησιμοποιώντας τον ακόλουθο μετασχημματισμό :

$$L_{ij}' = \frac{L_{ij} - 1}{L_{ij} + 1} = \frac{N_{ij} - E[N_{ij}]}{N_{ij} + E[N_{ij}]}$$

όπου $\epsilon = \frac{1}{E[N_{ij}]}$ [27].

Η διακύμανση του L_{ij}' υπολογίζεται χρησιμοποιώντας την μέθοδο Delta στην εξής συνάρτηση:

$$L'(N_{ij}) = \frac{N_{ij} - E[N_{ij}]}{N_{ij} + E[N_{ij}]}$$

Αυτό οδηγεί στην έκφραση : $V[Li\,j'] = V[Nij]\frac{d\,L'^2}{d\,N_{ij}} = V[Nij]\left(\frac{2\,E[N_{ij}]}{(N_{ij} + E[N_{ij}])^2}\right)^2$ όπου η διακύμανση του N_{ij} $(V[N_{ij}])$ εκτιμάται χρησιμοποιώντας μια Bayesian προσέγγιση [27].

Χρησιμοποιείται μια αρχική Beta κατανομή (conjugate prior) για την P_{ij} την πιθανότητα αλληλεπίδρασης $\frac{N_{ij}}{N}$. Η κατανομή της P_{ij} που προκύπτει (posterior distribution) είναι επίσης μια Beta κατανομή:

$$P_{ij} \sim BETA[n_{ij} + \alpha, n - n_{ij} + \beta]$$

όπου n_{ij} είναι ο αριθμός των ατομικών αλληλεπιδράσεων που αντιστοιχούν στο βάρος N_{ij}, και n είναι ο συνολικός αριθμός αλληλεπιδράσεων (N). Οι παράμετροι α και β της prior Beta κατανομής καθορίζονται με βάση ένα υπεργεωμετρικό μοντέλο.

Η αναμενόμενη τιμή της posterior κατανομής της P_{ij} χρησιμοποιείται για τον επαναϋπολογισμό της διακύμανσης του N_{ij} : $V[N_{ij}] = NE[P_{ij} \mid \delta\epsilon\delta o\mu\acute{\epsilon}\nu\alpha](1 - E[P_{ij} \mid \delta\epsilon\delta o\mu\acute{\epsilon}\nu\alpha])$. Αυτό αποφεύγει την εκτίμηση μηδενικής διακύμανσης όταν $N_{ij} = 0$ [27]. Μια ακμή διατηρείται εάν το παρατηρούμενο μετασχηματισμένο $Lift(L_{ij}')$ είναι υψηλότερο από ένα κατώφλι που βασίζεται στην τυπική απόκλιση του $L_{ij}' : L_{ij}' > \zeta V[L_{ij}']$ όπου ζ είναι μια παράμετρος που καθορίζει την ανοχή στον θόρυβο. Υψηλότερες τιμές του ζ οδηγούν σε πιο αυστηρό φιλτράρισμα και μπορούν να συσχετιστούν εμπειρικά με p-values ενός μονομερούς στατιστικού ελέγχου [27].

4.8 Υπεργεωμετρικός έλεγχος

Η βασική ιδέα αυτής της μεθόδου είναι να αξιολογήσει τη στατιστική σημαντικότητα των κοινών συνδέσεων μεταξύ δύο κόμβων σε ένα πολύπλοκο δίκτυο, όπου οι συνδέσεις έχουν διαφορετικούς τύπους, χρησιμοποιώντας ένα πρότυπο μηδενικής υπόθεσης που λαμβάνει υπόψη τη συνολική δραστηριότητα και τις προτιμήσεις των κόμβων [28]. Με βάση αυτή την μέθοδο για κάθε κόμβο $i \in U$ (π.χ. στο σύνολο γονιδίων) και κάθε κόμβο $j \in U$ υπολογίζονται τα ακόλουθα χαρακτηριστικά:

α) Το **επίπεδο δραστηριότητας**, δηλαδή ο συνολικός αριθμός γεγονότων (N) στα οποία ο κόμβος i και j συνδέονται $N_i = \sum_{t=1}^{N} 1 \;\; B_{it} \in a.$

β) Αριθμός κοινών γειτόνων (N_{ij}), ο συνολικός αριθμός γεγονότων στα οποία τόσο ο κόμβος i όσο και ο κόμβος j είναι συνδεδεμένοι.

γ) Ο **αριθμός κοινών γειτόνων με ίδιο τύπο σύνδεσης** (N_{ij}), δηλαδή υπολογίζεται ο αριθμός των γεγονότων στα οποία οι κόμβοι i και j συνδέονται [28].

$$N_i = \sum_{t=1}^{N} 1 \quad B_{it} = B_{jt}$$

δ) Οι **προτιμήσεις τύπων συνδέσεων** $(p_i(a_k))$ για κάθε κόμβο i, εκτιμάται η εμπειρική πιθανότητα με την οποία ο κόμβος χρησιμοποιεί κάθε τύπο σύνδεσης a_k [28].

ε) Η **συνολική πιθανότητα κοινού τύπου σύνδεσης** (p_{ij}) δηλαδή η πιθανότητα δύο κόμβων i και j να χρησιμοποιήσουν τυχαία τον ίδιο τύπο σύνδεσης σε ένα δεδομένο γεγονός [28]. Η παράμετρος p_{ij} αντιπροσωπεύει τη συνολική πιθανότητα οι δύο κόμβοι, ο i και ο j, να επιλέξουν τυχαία τον ίδιο τύπο σύνδεσης σε ένα γεγονός. Για να συμβεί αυτό, πρέπει να ισχύουν ταυτόχρονα δύο συνθήκες, ο κόμβος i να επιλέξει έναν συγκεκριμένο τύπο σύνδεσης (έστω τον k) και ο κόμβος j να επιλέξει ακριβώς τον ίδιο τύπο σύνδεσης (k). Επειδή, σύμφωνα με το μοντέλο μηδενικής υπόθεσης, οι επιλογές των κόμβων i και j είναι ανεξάρτητες, η πιθανότητα να επιλέξουν και οι δύο έναν συγκεκριμένο τύπο σύνδεσης a_k ισούται με το γινόμενο των επιμέρους πιθανοτήτων τους ως:

$$p_{ij} = \sum_{k=1}^{n} p_i(a_k)\, p_j(a_k)$$

Το null model υποθέτει ότι οι κόμβοι του ενός συνόλου επιλέγουν τυχαία γεγονότα για σύνδεση και τυχαία τους τύπους σύνδεσης. Το μοντέλο αυτό διατηρεί ακριβώς την ακολουθία βαθμού για τους κόμβους του συνόλου U (π.χ., γονίδια) και διατηρεί τον αριθμό των διαφορετικών τύπων συνδέσεων που χρησιμοποιούνται από αυτούς τους κόμβους ως προσδοκίες συνόλου [28]. Η στατιστική σημαντικότητα του παρατηρούμενου αριθμού κοινών γεγονότων με τους ίδιους τύπους συνδέσεων (N_{ij}) αξιολογείται χρησιμοποιώντας την υπεργεωμετρική-διωνυμική κατανομή. Το p-value δίνεται από τη σχέση:

$$P(X \geq N_{ij} \mid N_i, , N_j, p_i, p_j) = \sum_{X=0}^{min(N_i, N_j)} H(X \mid N, Nj) \cdot \sum_{Y=N_{ij}}^{X} B(Y \mid X, p_{ij})$$

Όπου:

- $H(X \mid N, N_j)$ είναι η πιθανότητα της υπεργεωμετρικής κατανομής, που περιγράφει την πιθανότητα να παρατηρηθούν ακριβώς X κοινοί γείτονες μεταξύ των κόμβων i και j.
- $B(Y \mid X, p_{ij})$ είναι η πιθανότητα της διωνυμικής κατανομής, που περιγράφει την πιθανότητα να παρατηρηθούν ακριβώς Y κοινές συνδέσεις του ίδιου τύπου, δεδομένου ότι υπάρχουν X κοινοί γείτονες και η πιθανότητα κοινού τύπου είναι p_{ij} [28].

Αυτό το p-value εκφράζει την πιθανότητα να παρατηρηθεί ο αριθμός N_{ij} (ή μεγαλύτερος) κοινών γεγονότων με τον ίδιο τύπο σύνδεσης, τυχαία, λαμβάνοντας υπόψη τον συνολικό αριθμό γεγονότων, τα επίπεδα δραστηριότητας των κόμβων και τις προτιμήσεις τους για τους τύπους συνδέσεων. Παρόμοια με τις καθιερωμένες μεθόδους, μετά τον υπολογισμό των p-values για όλες τις δυνητικές ακμές, εφαρμόζεται διόρθωση πολλαπλών ελέγχων (Benjamini-Hochberg FDR correction). Ακμές με διορθωμένο p-value κάτω από ένα προκαθορισμένο όριο (π.χ., 0.05) θεωρούνται στατιστικά σημαντικές και διατηρούνται στο τελικό μονομερές δίκτυο [28].

4.9 Locally Adaptive Network Sparsification Filter

Η Locally Adaptive Network Sparsification (LANS) Filter είναι μια μέθοδος εξαγωγής backbone που δεν βασίζεται σε υποθέσεις σχετικά με την κατανομή των βαρών των ακμών. Αντίθετα, χρησιμοποιεί την εμπειρική σωρευτική συνάρτηση κατανομής (empirical cumulative density function, ECDF) για την αξιολόγηση της στατιστικής σημαντικότητας των ακμών. Από την οπτική των κόμβων που συνδέονται με μια ακμή, η μέθοδος υπολογίζει την πιθανότητα επιλογής μιας ακμής με βάρος ίσο με το παρατηρούμενο. Με αυτόν τον τρόπο, η LANS Filter επιτρέπει την τοπικά προσαρμοστική αραίωση του δικτύου, διατηρώντας σημαντικές ακμές χωρίς να επιβάλλει προκαθορισμένα μοντέλα κατανομής βαρών. Η μέθοδος LANS θεωρείται μη παραμετρική, διότι δεν κάνει υποθέσεις για τη μορφή της κατανομής των βαρών ακμών, αλλά αντλεί από την εμπειρική κατανομή των βαρών για να εκτιμήσει τη σημαντικότητα των ακμών [29].

Η μέθοδος λαμβάνει ως είσοδο ένα δίκτυο με βάρη ακμών w_{ij}, όπου w_{ij} αντιπροσωπεύει το βάρος της ακμής από τον κόμβο i προς τον κόμβο j. Για κάθε κόμβο i και κάθε γείτονά του j (όπου w_{ij}>0), υπολογίζεται το κλασματικό βάρος της ακμής p_{ij} ως εξής:

$$p_{ij} = \frac{w_{ij}}{\sum_{k=1}^{N_i} w_{ik}},$$

όπου N_i είναι ο αριθμός των γειτόνων του κόμβου i [29].

Για κάθε κόμβο i, υπολογίζεται η εμπειρική συνάρτηση κατανομής των σχετικών βαρών των ακμών που ξεκινούν από αυτόν τον κόμβο [29]. Για ένα συγκεκριμένο σχετικό βάρος ακμής p_{ij} από τον κόμβο i προς τον κόμβο j, η εμπειρική CDF, συμβολίζεται ως F^(p_{ij}), υπολογίζεται ως το κλάσμα των μη μηδενικών ακμών που ξεκινούν από τον κόμβο i και έχουν σχετικό βάρος μικρότερο ή ίσο με p_{ij}:

$$\hat{F}(p_{ij}) = \frac{1}{N_i}\sum_{k=1}^{N_i} 1\{pik \le pij\}, \quad \text{με} \quad p_{ik} \le p_{ij} = \begin{cases} 1, & p_{ik} \le p_{ij} \\ 0, & \textit{Διαφορετικά} \end{cases}$$

Για κάθε ακμή από τον κόμβο i προς τον κόμβο j, συγκρίνεται η τιμή $1-\hat{F}(p_{ij})$ με ένα προκαθορισμένο επίπεδο στατιστικής σημαντικότητας α. Εάν $1-\hat{F}(p_{ij})$ < α, τότε η ακμή από τον κόμβο i προς τον κόμβο j θεωρείται τοπικά στατιστικά σημαντική και συμπεριλαμβάνεται στον κορμό του δικτύου. Αυτό σημαίνει ότι το παρατηρούμενο σχετικό βάρος p_{ij} είναι στατιστικά απίθανο με βάση την εμπειρική κατανομή των βαρών που ξεκινούν από τον κόμβο i [29].

Η μέθοδος LANS απαιτεί τον υπολογισμό της εμπειρικής κατανομής σε κάθε κόμβο, γεγονός που οδηγεί σε υπολογιστική πολυπλοκότητα $O(n2logN)$, όπου n είναι ο αριθμός των κόμβων και N είναι ο μέγιστος βαθμός του δικτύου. Αν το δίκτυο είναι αραιό, N μπορεί να είναι σημαντικά μικρότερο από n. Η μέθοδος είναι πλήρως παραλληλοποιήσιμη, κάτι που μπορεί να μειώσει την υπολογιστική επιβάρυνση. Η LANS, που προσαρμόζεται τοπικά στη κατανομή των βαρών κάθε κόμβου, αντιμετωπίζει την υψηλή ετερογένεια που παρατηρείται σε πραγματικά δίκτυα, σε αντίθεση με το disparity filter που χρησιμοποιεί μια παραμετρική κατανομή για να συγκρίνει τα βάρη, η οποία μπορεί να μην αντικατοπτρίζει σωστά την ετερογένεια των τοπικών κατανομών και ενδέχεται να αγνοεί την τοπική κατανομή των βαρών, κάτι που μπορεί να οδηγήσει σε σημαντική παραμόρφωση της δομής του δικτύου [29].

4.10 Multiple Linkage Analysis Filter

Η συγκεκριμένη μέθοδος υποθέτει ότι τα βάρη κατανέμονται ομοιόμορφα μεταξύ των γειτονικών κόμβων ενός κόμβου, με τιμές από 1 έως n. Για την επιλογή των ακμών που θα διατηρηθούν, υπολογίζει ένα μέτρο προσαρμογής (goodness of fit) συγκρίνοντας την παρατηρούμενη κατανομή βαρών με υποθετικές κατανομές, χρησιμοποιώντας συντελεστή συσχέτισης. Ο βέλτιστος αριθμός ακμών που θα διατηρηθεί για κάθε κόμβο καθορίζεται από τον αριθμό που μεγιστοποιεί τον συντελεστή συσχέτισης, διατηρώντας έτσι τη σημαντικότερη τοπολογική πληροφορία. Η μέθοδος ορίζεται από τα ακόλουθα βήματα.

Για κάθε κόμβο i, καταγράφονται όλες οι ροές προς/από άλλους κόμβους:

Έστω ότι υπάρχουν k ροές: $w_1, w_2, \ldots, w_k$. Οι ροές ταξινομούνται κατά φθίνουσα σειρά $w_1 \geq w_2 \geq \cdots \geq w_k$

Στην συνέχεια δημιουργούνται κύκλοι αναμενόμενων ροών. Με τον όρο κύκλους στην περιγραφή της μεθόδου Multiple Linkage Analysis, δεν αναφερόμαστε σε κυκλικές διαδρομές ή κλειστά μονοπάτια μέσα στο δίκτυο. Αντίθετα, χρησιμοποιούμε τον όρο μεταφορικά για να περιγράψουμε μια διαφορετική υπόθεση σχετικά με τον τρόπο κατανομής των ροών από ή προς μια χωρική μονάδα. Κάθε "κύκλος" στην MLA αντιπροσωπεύει μια υποθετική κατανομή των συνολικών ροών σε έναν συγκεκριμένο αριθμό των ισχυρότερων συνδέσεων, υποθέτοντας ότι αυτές οι συνδέσεις έχουν ίση ισχύ μεταξύ τους στην ιδεατή αυτή κατανομή. Οι ροές αναφέρονται σε κάθε είδους μετακίνηση, αλληλεπίδραση ή σύνδεση μεταξύ των μονάδων που εξετάζονται και αντιπροσωπεύουν την ένταση τ ης σύνδεσης μεταξύ των κόμβων. Δημιουργείται μια σειρά "ιδανικών" κατανομών ροών (κύκλοι αναμενόμενων ροών, που συμβολίζονται ως w_i) [30]. Κάθε κύκλος υποθέτει ότι η συνολική ροή της χωρικής μονάδας κατανέμεται εξίσου σε έναν αυξανόμενο αριθμό προορισμών:

1ος Κύκλος: Υποθέτει ότι όλη η ροή συγκεντρώνεται στη μοναδική μεγαλύτερη σύνδεση [30].

$$\widetilde{w_{i1}} = \sum_{i \in I} w_i \ , \ \widetilde{w_2} = \widetilde{w_3} = \ldots = \widetilde{w_k} = 0$$

2ος Κύκλος: Υποθέτει ότι η συνολική ροή κατανέμεται εξίσου μεταξύ των δύο μεγαλύτερων συνδέσεων [30].

$$\widetilde{w_{i1}}=\widetilde{w_{i2}}=\frac{1}{2}\sum_{i\in I} w_i \text{ , } \widetilde{w_3}= \widetilde{w_4}=\ldots=\widetilde{w_k}=0$$

j-οστός κύκλος (j < k): Υποθέτει ότι η συνολική ροή κατανέμεται εξίσου μεταξύ των j μεγαλύτερων συνδέσεων.

$$\widetilde{w_1}= \widetilde{w_2}=\ldots=\widetilde{w_{ij}}=\frac{1}{j}\sum_{i\in I} w_i$$

k-οστός κύκλος: Υποθέτει ότι η συνολική ροή κατανέμεται εξίσου σε όλες τις k συνδέσεις.

$$\widetilde{w_1}= \widetilde{w_2}=\ldots=\widetilde{w_{ij}}=\frac{1}{k}\sum_{i\in I} w_i$$

Στον k-οστό κύκλο, η συνολική ροή του κόμβου κατανέμεται ομοιόμορφα μεταξύ των k πρώτων συνδέσεων, ενώ οι υπόλοιπες συνδέσεις θεωρούνται ότι δεν έχουν ροή. Για κάθε κύκλο αναμενόμενων ροών, μετράται η ποιότητα της προσαρμογής μεταξύ της παρατηρούμενης κατανομής ροών και της ιδανικής κατανομής χρησιμοποιώντας τον συντελεστή προσδιορισμού (COD). Ο συντελεστής COD ποσοτικοποιεί πόσο καλά κάθε ιδανική κατανομή εξηγεί τη διακύμανση στην πραγματική κατανομή ροών [30].

$$R_j^2=1-\frac{\sum\left(w_i+\widehat{w}_j\right)^2}{\sum\left(w_i+\overline{w}_j\right)^2}$$

Ο αριθμός των σημαντικών ροών για μια χωρική μονάδα καθορίζεται εντοπίζοντας τον κύκλο αναμενόμενων ροών που αποδίδει την υψηλότερη τιμή COD. Ο αριθμός των συνδέσεων που υποτίθεται ότι έχουν ίση ροή σε αυτόν τον κύκλο με το υψηλότερο COD αντιπροσωπεύει τον αριθμό των ουσιαστικών συνδέσεων στη δομή συνδεσιμότητας της χωρικής μονάδας, ξεπερνώντας τους περιορισμούς της εξέτασης μόνο της κυρίαρχης ροής. Τέλος γίνεται επιλογή του κύκλου με τον υψηλότερο COD [30].

Συνεπώς, η MLA συγκρίνει συστηματικά την πραγματική κατανομή των συνδέσεων με μια σειρά απλοποιημένων σεναρίων όπου οι συνδέσεις θεωρούνται όλο και περισσότερο ισοδύναμες [30]. Ο κύκλος που ταιριάζει καλύτερα στην πραγματική κατανομή (όπως υποδεικνύεται από τον υψηλότερο COD) αποκαλύπτει πόσες από τις συνδέσεις είναι πραγματικά σημαντικές για τον καθορισμό της δομής του δικτύου με επίκεντρο αυτή τη χωρική μονάδα, προχωρώντας πέρα από τους περιορισμούς της εξέτασης μόνο της κυρίαρχης ροής.

Η μέθοδος Multiple Linkage Analysis (MLA) παρέχει ως κύρια έξοδο για κάθε κόμβο (π.χ., πόλη, γονίδιο) στο δίκτυο τον αριθμό των σημαντικών συνδέσεων (Number of Significant Flows). Αυτός είναι ο βασικός στόχος της μεθόδου. Υποδεικνύει πόσες από τις συνδέσεις του κόμβου θεωρούνται στατιστικά σημαντικές για την περιγραφή της δομής της συνδεσιμότητάς του. Προκύπτει από τον αριθμό (j) του κύκλου αναφοράς (ιδανικής κατανομής ροών) που έχει τον υψηλότερο συντελεστή προσδιορισμού (COD) σε σύγκριση με την πραγματική κατανομή των ροών του κόμβου [30].

4.11 Modularity Filter

Αυτή η μέθοδος αποτελεί μια τεχνική εξαγωγής ραχοκοκαλιάς δικτύου (backbone extraction) που βασίζεται στη διατήρηση της δομής των κοινοτήτων ενός δικτύου [31]. Συγκεκριμένα μειώνει το μέγεθος ενός δικτύου αφαιρώντας κόμβους που έχουν τη μικρότερη συνεισφορά στη συνολική του αρθρωτότητα (modularity). Η αρθρωτότητα είναι ένα μέτρο που ποσοτικοποιεί την ποιότητα της κοινοτικής δομής ενός δικτύου, συγκρίνοντας την πυκνότητα των συνδέσεων εντός των κοινοτήτων με μια τυχαία διασύνδεση. Σε αντίθεση με τις απλές μεθόδους που βασίζονται σε στατιστικά μέτρα όπως η βαρύτητα των ακμών (όπως disparity filter κ.λ.π.) ή τον βαθμό των κόμβων, αυτή η μέθοδος δίνει προτεραιότητα στη διατήρηση της κοινοτικής δομής του δικτύου. Θεωρεί ότι οι κόμβοι που είναι κρίσιμοι για τη συνοχή των κοινοτήτων τους είναι οι πιο σημαντικοί. Αντί να αφαιρεί ακμές (edges), η μέθοδος αυτή αφαιρεί ολόκληρους κόμβους [31]. Το κριτήριο αφαίρεσης δεν είναι ο αριθμός των συνδέσεων ενός κόμβου, αλλά η συνεισφορά του στην αρθρωτότητα του δικτύου. Αυτό είναι ένα πιο εξελιγμένο μέτρο που λαμβάνει υπόψη τη συνολική δομή του δικτύου και όχι μόνο τις τοπικές συνδέσεις. Επίσης υπολογίζει την επίδραση κάθε κόμβου στο σύνολο του δικτύου (modularity vitality), και όχι μόνο σε τοπικό επίπεδο. Αυτό της επιτρέπει να εντοπίσει και να διατηρήσει σημαντικούς κόμβους που λειτουργούν ως "γέφυρες" μεταξύ των κοινοτήτων, κάτι που άλλες μέθοδοι μπορεί να αγνοήσουν.

Αρχικά, υπολογίζεται η αρθρωτότητα $Q(G)$ για ολόκληρο το αρχικό δίκτυο , με βάση τον τύπο:

$$Q = \frac{1}{2|E|}\sum_{i,j}\left[A_{ij}-\frac{k_i k_j}{2|E|}\right]\delta(c_i,c_j)$$

όπου A_{ij} είναι το στοιχείο του πίνακα γειτνίασης. k_i είναι ο βαθμός του κόμβου i (δηλαδή το άθροισμα των βαρών των ακμών του, αν το δίκτυο είναι σταθμισμένο), |E| ο συνολικός αριθμός των ακμών (το άθροισμα των βαρών τους, αν το δίκτυο είναι σταθμισμένο), c_i και c_j είναι οι κοινότητες στις οποίες ανήκουν οι κόμβοι i και j, $\delta(c_i,c_j)$είναι η συνάρτηση Kronecker-delta, η οποία ισούται με 1 αν οι κόμβοι i και j ανήκουν στην ίδια κοινότητα, και με 0 διαφορετικά [31].

Ουσιαστικά, ο τύπος αυτός συγκρίνει τον πραγματικό αριθμό των ακμών εντός των κοινοτήτων (A_{ij}) με τον αναμενόμενο αριθμό των ακμών $\frac{k_i * k_j}{2 \mid E \mid}$ που θα υπήρχε σε ένα τυχαίο δίκτυο με την ίδια κατανομή βαθμού. Μια υψηλή τιμή Q υποδηλώνει μια ισχυρή κοινοτική δομή, ενώ μια τιμή κοντά στο μηδέν υποδηλώνει ότι το δίκτυο έχει τυχαία διασύνδεση [31].

Για κάθε κόμβο v στο δίκτυο, υπολογίζεται η τιμή vitality, η οποία ορίζεται ως η διαφορά στην αρθρωτότητα του δικτύου με και χωρίς τον συγκεκριμένο κόμβο: $\alpha(v)=Q(G)-Q(G\setminus v)$. Η αρθρωτότητα του δικτύου χωρίς τον κόμβο v υπολογίζεται παραπάνω [31]. Οι κόμβοι κατατάσσονται με βάση την απόλυτη τιμή της βαθμολογίας vitality, $\mid \alpha(v) \mid$, σε αύξουσα σειρά. Αυτό γίνεται για να δοθεί ίση σημασία σε κόμβους που αυξάνουν την αρθρωτότητα (π.χ., hubs) και σε αυτούς που τη μειώνουν (π.χ., bridges). Έπειτα αφαιρούνται διαδοχικά οι κόμβοι με τη χαμηλότερη συνεισφορά στην αρθρωτότητα, έως ότου το δίκτυο φτάσει στο επιθυμητό μέγεθος. Τελικά οι εναπομείναντες κόμβοι σχηματίζουν τη ραχοκοκαλιά (backbone) του δικτύου [31].

4.12 Maximall Spanning Tree

Η μέθοδος του Maximal Spanning Tree (MST) είναι μια τεχνική βελτιστοποίησης που βρίσκει ένα υποσύνολο ακμών σε έναν γράφο. Ο κύριος σκοπός της είναι να συνδέσει όλους τους κόμβους του γράφου με το μεγαλύτερο δυνατό συνολικό βάρος, χωρίς να δημιουργηθούν κύκλοι. Η μέθοδος του Maximal Spanning Tree είναι χρήσιμη σε περιπτώσεις όπου ο στόχος είναι η εύρεση του πιο «ισχυρού» δικτύου σύνδεσης. Με την εύρεση του MST, μπορούμε να αναγνωρίσουμε τις πιο ισχυρές αλληλεπιδράσεις που διατηρούν τη συνολική συνδεσιμότητα του δικτύου. Αυτό μας επιτρέπει να επικεντρωθούμε στα πιο σημαντικά ζεύγη του δικτύου.

Η μέθοδος βασίζεται στον αλγόριθμο Kruskal, ο οποίος βρίσκει το μέγιστο spanning tree T ενός σταθμισμένου, μη κατευθυνόμενου γράφου $G=(V,E)$, όπου V είναι το σύνολο των κόμβων (γονίδια) και E το σύνολο των ακμών (αλληλεπιδράσεις). Κάθε ακμή $e \in E$ έχει ένα βάρος $w(e)>0$ [32].

Ο αλγόριθμος στοχεύει να μεγιστοποιήσει το συνολικό βάρος του δέντρου, το οποίο ορίζεται ως το άθροισμα των βαρών όλων των ακμών του: $max_{T \subseteq E} \sum_{e \in T} w(e)$, με την προϋπόθεση ότι το T είναι ένα spanning tree του G. Οι ακμές ταξινομούνται σε μια λίστα (sorted list) κατά φθίνουσα σειρά βάρους. Αν $e_1, e_2, \ldots, e_m$ είναι οι ακμές του γράφου, τότε ισχύει:

$$w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$$

Έπειτα δημιουργείται ένα κενό σύνολο ακμών $T=\varnothing$. Για κάθε ακμή e_i στη λίστα sorted (με i από 1 έως m) ελέγχεται αν η προσθήκη της ακμής e_i στο σύνολο T θα δημιουργήσει κύκλο. Αυτό ισοδυναμεί με τον έλεγχο αν οι δύο κόμβοι που συνδέει η ακμή e_i ανήκουν ήδη στην ίδια συνδεδεμένη συνιστώσα. Αν η ακμή e_i δεν δημιουργεί κύκλο, τότε προστίθεται στο σύνολο T:

$$T := T \cup e_i$$

Η επανάληψη τερματίζεται όταν το σύνολο T περιέχει ακριβώς $|V|-1$ ακμές. Σε αυτό το σημείο, το T αποτελεί το Maximal Spanning Tree και ο εξαγόμενος κορμός του αρχικού μας δικτύου [32].

4.13 Doubly Stochastic Filter

Η συγκεκριμένη μέθοδος αποτελεί έναν αλγόριθμο δύο σταδίων που χρησιμοποιείται για την εξαγωγή backbone σύνθετων, σταθμισμένων δικτύων. Ο πρωταρχικός της στόχος είναι να απλοποιήσει ένα πολύπλοκο δίκτυο, διατηρώντας μόνο τις πιο σημαντικές και ουσιαστικές συνδέσεις του. Η μέθοδος ξεχωρίζει από άλλες προσεγγίσεις, επειδή βασίζεται στην πλήρη ιεραρχική ομαδοποίηση όλων των κόμβων μέχρι να επιτευχθεί μια ενιαία, ισχυρά συνδεδεμένη συνιστώσα, αντί να φιλτράρει τις συνδέσεις με βάση ένα προκαθορισμένο επίπεδο σημαντικότητας. Αυτό το χαρακτηριστικό διασφαλίζει ότι η τελική ραχοκοκαλιά περιλαμβάνει το σύνολο των κόμβων, αποκαλύπτοντας τις κεντρικές δομές και ροές του δικτύου χωρίς να παραλείπει κανένα τμήμα του. Η χρησιμότητα της μεθόδου έγκειται στην ικανότητά της να μειώνει την πολυπλοκότητα της ανάλυσης, να αναγνωρίζει κρυμμένες περιφερειακές δομές και κεντρικούς κόμβους, και να παρέχει μια σαφή και

οπτικά ελκυστική αναπαράσταση των σημαντικότερων σχέσεων σε δίκτυα ροών, όπως οι μεταναστευτικές κινήσεις ή οι εμπορικές συναλλαγές [33].

Αρχικά, η μέθοδος παίρνει ως είσοδο μια μήτρα ροών, όπου κάθε στοιχείο της αντιπροσωπεύει τη δύναμη της σύνδεσης μεταξύ δύο κόμβων. Σκοπός είναι να μετατρέψει αυτή τη μήτρα σε μια μορφή, στην οποία όλα τα αθροίσματα των γραμμών και των στηλών είναι ίσα με 1. Αυτό επιτυγχάνεται επαναλαμβάνοντας εναλλάξ την κλιμάκωση των γραμμών και των στηλών. Αυτή η διαδικασία εξουδετερώνει τις διαφορές στις συνολικές ροές που εισέρχονται ή εξέρχονται από κάθε κόμβο, επιτρέποντας στην ανάλυση να επικεντρωθεί στη σημαντικότητα κάθε σύνδεσης [33].

Σε κάθε επανάληψη, γίνονται δύο βήματα:

1. **Κανονικοποίηση Γραμμής**: Για κάθε γραμμή της μήτρας, διαιρούμε κάθε στοιχείο με το άθροισμα όλων των στοιχείων αυτής της γραμμής. Έτσι, το άθροισμα κάθε γραμμής γίνεται 1.
2. **Κανονικοποίηση Στήλης**: Στη συνέχεια, κάνουμε το ίδιο για τις στήλες: διαιρούμε κάθε στοιχείο με το άθροισμα της στήλης του, κάνοντας το άθροισμα κάθε στήλης 1 [33].

Αυτά τα δύο βήματα επαναλαμβάνονται εναλλάξ μέχρι η μήτρα να συγκλίνει σε μια διπλά στοχαστική μορφή (Doubly Stochastic), δηλαδή μέχρι τα αθροίσματα τόσο των γραμμών όσο και των στηλών να είναι ίσα με 1. Το αποτέλεσμα είναι μια κανονικοποιημένη μήτρα, S, όπου τα στοιχεία της, s_{ij}, αντιπροσωπεύουν τη σχετική δύναμη της σύνδεσης, ανεξάρτητα από την συνολική ροή του κόμβου [33].

Στη συνέχεια, το δίκτυο χτίζεται σταδιακά από την αρχή. Πρώτα υπάρχει ένα γράφημα που περιέχει όλους τους κόμβους, αλλά χωρίς συνδέσεις. Έπειτα, προστίθενται τις συνδέσεις μία-μία, ξεκινώντας από αυτή που είχε το υψηλότερο κανονικοποιημένο βάρος στο προηγούμενο βήμα. Η διαδικασία αυτή συνεχίζεται μέχρι το γράφημα να γίνει ισχυρά συνδεδεμένο, δηλαδή μέχρι να υπάρχει μια διαδρομή από κάθε κόμβο προς κάθε άλλο κόμβο του δικτύου. Ένα γράφημα είναι ισχυρά συνδεδεμένο όταν υπάρχει μια κατευθυνόμενη διαδρομή από κάθε κόμβο σε κάθε άλλο κόμβο. Η ραχοκοκαλιά ορίζεται ως το σύνολο των ακμών που απαιτούνται για να επιτευχθεί αυτή η συνδεσιμότητα. Ο αλγόριθμος Tarjan προτείνεται ως ένας αποτελεσματικός τρόπο για να ελέγχεται η ισχυρή συνδεσιμότητα του γραφήματος σε κάθε βήμα. Το δίκτυο που προκύπτει ακριβώς τη στιγμή που συμβαίνει αυτό είναι η ραχοκοκαλιά, καθώς περιέχει το ελάχιστο σύνολο των συνδέσεων που απαιτούνται για να συνδέσουν όλους τους κόμβους [33].

4.14 Edge Betweenness

Η συγκεκριμένη μέθοδος είναι ένας αλγόριθμος για τον εντοπισμό της δομής κοινοτήτων σε δίκτυα, όπως τα κοινωνικά ή βιολογικά δίκτυα. Οι κοινότητες ορίζονται ως υποσύνολα κόμβων με πυκνές εσωτερικές συνδέσεις, ενώ οι συνδέσεις μεταξύ των κοινοτήτων είναι πιο αραιές. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη για την κατανόηση της οργάνωσης και λειτουργίας πολύπλοκων συστημάτων. Σε αντίθεση με τις παραδοσιακές μεθόδους ιεραρχικής ομαδοποίησης που εστιάζουν στους πυρήνες των κοινοτήτων, εστιάζει στα όρια τους [34]. Βασίζεται στην έννοια της edge betweenness για να εντοπίσει τις ακμές που λειτουργούν ως «γέφυρες» μεταξύ των κοινοτήτων [34].

Η διαμεσότητα ενός κόμβου i ορίζεται ως ο αριθμός των συντομότερων διαδρομών μεταξύ ζευγών άλλων κόμβων που διέρχονται από τον i. Αντίστοιχα, η edge betweenness ορίζεται ως ο αριθμός των συντομότερων διαδρομών μεταξύ όλων των ζευγών κόμβων του δικτύου που περνούν μέσα από την ακμή e. Αν υπάρχουν πολλαπλές συντομότερες διαδρομές μεταξύ ενός ζεύγους κόμβων, κάθε διαδρομή λαμβάνει ίση βαρύτητα, ώστε το συνολικό άθροισμα βαρύτητας όλων των διαδρομών να είναι μονάδα. Το κύριο πλεονέκτημα της μεθόδου είναι η ικανότητά της να εντοπίζει με ακρίβεια τα όρια των κοινοτήτων, ξεπερνώντας τις αδυναμίες των παραδοσιακών μεθόδων που δυσκολεύονται να χειριστούν περιφερειακούς κόμβους. Ωστόσο, το κυριότερο μειονέκτημα είναι η υψηλή υπολογιστική πολυπλοκότητα. Ο αλγόριθμος εκτελείται σε χρόνο $O(m^2 n)$ στην χειρότερη περίπτωση (όπου m είναι ο αριθμός των ακμών και n είναι ο αριθμός των κόμβων), καθιστώντας τον μη πρακτικό για πολύ μεγάλα δίκτυα [34].

Η τιμή betweenness μιας ακμής μπορεί να υπολογιστεί με τον ακόλουθο τύπο:

$$B(e) = \sum_{s,t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

όπου :

- e είναι η εξεταζόμενη ακμή
- V είναι το σύνολο των κόμβων του δικτύου
- σ_{st}είναι ο συνολικός αριθμός συντομότερων διαδρομών μεταξύ του κόμβου s και του κόμβου t
- $\sigma_{st}(e)$ είναι ο αριθμός των συντομότερων διαδρομών μεταξύ του s και του t που διέρχονται από την ακμή e [34].

Ακμές που συνδέουν διαφορετικές κοινότητες τείνουν να έχουν υψηλή διαμεσότητα, καθώς πολλές συντομότερες διαδρομές μεταξύ διαφορετικών ομάδων περνούν αναγκαστικά από αυτές. Ο αλγόριθμος Girvan-Newman για τον εντοπισμό κοινοτήτων είναι μια επαναληπτική διαδικασία που αφαιρεί διαδοχικά τις ακμές του δικτύου [34]. Τα βήματα είναι τα εξής:

1. Υπολογίζεται η διαμεσότητα για όλες τις ακμές του δικτύου.
2. Εντοπίζεται η ακμή με την υψηλότερη διαμεσότητα και αφαιρείται από το δίκτυο.
3. Η διαμεσότητα επαναϋπολογίζεται για όλες τις ακμές που επηρεάστηκαν από την αφαίρεση.
4. Τα βήματα 2 και 3 επαναλαμβάνονται μέχρι να μην υπάρχουν άλλες ακμές στο δίκτυο.

Καθώς οι ακμές αφαιρούνται, το δίκτυο διασπάται σε ξεχωριστές συνιστώσες, οι οποίες θεωρούνται ως κοινότητες. Ο αλγόριθμος παράγει μια ιεραρχική δομή που δείχνει πώς οι κόμβοι ομαδοποιούνται σε κοινότητες καθώς αφαιρούνται οι ακμές. Η διαμεσότητα πρέπει να επαναϋπολογίζεται σε κάθε βήμα, καθώς η αφαίρεση μιας ακμής μπορεί να αλλάξει τις συντομότερες διαδρομές και, κατά συνέπεια, τη διαμεσότητα των υπόλοιπων ακμών [34].

4.15 Η Μέθοδος Polya Filter

Η μέθοδος του Pólya είναι μια στατιστική τεχνική φιλτραρίσματος που χρησιμοποιείται για να αναγνωρίσει τους πιο σημαντικούς (backbone) συνδέσμους σε ένα δίκτυο, αγνοώντας τις τυχαίες ή αδύναμες συνδέσεις. Στόχος της είναι να ξεχωρίσει τους συνδέσμους που είναι πραγματικά ισχυροί από αυτούς που εμφανίζονται απλά λόγω της ετερογένειας του δικτύου. Η μέθοδος αυτή είναι εμπνευσμένη από το μοντέλο της κάλπης του Pólya με μπάλες διαφορετικών χρωμάτων. Κάθε φορά που εξέρχεται μια μπάλα από την κάλπη, τοποθετείται πίσω μαζί με μια επιπλέον μπάλα του ίδιου χρώματος. Αυτός ο μηχανισμός αυτο-ενίσχυσης σημαίνει ότι όσο πιο συχνά εμφανίζεται ένα χρώμα, τόσο πιο πιθανό είναι να εμφανιστεί ξανά στο μέλλον.

Σε αντίθεση με άλλες μεθόδους, το Pólya Filter δεν υποθέτει ότι όλοι οι κόμβοι συμπεριφέρονται με τον ίδιο τρόπο. Αντιθέτως λαμβάνει υπόψη ότι οι κόμβοι που έχουν πολλούς ισχυρούς συνδέσμους είναι πιο πιθανό να σχηματίσουν νέους, ισχυρούς συνδέσμους, προσαρμόζοντας έτσι την ανάλυση στην πραγματική δομή του δικτύου. Για κάθε σύνδεσμο στο δίκτυο, υπολογίζει την πιθανότητα να έχει το συγκεκριμένο βάρος (πόσες φορές επαναλαμβάνεται) με βάση το μοντέλο της κάλπης. Αυτή η πιθανότητα μετατρέπεται σε ένα p-value, το οποίο μετρά πόσο "ασυνήθιστο" είναι το βάρος του συνδέσμου, λαμβάνοντας υπόψη τη συνολική ισχύ του κόμβου στον οποίο ανήκει. Χρησιμοποιώντας μια στατιστική διόρθωση (π.χ., Benjamini-Hochberg), η μέθοδος βρίσκει όλους τους συνδέσμους με στατιστικά σημαντικά χαμηλό p-value. Αυτοί οι στατιστικά σημαντικοί σύνδεσμοι αποτελούν τον κορμό (backbone) του δικτύου [35].

Η μέθοδος βασίζεται στην αρχική αναπαράσταση του δικτύου ως έναν πίνακα βαρών (A_{ij}), όπου κάθε στοιχείο αντιπροσωπεύει το βάρος του συνδέσμου μεταξύ των κόμβων i και j. Στη συνέχεια, υπολογίζονται οι εξής μετρικές για κάθε κόμβο:

- **Βαθμός** (degree k_i **)**: Ο αριθμός των μοναδικών συνδέσμων που συνδέονται με τον κόμβο i.
- **Ισχύς** (strength s_i): Το άθροισμα των βαρών όλων των συνδέσμων του κόμβου i.

Για έναν σύνδεσμο με βάρος w που ανήκει σε έναν κόμβο με βαθμό k και ισχύ s, το μοντέλο της κάλπης του Pólya περιγράφεται ως εξής με βάση την προσέγγιση της μεθόδου:

- Μία κόκκινη μπάλα αντιπροσωπεύει τον συγκεκριμένο σύνδεσμο.
- $k-1$ μαύρες μπάλες αντιπροσωπεύουν τους υπόλοιπους $k-1$ μοναδικούς συνδέσμους του κόμβου.
- $s-k$ κενές μπάλες αντιπροσωπεύουν τις υπόλοιπες επαναλήψεις των συνδέσμων.

Σε κάθε βήμα, επιλέγεται τυχαία μία μπάλα και επιστρέφεται στην κάλπη προσθέτοντας μία επιπλέον μπάλα του ίδιου χρώματος (ή τύπου). Αυτός είναι ο μηχανισμός αυτο-ενίσχυσης που περιγράφεται από την παράμετρο a [35]. Η πιθανότητα $P(w \mid k, s, a)$ να εμφανιστεί ο σύνδεσμος με βάρος w σε μια κάλπη με συνολική ισχύ s και βαθμό k, με την παράμετρο a, δίνεται από τον ακόλουθο τύπο:

$$P(w \mid k, s, a) = \binom{w}{s} \frac{B\left(\frac{w}{a}+1, \frac{s-w}{a}+\frac{k-1}{a}\right)}{B\left(\frac{1}{a}, \frac{k-1}{a}\right)}$$

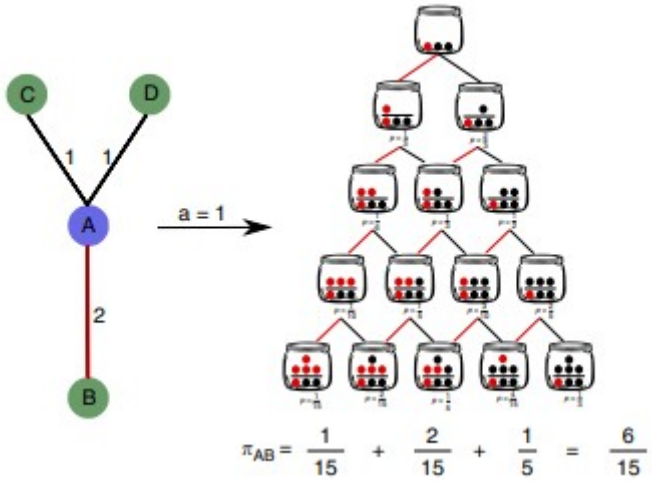
όπου:

- $\binom{w}{s}$ είναι ο δυωνυμικός συντελεστής (binomial coefficient).
- $B(x, y)$ είναι η συνάρτηση Beta, $B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$

Το p-value για έναν σύνδεσμο με βάρος w είναι η πιθανότητα να παρατηρήσουμε ένα βάρος τουλάχιστον τόσο μεγάλο όσο το w (ή μεγαλύτερο) σε μια τυχαία διαδικασία [35]. Αυτό υπολογίζεται ως το άθροισμα των πιθανοτήτων για όλα τα δυνατά βάρη από το *w* έως το *s*:

$$p - value = \sum_{x=w}^{s} P(x \mid k, s, a)$$

Για να ελεγχθεί η στατιστική σημαντικότητα όλων των συνδέσμων ταυτόχρονα, εφαρμόζεται διόρθωση Benjamini-Hochberg (FDR). Ένας σύνδεσμος θεωρείται στατιστικά σημαντικός εάν το διορθωμένο p-value του είναι μικρότερο από ένα επίπεδο σημαντικότητας α (συνήθως α=0.05) [35].



Εικόνα 4.4 Η διαδικασία της κάλπης του Pólya ως μια αναλογία για την αξιολόγηση της στατιστικής σημασίας ενός συνδέσμου σε ένα δίκτυο. Ειδικότερα, εξηγεί πώς, για έναν σύνδεσμο με βάρος w=2 σε έναν κόμβο με βαθμό k=3 και ισχύ s=4, η μέθοδος υπολογίζει το p-value ως το άθροισμα των πιθανοτήτων για όλα τα ενδεχόμενα στα οποία το βάρος είναι ίσο ή μεγαλύτερο από το 2, μετά από s κληρώσεις από την κάλπη.Η εικόνα ανήκει στο άρθρο με τίτλο “A Pólya urn approach to information filtering in complex networks” [35].

4.16 High Salience Skeleton Filter

Η μέθοδος ταξινόμησης των συνδέσμων σε σύνθετα δίκτυα, βασίζεται στην έννοια της «σημασίας συνδέσμου» (link salience). Αυτή η προσέγγιση επιτρέπει την ταξινόμηση των συνδέσμων σε διακριτές ομάδες χωρίς την ανάγκη εξωτερικών παραμέτρων, καθώς η ίδια η δομή του δικτύου καθορίζει τα κριτήρια.

Αρχικά, το δίκτυο μοντελοποιείται ως ένας συμμετρικός, σταθμισμένος πίνακας $N \, x \, N$, όπου N είναι ο αριθμός των κόμβων. Κάθε στοιχείο του πίνακα, w_{ij}, αντιπροσωπεύει την ισχύ σύνδεσης μεταξύ των κόμβων i και j. Στη συνέχεια, εισάγεται η έννοια της «αποτελεσματικής εγγύτητας» d_{ij}, η οποία ορίζεται ως το αντίστροφο της ισχύος σύνδεσης, δηλαδή $d_{ij}=1/w_{ij}$. Αυτός ο μετασχηματισμός επιτρέπει την εφαρμογή αλγορίθμων συντομότερων διαδρομών.

Στο τρίτο βήμα, υπολογίζεται η συντομότερη διαδρομή μεταξύ δύο οποιονδήποτε κόμβων ως εκείνη που ελαχιστοποιεί τη συνολική «αποτελεσματική απόσταση» $l=\Sigma \, d\left(n_i, n_j+1\right)$ δηλαδή το άθροισμα των αποτελεσματικών εγγυτήτων κατά μήκος της διαδρομής. Για κάθε κόμβο του δικτύου που λειτουργεί ως σημείο αναφοράς (r), δημιουργείται ένα «δέντρο συντομότερης διαδρομής» (T(r)). Το δέντρο αυτό συνοψίζει τις πιο αποτελεσματικές διαδρομές από τον συγκεκριμένο κόμβο προς όλους τους άλλους. Η «σημασία συνδέσμου» του δικτύου ορίζεται ως ο μέσος όρος όλων των SPTs. Το στοιχείο s_{ij} του πίνακα S ποσοτικοποιεί το ποσοστό των SPTs στα οποία συμμετέχει ένας συγκεκριμένος σύνδεσμος (i,j). Τέλος, οι σύνδεσμοι ταξινομούνται με βάση την κατανομή της σημασίας τους (p(s)). Αυτή η κατανομή παρουσιάζει μια διτροπική μορφή, επιτρέποντας μια φυσική ταξινόμηση των συνδέσμων σε δύο διακριτές ομάδες: «σημαντικοί» (με s κοντά στο 1) και «μη σημαντικοί» (με s κοντά στο 0). Η ταξινόμηση αυτή είναι εγγενής στο δίκτυο και δεν απαιτεί εξωτερικές παραμέτρους ή όρια [36].

Πίνακας 4.1 Σύνοψη των μεθόδων εξαγωγής backbone και σύντομη περιγραφή.

Κατηγορία Μεθόδου	**Μέθοδος**	**Βασική Ιδέα**
Στατιστική	Bipartite Configuration Model (BiCM)	Εντοπίζει σημαντικές ακμές μέσω τυχαιοποίησης διατηρώντας τους βαθμούς των κόμβων και σύγκρισης του με το εξεταζόμενο δίκτυο.
	Stochastic Degree Sequence Model (SDSM)	Πιο ευέλικτη εκδοχή του BiCM, διατηρεί κατά μέσο όρο τους βαθμούς των κόμβων στην τυχαιοποίηση.
	Disparity Filter	Κρατά συνδέσμους ασυνήθιστα ισχυρούς για έναν συγκεκριμένο κόμβο.
	Marginal Likelihood Filter (MLF)	Εκτιμά τη σημασία ενός δεσμού βάσει των δύο συνδεδεμένων κόμβων.
	Noise Corrected Filter (NCF)	Κρατά συνδέσμους ισχυρότερους από τον τυχαίο θόρυβο.
	Hypergeometric Test	Εξετάζει αν δύο κόμβοι μοιράζονται περισσότερους συνδέσμους απ' ό,τι αναμένεται.
	Pólya Filter	Εντοπίζει σημαντικούς συνδέσμους σε

		αναπτυσσόμενα δίκτυα.
Δομική	Global Threshold	Κρατά συνδέσεις ισχυρότερες από μια σταθερή τιμή.
	Maximal Spanning Tree (MST)	Βρίσκει τον "σκελετό" που συνδέει όλους τους κόμβους με το μεγαλύτερο βάρος.
	Edge Betweenness	Αφαιρεί συνδέσμους που συχνά λειτουργούν ως "γέφυρες" στο δίκτυο.
	High Salience Skeleton	Κρατά συνδέσμους κρίσιμους για συντομότερα μονοπάτια.
Υβριδική	Hierarchical Backbone	Αναδεικνύει ιεραρχικές σχέσεις μεταξύ κόμβων.
	Locally Adaptive Network Sparsification (LANS)	Μη παραμετρική μέθοδος που προσαρμόζεται στη δομή του τοπικού δικτύου.
	Multiple Linkage Analysis (MLA)	Καθορίζει τον βέλτιστο αριθμό σημαντικών συνδέσμων ανά κόμβο.
	Modularity Filter	Αφαιρεί κόμβους που συνεισφέρουν ελάχιστα στη δομή κοινοτήτων του δικτύου.
	Doubly Stochastic Filter	Μετασχηματίζει τον δίκτυο και κατασκευάζει ένα backbone.

Κεφάλαιο 5

5. Αξιολόγηση του Σκελετού - Μετρικές

Η αξιολόγηση του backbone είναι κρίσιμη διότι μας επιτρέπει να κατανοήσουμε κατά πόσο η απλοποιημένη εκδοχή του αρχικού δικτύου διατηρεί τα ουσιώδη χαρακτηριστικά του. Αν ο κορμός δεν είναι αντιπροσωπευτικός, μπορεί να οδηγήσει σε εσφαλμένα συμπεράσματα όταν χρησιμοποιείται για ανάλυση. Επιπρόσθετα εξασφαλίζει ότι περιέχονται οι πιο σχετικές πληροφορίες του αρχικού δικτύου, ενώ διατηρεί τη δομή και τις σχέσεις μεταξύ κόμβων, βοηθάει στο να αξιολογηθεί εάν έχουν εισαχθεί σημαντικές αλλαγές που αλλοιώνουν την πραγματική εικόνα του δικτύου, και τέλος επιτρέπει πιο γρήγορες και αποδοτικές αναλύσεις χωρίς απώλεια κρίσιμης πληροφορίας [17].

Προκειμένου να αξιολογηθεί η δομική και στατιστική ποιότητα του backbone και η σύγκριση του με το αρχικό δίκτυο χρησιμοποιήθηκαν και κατασκευάστηκαν διάφορες μετρικές που υπάρχουν στην θεωρία γραφημάτων και ελέγχουν αν ο εξαγόμενος κορμός διατηρεί τη συνολική δομή του δικτύου, τις σχέσεις μεταξύ των κόμβων, και αν μπορεί να αναπαραστήσει σωστά την αρχική πληροφορία με λιγότερες ακμές. Ανάλογα με το υπό εξέταση χαρακτηριστικό που επιλέγουμε, μετρήθηκε η ποσοστιαία μεταβολή του, στο νέο δίκτυο, επιλέγεται η αντίστοιχη μετρική που το ορίζει και κάθε ένα παρέχει διαφορετική πληροφορία σχετικά με την τοπολογία του δικτύου και τη συμπεριφορά του backbone. Ακολούθως ορίζονται δημοφιλείς μετρικές, που χρησιμοποιούνται μεταξύ άλλων για τον σκοπό αυτόν.

Πυκνότητα (Density): Μετρά τον αριθμό των συνδέσεων του δικτύου σε σχέση με τις πιθανές συνδέσεις [17].

$$\omega = \frac{2E}{V(V-1)},$$

όπου E είναι οι ακμές και V οι κόμβοι.

Διάμετρος (Diameter): Μετρά το μέγιστο μήκος της συντομότερης διαδρομής μεταξύ δύο κόμβων [17].

$$d = Max(d_{ij})$$

Μέσο μήκος συντομότερης διαδρομής (Average shortest path length): Μετρά τη μέση απόσταση μεταξύ όλων των ζευγών κόμβων. Όπου ⟨k⟩ είναι ο μέσος βαθμός κόμβου [17].

$$L = \frac{\ln(V)}{\ln(k)}$$

Μέσος βαθμός κόμβων (Average node degree): ο μέσος αριθμός συνδέσεων ανά κόμβο [17].

$$\langle k \rangle = \sum_i k_i$$

Μέγιστος Βαθμός κόμβων: Ο μέγιστος αριθμός συνδέσεων που έχει κάποιος κόμβος [17].

Συντελεστής συσχετισμού (Assortativity coefficient) : Μετρά αν οι κόμβοι συνδέονται με άλλους κόμβους παρόμοιου βαθμού, με σύνολο τιμών [-1 , 1] όταν είναι θετικό οι κόμβοι συνδέονται με κόμβους παρόμοιου βαθμού. Όταν είναι αρνητικό οι κόμβοι συνδέονται με κόμβους πολύ διαφορετικού βαθμού [17].

$$r = \frac{\sum_i \left(k_i k_{nni}\right) - \left[\sum_i 0{,}5 \cdot \left(k_i + k_{nmi}\right)\right]^2}{\left[\sum_i 0{,}5 \cdot \left(k_i^2 + k_{nni}^2\right)\right] - \left[\sum_i 0{,}5 \cdot \left(k_i + k_{nmi}\right)\right]^2}$$

με k_i να δηλώνει το βαθμό ενός κόμβου *i* και k_{nni}είναι ο μέσος βαθμός των κόμβων που συνδέονται σε έναν κόμβο i [17].

Συντελεστής συσσώρευσης (Clustering coefficient): Μετρά πόσο πιθανό είναι οι γείτονες ενός κόμβου να είναι επίσης συνδεδεμένοι μεταξύ τους (δηλαδή αν σχηματίζουν τρίγωνα) [17].

$$C \equiv \frac{\langle n_i \rangle}{\langle k_i \rangle \frac{\left(k_i - 1\right)}{2}}$$

όπου το $\langle n_i \rangle$ αντιπροσωπεύει τον αριθμό των συνδέσεων μεταξύ των πλησιέστερων γειτόνων του ο κόμβος i, το $\langle k_i \rangle$αντιπροσωπεύει το βαθμό του κόμβου i.

Modularity: Ποσοτικοποιεί τη διαφορά μεταξύ του πραγματικού αριθμού ακμών που πέφτουν μέσα στις κοινότητες (που βρέθηκαν από τον αλγόριθμο) και του αναμενόμενου αριθμού ακμών μέσα στις ίδιες ομάδες σε ένα τυχαίο δίκτυο με την ίδια ακριβώς κατανομή βαθμών [17].

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] delta\left(c_i, c_j\right)$$

Όπου:

- m: Συνολικός αριθμός ακμών.
- A_{ij}: Βάρος της ακμής μεταξύ i και j.
- k_i, k_j: Βαθμοί των κόμβων i και j.
- $\frac{k_i, k_j}{2 * m}$: Αναμενόμενη τιμή ακμής σε τυχαίο δίκτυο (null model).
- $\delta\left(c_i, c_j\right)$ *Kronecker delta* είναι ίσο με 1 αν οι κόμβοι i και j ανήκουν στην ίδια κοινότητα c, και 0 διαφορετικά [17].

Κεντρικότητα Ενδιάμεσης Θέσης (Betweenness Centrality): Μετρά τον βαθμό στον οποίο ένας κόμβος v λειτουργεί ως γέφυρα ή διαμεσολαβητής στη ροή πληροφορίας, υπολογίζοντας πόσες φορές βρίσκεται στην πιο σύντομη διαδρομή μεταξύ οποιουδήποτε ζεύγους άλλων κόμβων [17].

$$C = \frac{1}{N} \sum_{i=1}^{N} C_i \quad \text{όπου} \quad C_i = \frac{2 E_i}{k_i (k_i - 1)}$$

- C_i: Ο τοπικός συντελεστής συσσώρευσης του κόμβου i. Πρόκειται για το μέτρο της τοπικής συνεκτικότητας γύρω από τον συγκεκριμένο κόμβο.
- E_i: Ο αριθμός των συνδέσεων (ακμών) μεταξύ των πλησιέστερων γειτόνων του κόμβου i. Αντιπροσωπεύει τον αριθμό των τριγώνων στα οποία συμμετέχει ο κόμβος i.
- k_i: Ο βαθμός (αριθμός συνδέσεων) του κόμβου i. Δείχνει πόσους άμεσους γείτονες έχει ο κόμβος i.
- N: Ο συνολικός αριθμός κόμβων στο δίκτυο. Αντιπροσωπεύει το μέγεθος του δικτύου [17].

Κεφάλαιο 6

6. Αποτελέσματα

Στο πλαίσιο της εφαρμογής των μεθόδων που υλοποιήθηκαν, ελέγξαμε τη λειτουργία τους με δύο διαφορετικά δίκτυα, το δίκτυο μονομερούς προβολής ως προς τα γονίδια, με 3.579 κόμβους και 140.826 ακμές, και το δεύτερο ως προς τις ασθένειες, με 954 κόμβους ασθενειών και 5.667 ακμές. Για την επικύρωση των αποτελεσμάτων χρησιμοποιήθηκαν δειγματοληπτικά τόσο στατιστικές, όσο και δομικές μέθοδοι, για να εξετάσουμε στατιστικά την αποτελεσματικότητα εξαγωγής κορμού των δικτύων αλλά και δομικά ως προς την τοπολογία τους. Για την αναπαράσταση των αποτελεσμάτων χρησιμοποιήθηκε το λογισμικό Gephi, ένα από τα πιο δημοφιλή και ισχυρά, δωρεάν και ανοιχτού κώδικα (open-source) λογισμικά για ανάλυση και οπτικοποίηση δικτύων (Network Analysis and Visualization). Το λογισμικό είναι διαθέσιμο στον διαδικτυακό ιστότοπο https://gephi.org/. Το Gephi παίρνει ως είσοδο από τον χρήστη δεδομένα που περιγράφουν σχέσεις μεταξύ αντικειμένων σε συγκεκριμένη δομή με επικεφαλίδες Source, Target και προεραιτικά Weight για να αναγνωρίσει σωστά τα δεδομένα και τα μετατρέπει σε οπτικά δίκτυα (ή γραφήματα), τα οποία μπορεί ο χρήστης να εξερευνήσει, να αναλύσει και να χειριστεί σε πραγματικό χρόνο.

Αφού υπολογίστηκαν τα στατιστικά για τα αρχικά one-mode projected δίκτυα (gene-gene και disease-disease), η πρώτη φάση της αξιολόγησης επικεντρώθηκε στη στατιστική ανάλυση του εξαγόμενου κορμού. Συγκεκριμένα, εξετάσαμε τη διατήρηση των βασικών στατιστικών ιδιοτήτων των αρχικών δικτύων στον κορμό, όπως η κατανομή βαθμών, η μέση διαμεσότητα (betweenness centrality) και ο συντελεστής clustering coefficient. Η βασική παραδοχή ήταν ότι ένας αποτελεσματικός αλγόριθμος εξαγωγής κορμού πρέπει να διατηρεί τις κρίσιμες πληροφορίες και την ετερογένεια του δικτύου, ενώ ταυτόχρονα να αφαιρεί τις πλεονάζουσες ή λιγότερο σημαντικές ακμές. Τα στατιστικά αποτελέσματα έδειξαν ότι οι μέθοδοί μας επιτυγχάνουν σημαντική μείωση του αριθμού των ακμών (της τάξεως του 60,18% για το δίκτυο γονιδίων (για το Modularity Filter) και 65,4% για το δίκτυο ασθενειών, όπου 65,4% είναι το ποσοστό μείωσης των ακμών από 5667 σε 1962 για το Modularity Filter) με ελάχιστη απώλεια στην ακρίβεια της κατανομής βαθμών και στη μέση διαμεσότητα.

Η δεύτερη φάση αφορούσε την δομική και τοπολογική επικύρωση. Εδώ, η έμφαση δόθηκε στη διατήρηση της δομής των σημαντικών συνιστωσών και της αποδοτικότητας της ροής πληροφορίας του δικτύου. Χρησιμοποιήθηκαν δείκτες όπως η μέση συντομότερη διαδρομή (average shortest path length) και η μεγαλύτερη συνεκτική συνιστώσα (largest connected component - LCC). Στο δίκτυο γονιδίων, παρατηρήθηκε ότι ο εξαγόμενος κορμός διατηρούσε μια εξαιρετικά υψηλή συνδεσιμότητα, με την LCC να καλύπτει άνω του 100% των κόμβων (π.χ. MST/Doubly Stochastic), τιμή συγκρίσιμη με εκείνη του πλήρους δικτύου. Αυτό υποδηλώνει ότι οι μέθοδοί μας δεν διασπούν τις βιολογικά σημαντικές συσχετίσεις μεταξύ των γονιδίων στο συγκεκριμένο σύνολο δεδομένων. Επιπλέον, η μέση συντομότερη διαδρομή παρουσίασε μόνο μια οριακή αύξηση Delta L=1,89 για το SDSM, επιβεβαιώνοντας ότι η αποδοτικότητα της δικτυακής επικοινωνίας παραμένει σε υψηλά επίπεδα. Αντίστοιχα, στο δίκτυο ασθενειών, η εφαρμογή των δομικών μεθόδων ανέδειξε τη διατήρηση των βασικών κοινοτήτων ασθενειών (disease clusters). Η τοπολογική ανάλυση επιβεβαίωσε την αποτελεσματική διατήρηση των πυρήνων του δικτύου, δηλαδή των κόμβων και των ακμών που φέρουν τη μέγιστη πληροφορία. Ειδικότερα, η μέθοδος Modularity Filter παρουσίασε τον πιο πυκνό κορμό (Πυκνότητα: 0,05, Μέσος Όρος Clustering: 0,72) με τη μικρότερη διάμετρο με τιμή 6,

καθιστώντας την καταλληλότερη για τη μελέτη συστάδων με υψηλή συνεκτικότητα. Αντίθετα, μέθοδοι όπως το SDSM (Διάμετρος: 24 και η Kendall Correlation (Διάμετρος: 11) διατήρησαν μεγάλο αριθμό κόμβων και στατιστικά σημαντικές συνδέσεις, αλλά μείωσαν την αποτελεσματικότητα της διάδοσης πληροφορίας.

Συνοψίζοντας, η συνδυαστική χρήση στατιστικών και δομικών δεικτών παρείχε μια ολιστική και ισχυρή επικύρωση της ικανότητας των υλοποιημένων μεθόδων να εξάγουν έναν πυκνό και πληροφοριακά πλούσιο κορμό από τα αρχικά σύνθετα δίκτυα, διατηρώντας ταυτόχρονα τις κρίσιες ιδιότητες τόσο σε επίπεδο μεμονωμένων κόμβων όσο και σε επίπεδο συνολικής δικτυακής τοπολογίας. Οι διαφορετικές τοπολογικές δομές που προκύπτουν από κάθε μέθοδο, από τον ελάχιστο σκελετό των MST/HSS έως την υψηλή ομαδοποίηση του Modularity Filter, αναδεικνύουν τη σημασία της επιλογής του κατάλληλου αλγορίθμου ανάλογα με τον στόχο της βιολογικής ανάλυσης.

6.1 Αποτελέσματα Δικτύου Γονιδίων

Η ερμηνεία των αποτελεσμάτων για το δίκτυο γονιδίων, όπως παρουσιάζονται στους Πίνακες 6.1 και 6.2, αναδεικνύει τις διαφορετικές τοπολογικές δομές που προκύπτουν από κάθε μέθοδο εξαγωγής κορμού, γεγονός που αντικατοπτρίζει τη διαφορετική στόχευση κάθε αλγορίθμου, δηλαδή είτε την αραίωση με διατήρηση της δομής, είτε την εστίαση στη διατήρηση της πυκνότητας και της συσσωμάτωσης. Για τον συντελεστή Kendall με χρήση της μεθόδου Global Threshold Filter αν και η τέλεια θετική συσχέτιση αντιστοιχεί στην τιμή 1.0 προς αποφυγή ενός υπερβολικά αραιού δικτύου χρησιμοποιήθηκε η τιμή 0.8 για να διατηρηθούν οι ακμές που έχουν βάρος μεγαλύτερο η ίσο από αυτή την τιμή.

Πίνακας 6.1 Στατιστική Σύγκριση των εξαγόμενων Backbone στο δίκτυο γονιδίων. Σημειώνεται ότι τα μέτρα που δεν αντιστοιχούν σε ακέραιους συντελεστές έχουν στρογγυλοποιηθεί σε ακρίβεια δύο δεκαδικών ψηφίων.

Μέθοδος	Αριθμός Κόμβων	Αριθμός Ακμών	Πυκνότητα	Μέσος όρος Clustering	Μέσο Συντομότερο Μονοπάτι
One Mode Projection (Αρχικό Γράφημα)	3581	140827	0.02	0,85	3.15
Modularity Filter	1073	56073	0,1	0,92	2.35
Maximum Spanning Tree	3581	3551	0	0	12.27
SDSM	1088	1406	0	0,49	5.04
Polya’s Method	80	182	0,06	0,9	1.0
Disparity Filter	97	177	0,04	0,41	2.96
High Salience Skeleton	1482	1863	0	0,03	4.57
Kendall Correlation	70	46	0,02	0.97	1.04
Doubly Stochastic	3542	4572	0	0,14	15.59

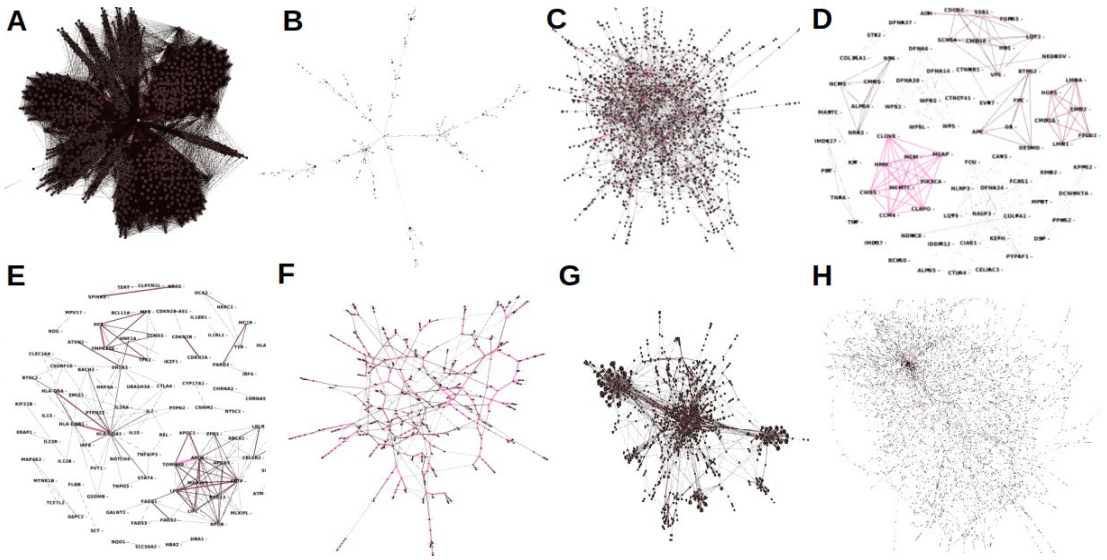
Πίνακας 6.2 Στατιστική Σύγκριση των εξαγόμενων Backbone στο δίκτυο γονιδίων.

Μέθοδος	Μέσος Βαθμός	Ελάχ. Βαθμός	Μέγ. Βαθμός	Διάμετρος
One Mode Projection (Αρχικό Γράφημα)	78.65	1	652	9
Modularity Filter	104,52	38	272	4
Maximum Spanning Tree	1,98	1	126	30
SDSM	2,58	1	12	12
Polya's Method	4,55	1	8	1
Disparity Filter	3,6	1	20	8
High Salience Skeleton	2,51	1	105	10
Kendall Correlation	1,31	1	3	1
Doubly Stochastic	2,58	1	6	48

Παρατηρείται μια ξεκάθαρη αντίθεση μεταξύ των μεθόδων ως προς τον βαθμό αραίωσης. Το Modularity Filter παράγει ένα δίκτυο με εξαιρετικά υψηλή πυκνότητα (0,1) και μέσο βαθμό 104,52 και τον μεγαλύτερο αριθμό ακμών (56.073) όπως φαίνονται στα αποτελέσματα (βλ. Πίνακα 6.1, Πίνακα 6.2), διατηρώντας όλες τις συνδέσεις που συμβάλλουν στον σχηματισμό βέλτιστων κοινοτήτων. Η οπτική απεικόνιση αυτού του πυκνού και συνεκτικού κορμού παρουσιάζεται στην Εικόνα 6.1 A (Modularity). Αντιθέτως, οι μέθοδοι MST και Doubly Stochastic διατηρούν τον μεγαλύτερο αριθμό κόμβων (3.581 και 3.542 αντίστοιχα), αλλά είναι εξαιρετικά αραιά (0,0005 και 0,0007 αντίστοιχα). Αυτή η διαφοροποίηση επιβεβαιώνει ότι το MST και το Doubly Stochastic εστιάζουν στην εξαγωγή του ελάχιστου δομικού σκελετού για τη διατήρηση της συνολικής συνδεσιμότητας. Ο σκελετός του MST φαίνεται στην Εικόνα 6.1 B (MST), ενώ η πιο σύνθετη αραιή δομή του Doubly Stochastic παρουσιάζεται στο Εικόνα 6.1 H (Doubly Stochastic). Στον αντίποδα, οι μέθοδοι Kendall Correlation Threshold (70 κόμβοι) και Polya's Method (80 κόμβοι) είναι οι πιο επιθετικές, αφήνοντας πίσω μόνο τον πυρήνα των ισχυρότερων και πιο στατιστικά σημαντικών συνδέσεων, (βλ. Εικόνες 6.1 G και 6.1 D). Οι κορμοί που προκύπτουν από το Disparity Filter (99 κόμβοι), το SDSM (1.088 κόμβοι) και το High Salience Skeleton χαρακτηρίζονται από πυκνότητα χαμηλή, επιτρέποντας σχέσεις στατιστικά σημαντικές ακμές να διαφοροποιηθούν (1.482 κόμβοι) (βλ. Εικόνες 6.1 E, 6.1 C, 6.1 F, Πίνακας 6.3).

Η ανάλυση των δεικτών συνδεσιμότητας αποκαλύπτει τη λειτουργική δομή των εξαγόμενων κορμών. Το Modularity Filter εμφανίζει τον υψηλότερο μέσο βαθμό (104,52), επιβεβαιώνοντας την υψηλή συνεκτικότητα και τη λειτουργική πολυλειτουργικότητα των γονιδίων εντός των κοινοτήτων που διατηρεί (βλ. Εικόνα 6.1 A Πίνακα 6.3). Αντίθετα, το MST και το Doubly Stochastic έχουν πολύ χαμηλό μέσο βαθμό (1,98 και 2,58 αντίστοιχα), που είναι χαρακτηριστικό αραιών ή δενδρικών δικτύων (βλ. Εικόνες 6.1 B και 6.1 H). Ως προς την αποτελεσματικότητα της διάδοσης πληροφορίας, η διάμετρος αποτελεί βασικό δείκτη. Οι μέθοδοι Modularity Filter, Polya's Method και Kendall Correlation Threshold παρουσιάζουν εξαιρετικά μικρή διάμετρο 4, 1 και 1 αντίστοιχα (βλ. Πίνακα 6.2). Αντίθετα, το Doubly Stochastic έχει τη μεγαλύτερη διάμετρο (48), και το MST μία μεγάλη διάμετρο (30). Η αύξηση της διαμέτρου σημαίνει ότι ο κορμός έχει επιμηκυνθεί, υποδηλώνοντας ότι, ενώ διατηρήθηκε η συνολική συνδεσιμότητα, η αποτελεσματικότητα της διάδοσης πληροφορίας μεταξύ των πιο απομακρυσμένων γονιδίων έχει μειωθεί σημαντικά σε σχέση με το πλήρες δίκτυο (βλ. Εικόνες 6.1 B και 6.1 H).

Συνοψίζοντας, η ανάλυση επιβεβαιώνει ότι η Modularity Filter είναι ιδανική για τη μελέτη των συστάδων γονιδίων, η MST για την ανακάλυψη του ελάχιστου ιεραρχικού σκελετού, και οι Doubly Stochastic/SDSM για τον εντοπισμό των στατιστικά σημαντικών συνδέσεων διατηρώντας μεγάλο αριθμό κόμβων.



Εικόνα 6.1 Οπτική απεικόνιση των εξαγόμενων κορμών (backbones) του δικτύου γονιδίων με χρήση του λογισμικού Gephi, συγκρίνοντας οκτώ διαφορετικές μεθόδους αραίωσης. Κάθε επιμέρους δίκτυο (A-H) αντιπροσωπεύει τη διαφορετική τοπολογική δομή που διατηρείται από τον αντίστοιχο αλγόριθμο. Οι κόμβοι με κόκκινο χρώμα και ακμές επισημαίνουν γονίδια και συνδέσεις που έχουν ιδιαίτερη στατιστική ή τοπολογική σημασία (π.χ. υψηλή σημαντικότητα ή συμμετοχή σε συγκεκριμένες συστάδες).

Πίνακας 6.3 Ο πίνακας παραθέτει τις μεθόδους εξαγωγής κορμού από δίκτυα, μαζί με τον στόχο και το κύριο χαρακτηριστικό κάθε μεθόδου. Κάθε μέθοδος χαρακτηρίζεται από διαφορετική στρατηγική στην επιλογή και διατήρηση των ακμών του δικτύου, ανάλογα με τις ανάγκες της ανάλυσης: από την αναγνώριση κοινοτήτων και την εξαιρετική συνδεσιμότητα μέχρι την επιθετική αραίωση και την ανάδειξη σημαντικών ακμών.

Διάγραμμα	Μέθοδος Εξαγωγής Κορμού	Κύριο Χαρακτηριστικό
A	Modularity Filter	Εξαιρετικά πυκνό δίκτυο. Διατήρηση όλων των συνδέσεων που συμβάλλουν στον σχηματισμό βέλτιστων κοινοτήτων.
B	Maximum Spanning Tree (MST)	Αραιός, δενδρικός σκελετός. Διατήρηση της συνολικής συνδεσιμότητας με τον ελάχιστο αριθμό ακμών.
C	SDSM	Αρκετά αραιό δίκτυο. Εντοπισμός στατιστικά σημαντικών συνδέσεων.
D	Polya’s Method	Πολύ μικρός και πυκνός πυρήνας. Σημαντική αραίωση, διατήρηση μόνο των ισχυρότερων συνδέσεων.
E	Disparity Filter	Μικρός, πυκνός κορμός. Διατήρηση τοπικά σημαντικών συνδέσεων με βάση τη δύναμη του κόμβου.

F	High Salience Skeleton (HSS)	Μέτριος, αραιός κορμός. Εντοπισμός ακμών με υψηλή σπουδαιότητα (salience).
G	Kendall Correlation Threshold	Ο μικρότερος, εξαιρετικά πυκνός πυρήνας (thresholding). Διατήρηση των ισχυρότερων συσχετίσεων.
H	Doubly Stochastic	Πολύ αραιό δίκτυο. Εστίαση στη διατήρηση της συνολικής συνδεσιμότητας με μεγάλη διάμετρο.

6.2 Αποτελέσματα Δικτύου Ασθενειών

Η ερμηνεία των αποτελεσμάτων για το δίκτυο ασθενειών, αναδεικνύει σαφώς τις διαφορετικές τοπολογικές δομές που προκύπτουν από κάθε μέθοδο εξαγωγής κορμού (βλ. Πίνακες 6.4 και 6.5). Αυτή η ποικιλομορφία αντικατοπτρίζει τη διαφορετική στόχευση κάθε αλγορίθμου είτε την σημαντική αραίωση για την αποκάλυψη του ελάχιστου δομικού σκελετού, είτε τη διατήρηση της υψηλής πυκνότητας και των συστάδων. Παρατηρείται μια ξεκάθαρη διαφορά ως προς την πυκνότητα και το μέγεθος των εξαγόμενων κορμών, με το Modularity Filter να κυριαρχεί στη διατήρηση της δομής. Συγκεκριμένα, το Modularity Filter παράγει τον πιο πυκνό κορμό (Πυκνότητα: 0.05, βλ. Πίνακας 6.3) με τον μεγαλύτερο αριθμό ακμών (1962) και κόμβων (286). Αυτό επιβεβαιώνεται από τον υψηλότερο Μέσο Όρο Clustering (0.72) και τον υψηλότερο Μέσο Βαθμό (13.72), υποδεικνύοντας ένα δίκτυο με ισχυρή τοπική συνεκτικότητα και ομαδοποίηση των ασθενειών, (βλ. Εικόνα 6.2 A) . Αντίθετα, μέθοδοι όπως η SDSM, η Polya’s Method, το Disparity Filter, και η Kendall Correlation Threshold παράγουν εξαιρετικά αραιά δίκτυα με πυκνότητα ίση ή κοντά στο 0.0. Η Polya’s Method διατηρεί τον χαμηλότερο μέσο βαθμό (1.07), γεγονός που είναι ενδεικτικό ενός σχεδόν αποσυνδεδεμένου ή δενδρικού σκελετού, όπως φαίνεται στην Εικόνα 6.2 D (Polya’s Method), η οποία είναι ιδιαίτερα αραιή. Η εικόνα Εικόνα 6.2 C (SDSM) δείχνει επίσης έναν πολύ αραιό σκελετό.

Αξιοσημείωτο είναι ότι το Maximum Spanning Tree (MST) και το High Salience Skeleton (HSS) έχουν ταυτόσημα αποτελέσματα ως προς τα βασικά τοπολογικά μεγέθη π.χ. Κόμβοι: 92, Ακμές: 122, Μέσος Βαθμός: 2.65, (βλ. Πίνακα 6.4, Πίνακα 6.5). Αυτή η ταύτιση υποδηλώνει ότι για το συγκεκριμένο δίκτυο ασθενειών, ο ελάχιστος δομικός σκελετός (MST) συμπίπτει με τη δομή που διατηρεί τις ακμές με την υψηλότερη σημασία (HSS) και πρόκειται για αρκετά αραιές δομές (βλ. Εικόνες 6.2 B και Εικόνες 6.2 F), με την εικόνα B να παρουσιάζει μεγαλύτερη ομοιομορφία στην αραίωση. Η Doubly Stochastic μέθοδος, αν και αραιή, διατηρεί μεγαλύτερο μέσο όρο clustering 0.45 (βλ. Πίνακα 6.4), υποδηλώνοντας τη διατήρηση κάποιων τοπικών συσσωματώσεων, όπως διακρίνεται στην πιο σύνθετη αραιή δομή της Εικόνας 6.2 H (Doubly Stochastic).

Η ανάλυση της διαμέτρου (δείκτης αποτελεσματικότητας της διάδοσης πληροφορίας) αποκαλύπτει σημαντικές διαφορές στη λειτουργική δομή. Το Modularity Filter παρουσιάζει τη μικρότερη διάμετρο με τιμή 6, ακολουθούμενο από την Polya’s Method με τιμή 3 και το Disparity Filter με τιμή 5 (βλ. Πίνακα 6.5). Η χαμηλή διάμετρος υποδηλώνει ότι οι λειτουργικές συνδέσεις μεταξύ των ασθενειών διαδίδονται αποτελεσματικά. Η αύξηση της διαμέτρου σημαίνει ότι ο κορμός έχει επιμηκυνθεί, υποδηλώνοντας ότι, ενώ διατηρήθηκε η συνολική συνδεσιμότητα, η

αποτελεσματικότητα της διάδοσης πληροφορίας μεταξύ των πιο απομακρυσμένων κόμβων έχει μειωθεί σημαντικά σε σχέση με το πλήρες δίκτυο.

Πίνακας 6.4 Στατιστική Σύγκριση των εξαγόμενων Backbone στο δίκτυο ασθενειών.

Μέθοδος	Αριθμός Κόμβων	Αριθμός Ακμών	Πυκνότητα	Μέσος όρος Clustering	Μέσο Συντομότερο Μονοπάτι
One Mode Projection (Αρχικό Γράφημα)	1269	5694	0.01	0.72	3.15
Modularity Filter	286	1962	0.05	0.72	2.86
Maximum Spanning Tree	92	122	0.03	0.22	7.32
SDSM	1000	966	0.0	0.0	1.67
Polya’s Method	169	90	0.01	0.0	2.82
Disparity Filter	267	157	0.0	0.0	3.91
High Salience Skeleton	92	122	0.03	0.22	4.27
Kendall Correlation	881	1007	0.0	0.01	1.0
Doubly Stochastic	173	162	0.01	0.45	10.21

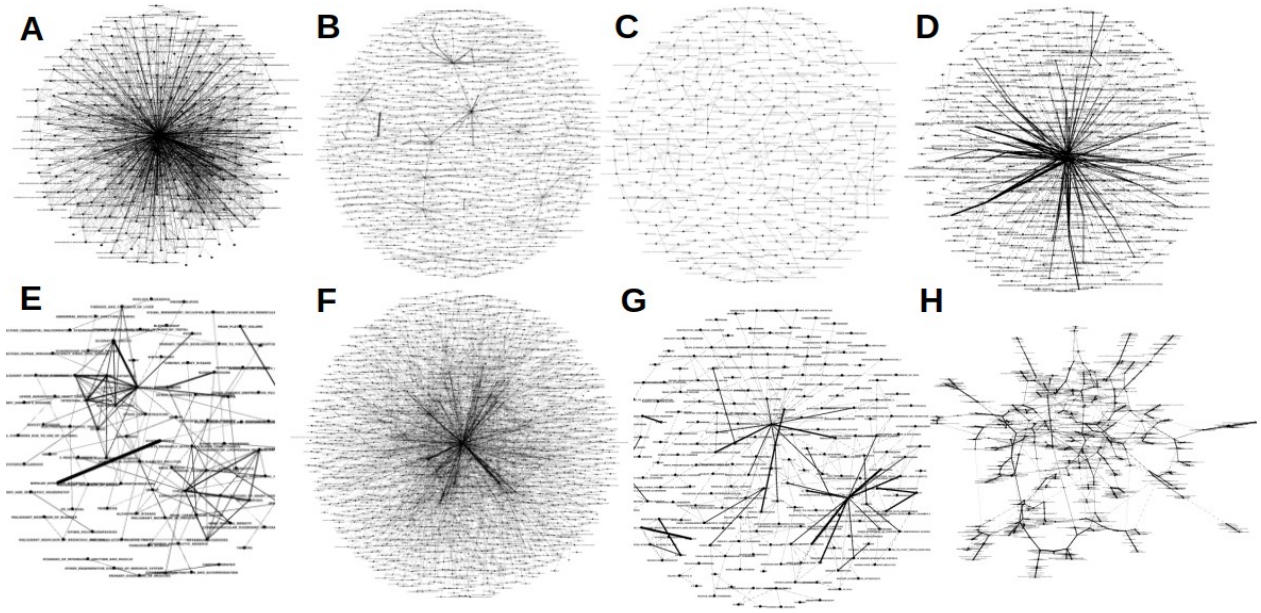
Πίνακας 6.5 Στατιστική Σύγκριση των εξαγόμενων Backbone στο δίκτυο ασθενειών.

Μέθοδος	Μέσος Βαθμός	Ελάχ. Βαθμός	Μέγ. Βαθμός	Διάμετρος
One Mode Projection (Αρχικό Γράφημα)	11.39	1	162	9
Modularity Filter	13.72	1	79	6
Maximum Spanning Tree	2.65	1	17	8
SDSM	1.93	1	49	24
Polya’s Method	1.07	1	2	3
Disparity Filter	1.18	1	6	5
High Salience Skeleton	2.65	1	17	8
Kendall Correlation	2.29	1	77	11
Doubly Stochastic	1.87	1	4	45

Αντίθετα, το SDSM εμφανίζει τη μεγαλύτερη διάμετρο τιμής 24 και ακολουθεί η Kendall Correlation με τιμή 11. Αυτή η μεγάλη διάμετρος υποδηλώνει ότι, αν και αυτές οι μέθοδοι διατηρούν ένα μεγάλο αριθμό κόμβων, ο κορμός έχει επιμηκυνθεί σημαντικά. Επομένως, η αποτελεσματικότητα της διάδοσης πληροφορίας μεταξύ των πιο απομακρυσμένων ασθενειών έχει μειωθεί, με την εστίαση να

είναι στη διατήρηση των στατιστικά σημαντικών συνδέσεων παρά στη βέλτιστη διαδρομή. Τέλος, το MST και το HSS έχουν μέτρια διάμετρο (8), η οποία είναι αναμενόμενη για δενδρικές δομές.

Συνοψίζοντας, η Modularity Filter είναι η καταλληλότερη για τη μελέτη των συστάδων συσχετιζόμενων ασθενειών με υψηλή συνεκτικότητα και αποτελεσματικότητα (βλ. Πίνακα 6.6). Αντίθετα, οι μέθοδοι MST και HSS είναι κατάλληλες για την ανακάλυψη του ελάχιστου ιεραρχικού σκελετού, ενώ το SDSM και η Kendall Correlation εντοπίζουν τις στατιστικά σημαντικές συνδέσεις θυσιάζοντας την αποτελεσματικότητα της διάδοσης πληροφορίας. Η Disparity Filter και η Kendall Correlation (βλ. Εικόνες 6.2 E, 6.2 G) ολοκληρώνουν την οπτική σύγκριση, δείχνοντας τις τοπολογίες που προκύπτουν από τις υπόλοιπες μεθόδους.



Εικόνα 6.2 Οπτική απεικόνιση των εξαγόμενων κορμών (backbones) με χρήση του λογισμικού Gephi του δικτύου ασθενειών με χρήση του λογισμικού Gephi, συγκρίνοντας οκτώ διαφορετικές μεθόδους αραίωσης. Κάθε επιμέρους δίκτυο (A-H) αντιπροσωπεύει τη διαφορετική τοπολογική δομή που διατηρείται από τον αντίστοιχο αλγόριθμο. Οι κόμβοι με μαύρο χρώμα και ακμές επισημαίνουν ασθένειες και συνδέσεις που έχουν ιδιαίτερη στατιστική ή τοπολογική σημασία (π.χ. υψηλή σημαντικότητα ή συμμετοχή σε συγκεκριμένες συστάδες).

Πίνακας 6.6 Στατιστική Σύγκριση των εξαγόμενων Backbone στο δίκτυο ασθενειών.

Διάγραμμα	Μέθοδος Εξαγωγής Κορμού	Ερμηνεία Οπτικής Αναπαράστασης
A	Modularity Filter	Εξαιρετικά πυκνό δίκτυο, συγκεντρωμένο γύρω από έναν κεντρικό κόμβο. Διατηρεί όλες τις ισχυρές συνδέσεις που συμβάλλουν στον σχηματισμό βέλτιστων, συνεκτικών συστάδων ασθενειών. Έχει τον υψηλότερο μέσο βαθμό.
B	Maximum Spanning Tree (MST)	Αραιός, δενδρικός σκελετός με μικρό αριθμό κόμβων. Διατηρεί την ελάχιστη δομή για τη συνολική συνδεσιμότητα, με μεγάλο αριθμό απομακρυσμένων κόμβων και μέτρια διάμετρο.
C	SDSM	Πολύ αραιό δίκτυο με μεγάλο αριθμό κόμβων και εξαιρετικά μεγάλη διάμετρο (24). Εστιάζει στη διατήρηση στατιστικά σημαντικών συνδέσεων, ακόμη κι αν αυτές δημιουργούν μακρές διαδρομές.

D	Polya’s Method	Ο πιο αραιός πυρήνας (Μέσος Βαθμός: 1.07). Πολύ επιθετική αραίωση, διατηρώντας μόνο τις ισχυρότερες συνδέσεις, με αποτέλεσμα μια σχεδόν αποσυνδεδεμένη δομή.
E	Disparity Filter	Μικρός, αραιός κορμός με χαμηλό μέσο βαθμό. Διατηρεί τις τοπικά σημαντικές συνδέσεις κάθε ασθένειας, με αποτέλεσμα να διακρίνονται κάποιες τοπικές συσσωματώσεις.
F	High Salience Skeleton (HSS)	Αραιός, δενδρικός σκελετός με δομή παρόμοια με το MST (βλ. Β). Διατηρεί τις ακμές με την υψηλότερη σπουδαιότητα (salience), οι οποίες συμπίπτουν με τον ελάχιστο δομικό σκελετό σε αυτό το δίκτυο.
G	Kendall Correlation	Αραιό δίκτυο που διατηρεί μεγάλο αριθμό κόμβων (881). Διατηρεί τις συνδέσεις πάνω από ένα κατώφλι στατιστικής συσχέτισης, με αποτέλεσμα έναν αραιό, διασυνδεδεμένο κορμό με μεγάλη διάμετρο (11).
H	Doubly Stochastic	Πολύ αραιό δίκτυο με σχετικά υψηλό μέσο όρο clustering (0.45) σε σχέση με τις άλλες αραιές μεθόδους. Εστιάζει στη διατήρηση της συνολικής συνδεσιμότητας με κάποιες τοπικές συσσωματώσεις.

7. Συζήτηση - Συμπεράσματα

Η παρούσα εργασία πραγματοποίησε μια συστηματική ανάλυση και εφαρμογή μεθόδων εξαγωγής κορμού σε διμερή δίκτυα γονιδίων–ασθενειών, με στόχο τη μείωση της πολυπλοκότητάς τους και την ανάδειξη των σημαντικότερων δομικών και λειτουργικών χαρακτηριστικών. Αξιοποιώντας δεδομένα από τις βάσεις GWAS και OMIM, κατασκευάστηκαν και αναλύθηκαν δίκτυα που αντιπροσωπεύουν τόσο τις πολυπαραγοντικές όσο και τις μονογονιδιακές πτυχές των ανθρώπινων νόσων, με στόχο την αξιολόγηση πολλαπλών μεθοδολογιών εξαγωγής κορμού (backbone extraction).

Ένα από τα σημαντικότερα ευρήματα που προέκυψαν από τη συγκριτική ανάλυση είναι η σαφής λειτουργική και τοπολογική διχοτόμηση μεταξύ των διαφορετικών κατηγοριών αλγορίθμων. Συγκεκριμένα, οι δομικές μέθοδοι, με προεξάρχον το Modularity Filter, παρήγαγαν δίκτυα που διατηρούν υψηλή πυκνότητα και ισχυρή δομή κοινοτήτων. Στο δίκτυο των ασθενειών, η προσέγγιση αυτή διατήρησε τον υψηλότερο μέσο βαθμό και συντελεστή συσσωμάτωσης, γεγονός που υποδεικνύει ότι αποτελεί την πλέον κατάλληλη επιλογή όταν το ερευνητικό ζητούμενο είναι ο εντοπισμός φαινοτυπικών ομάδων ή συν-νοσηροτήτων (comorbidities) με κοινό γενετικό υπόβαθρο, καθώς διαφυλάσσει τις ισχυρές τοπικές συσχετίσεις χωρίς να κατακερματίζει τη λειτουργική συνοχή του δικτύου.

Στον αντίποδα, οι στατιστικές μέθοδοι και οι τεχνικές ελάχιστου δέντρου, όπως το Polya Filter και το Maximum Spanning Tree (MST), λειτούργησαν πολύ πιο επιθετικά ως προς την αφαίρεση ακμών. Τα δίκτυα που προέκυψαν ήταν εξαιρετικά αραιά, συχνά με δενδρική μορφή και μικρή διάμετρο, αναδεικνύοντας μόνο τον απολύτως απαραίτητο «σκελετό» για τη διατήρηση της συνδεσιμότητας. Αυτή η προσέγγιση αποδεικνύεται κρίσιμη όταν στόχος της μελέτης είναι η αποκάλυψη ιεραρχικών σχέσεων ή η χαρτογράφηση των κυρίαρχων «λεωφόρων» ροής της βιολογικής πληροφορίας, καθώς επιτυγχάνει την πλήρη σχεδόν εξάλειψη του θορύβου που προκαλούν οι ασθενείς ή τυχαίες αλληλεπιδράσεις. Αξίζει επίσης να σημειωθεί η ιδιαίτερη συμπεριφορά του Stochastic Degree Sequence Model (SDSM), το οποίο, αν και στατιστικά αυστηρό, οδήγησε σε δίκτυα με πολύ μεγάλη διάμετρο, υποδηλώνοντας μια πιθανή επιμήκυνση των μονοπατιών επικοινωνίας και, κατ' επέκταση, μειωμένη αποτελεσματικότητα στη μεταφορά πληροφορίας συγκριτικά με άλλες μεθόδους.Σε επίπεδο βιολογικής ερμηνείας, οι εξαγόμενοι κορμοί αναδεικνύουν σημαντικά μοτίβα. Στο δίκτυο γονιδίων, οι πυκνές συστάδες αντανακλούν λειτουργική συνεργασία και ενότητες που εμπλέκονται σε κοινά βιολογικά μονοπάτια, ενώ στο δίκτυο ασθενειών, οι διατηρούμενες ομάδες υποδηλώνουν κοινά γενετικά υπόβαθρα και πιθανώς κοινές παθοφυσιολογικές οδούς. Αυτά τα ευρήματα υποστηρίζουν την υπόθεση ότι πολυπαραγοντικές ασθένειες μοιράζονται δικτυωμένους μηχανισμούς και ότι η ανάλυση δικτύων μπορεί να συμβάλει στη στρωμάτωση ασθενειών και στον εντοπισμό νέων θεραπευτικών στόχων.

Τα αποτελέσματα αυτά επιβεβαιώνουν ότι δεν υφίσταται μία ενιαία βέλτιστη μέθοδος για την ανάλυση βιολογικών δικτύων, αλλά η επιλογή του κατάλληλου αλγορίθμου εξαρτάται άμεσα από τη φύση του εκάστοτε βιολογικού ερωτήματος. Όταν ζητούμενο είναι η ανακάλυψη λειτουργικών modules, όπως ομάδες γονιδίων που συνεργάζονται σε ένα μονοπάτι, οι μέθοδοι που μεγιστοποιούν τη συσσωμάτωση είναι προτιμητέες. Αντίθετα, για τον εντοπισμό νέων, κρίσιμων βιοδεικτών ή κόμβων-διανομέων (hubs) που συνδέουν φαινομενικά ασύνδετες παθολογίες, οι στατιστικές μέθοδοι που απομονώνουν τις μη τυχαίες συνδέσεις κρίνονται πιο αξιόπιστες.

Η έρευνα συμβάλλει τόσο στη θεωρία της ανάλυσης δικτύων, επιβεβαιώνοντας την αξία πολυμεθοδικής προσέγγισης, όσο και στην πρακτική της βιοϊατρικής, προσφέροντας εργαλεία για την επεξεργασία και ερμηνεία μεγάλων και πολύπλοκων συνόλων δεδομένων. Ωστόσο, παραμένουν περιορισμοί, κυρίως σχετικά με την ποιότητα και πληρότητα των δεδομένων, την απλούστευση των στατικών μοντέλων και την ανάγκη πειραματικής επικύρωσης. Για μελλοντική έρευνα, προτείνεται η ενσωμάτωση δυναμικών ή πολυ-omics δεδομένων, η βελτιστοποίηση της υπολογιστικής απόδοσης για κλίμακα, και πάνω απ' όλα, η κλινική ή πειραματική επικύρωση των προβλεπόμενων κεντρικών στοιχείων. Συνολικά, η εργασία επισημαίνει τη σημασία της εξαγωγής κορμού ως κρίσιμου βήματος για τη μετατροπή πολύπλοκων δικτύων σε ερμηνεύσιμα και βιολογικά σημαντικά μοντέλα, ανοίγοντας νέους δρόμους για εφαρμογές στην ακριβή ιατρική και τη συστημική βιολογία.

Επιπροσθέτως, ιδιαίτερη πρακτική αξία θα είχε η δημιουργία μιας διαδικτυακής πλατφόρμας που θα ενσωματώνει αυτές τις μεθόδους, επιτρέποντας στους ερευνητές να αναλύουν τα δεδομένα τους και να λαμβάνουν αυτόματα προτάσεις για την κατάλληλη μέθοδο βάσει τωντοπολογικών χαρακτηριστικών του δικτύου τους, ενδεχομένως και με την ενσωμάτωση πολυστρωματικών δεδομένων (multi-omics).

-

Βιβλιογραφία

1. Welter, D., MacArthur, J., Morales, J., Burdett, T., Hall, P., Junkins, H., Klemm, A., Flicek, P., Manolio, T., Hindorff, L., & Parkinson, H. (2013). The NHGRI GWAS Catalog, a curated resource of SNP-trait associations. *Nucleic Acids Research*, *42*(D1), D1001–D1006. https://doi.org/10.1093/nar/gkt1229

2. Amberger, J. S., Bocchini, C. A., Schiettecatte, F., Scott, A. F., & Hamosh, A. (2014). OMIM.org: Online Mendelian Inheritance in Man (OMIM®), an online catalog of human genes and genetic disorders. *Nucleic Acids Research*, *43*(D1), D789–D798. https://doi.org/10.1093/nar/gku1205

3. Kontou, P. I., Pavlopoulou, A., Dimou, N. L., Pavlopoulos, G. A., & Bagos, P. G. (2016). Network analysis of genes and their association with diseases. *Gene*, *590*(1), 68–78. https://doi.org/10.1016/j.gene.2016.05.044

4. Pavlopoulos, G. A., Kontou, P. I., Pavlopoulou, A., Bouyioukos, C., Markou, E., & Bagos, P. G. (2018). Bipartite graphs in systems biology and medicine: a survey of methods and applications. *GigaScience*, *7*(4), 1–31. https://doi.org/10.1093/gigascience/giy014

5. Banerjee, S., Jenamani, M., & Pratihar, D. K. (2017, July 4). *Properties of a projected network of a bipartite network*. arXiv.org. https://arxiv.org/abs/1707.00912

6. Zhang, J., & Luo, Y. (2017b, January 1). *Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network*. In Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017). https://doi.org/10.2991/msam-17.2017.68

7. Daugulis, P., (2011). A note on a generalization of eigenvector centrality for bipartite graphs and applications. *Networks*, *59*(2), 261–264. https://doi.org/10.1002/net.20442

8. Banerjee, S., Jenamani, M., & Pratihar, D. K. (2017). *Algorithms for projecting a bipartite network* (Vol. 30, pp. 1–3). https://doi.org/10.1109/ic3.2017.8284345

9. Stram, R., Reuss, P., & Althoff, K. (2017). Weighted one mode projection of a bipartite graph as a local similarity measure. In *Lecture notes in computer science* (pp. 375–389). https://doi.org/10.1007/978-3-319-61030-6_26

10. Bag, S., Kumar, S. K., & Tiwari, M. K. (2019). An efficient recommendation generation using relevant Jaccard similarity. *Information Sciences*, *483*, 53–64. https://doi.org/10.1016/j.ins.2019.01.023

11. Lahitani, A. R., Permanasari, A. E., & Setiawan, N. A. (2016). Cosine similarity to determine similarity measure: Study case in online essay assessment. *In 2016 International Seminar on Application for Technology of Information and Communication (ISemantic)*, 1–6. https://doi.org/10.1109/citsm.2016.7577578

12. Adamic, L. A., & Adar, E. (2003b). Friends and neighbors on the Web. *Social Networks*, *25*(3), 211–230. https://doi.org/10.1016/s0378-8733(03)00009-1

13. El-Hashash, E. F., & Shiekh, R. H. A. (2022). A comparison of the Pearson, Spearman rank and Kendall Tau correlation coefficients using quantitative variables. *Asian Journal of Probability and Statistics*, 36–48. https://doi.org/10.9734/ajpas/2022/v20i3425

14. Yanagawa, T., Fujii, Y., & Mastuoka, J. (1994). Generalized Mantel-Haenszel procedures for 2 × J tables. *Environmental Health Perspectives*, *102*(suppl 8), 57–60. https://doi.org/10.1289/ehp.94102s857

15. Young, J., Petri, G., Peixoto, T. P., Young, J., Petri, G., & Peixoto, T. P. (2021). Hypergraph reconstruction from network data. *Communications Physics*, *4*(1). https://doi.org/10.1038/s42005-021-00637-w

16. De Arruda, G. F., Tizzani, M., & Moreno, Y. (2021). Phase transitions and stability of dynamical processes on hypergraphs. *Zaguan (University of Zaragoza Repository)*. https://doi.org/10.1038/s42005-021-00525-3

17. Hmaida, S., Cherifi, H., & Hassouni, M. E. (2024). A multilevel backbone extraction framework. *Applied Network Science*, *9*(1). https://doi.org/10.1007/s41109-024-00645-z

18. Yassin, A., Haidar, A., Cherifi, H., Seba, H., & Togni, O. (2023). An evaluation tool for backbone extraction techniques in weighted complex networks. *Scientific Reports*, *13*(1), 17000. https://doi.org/10.1038/s41598-023-42076-3

19. Neal, Z. P., (2022). backbone: An R package to extract network backbones. *PLoS ONE*, *17*(5), e0269137. https://doi.org/10.1371/journal.pone.0269137

20. Jo, W. S., Park, J., Luhur, A., Kim, B. J., & Ahn, Y. (2020, February 17). *Extracting hierarchical backbones from bipartite networks*. arXiv.org. https://arxiv.org/abs/2002.07239

21. Saracco, F., Di Clemente, R., Gabrielli, A., Squartini, T., Saracco, F., Di Clemente, R., Gabrielli, A., & Squartini, T. (2015). Randomizing bipartite networks: the case of the World Trade Web. *Scientific Reports*, *5*(1), 10595. https://doi.org/10.1038/srep10595

22. Neal, Z. P., & Neal, J. W. (2024). Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC) for Backbone Extraction. In *Studies in computational intelligence* (pp. 127–136). https://doi.org/10.1007/978-3-031-53468-3_11

23. Neal, Z. P., Domagalski, R., & Sagan, B. (2021). Comparing alternatives to the fixed degree sequence model for extracting the backbone of bipartite projections. *Scientific Reports*, *11*(1), 23929. https://doi.org/10.1038/s41598-021-03238-3

24. Serrano, M. Á., Boguñá, M., & Vespignani, A. (2009). Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, *106*(16), 6483–6488. https://doi.org/10.1073/pnas.0808904106

25. Gemmetto, V., Cardillo, A., & Garlaschelli, D. (2017, June 1). *Irreducible network backbones: unbiased graph filtering via maximum entropy*. arXiv.org. https://arxiv.org/abs/1706.00230

26. Dianati, N. (2016). Unwinding the hairball graph: Pruning algorithms for weighted complex networks. *Physical Review. E*, *93*(1), 012304. https://doi.org/10.1103/physreve.93.012304

27. *Network Backboning with Noisy Data*. (2017, April 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7929996

28. Baltakys, K. (2023). Inference of monopartite networks from bipartite systems with different link types. *Scientific Reports*, *13*(1). https://doi.org/10.1038/s41598-023-27744-8

29. Foti, N. J., Hughes, J. M., & Rockmore, D. N. (2011). Nonparametric sparsification of complex multiscale networks. *PLoS ONE*, *6*(2), e16431. https://doi.org/10.1371/journal.pone.0016431

30. Van Nuffel, N., Derudder, B., & Witlox, F. (2009). E*even important connections are not always meaningful: on the use of a polarisation measure in a typology of european cities in air transport networks. Tijdschrift Voor Economische En Sociale Geografie*, *101*(3), 333–348. https://doi.org/10.1111/j.1467-9663.2009.00547.x

31. Rajeh, S., Savonnet, M., Leclercq, E., & Cherifi, H. (2022b, January 30). *Modularity-based backbone extraction in weighted complex networks*. arXiv.org. https://arxiv.org/abs/2201.12905

32. Camerini, P. (1978). The min-max spanning tree problem and some extensions. *Information Processing Letters*, *7*(1), 10–14. https://doi.org/10.1016/0020-0190(78)90030-3

33. Slater, P. B. (2009). A two-stage algorithm for extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, *106*(26). https://doi.org/10.1073/pnas.0904725106

34. Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821–7826. https://doi.org/10.1073/pnas.122653799

35. Marcaccioli, R., Livan, G., Marcaccioli, R., & Livan, G. (2019). A Pólya urn approach to information filtering in complex networks. *Nature Communications*, *10*(1), 745. https://doi.org/10.1038/s41467-019-08667-3

36. Stram, R., Reuss, P., & Althoff, K. (2017). Weighted one mode projection of a bipartite graph as a local similarity measure. In *Lecture notes in computer science* (pp. 375–389). https://doi.org/10.1007/978-3-319-61030-6_26

Παράρτημα

1. Κατασκευή Μονομερούς Προβολής Διμερών Δικτύων με την Μέθοδο Brute Force Search

```
"""
Dependencies
This script requires the following Python libraries:
-pandas : For efficient data manipulation, especially when handling large
datasets and reading files in chunks.
-psutil: For monitoring system resource usage, such as memory.
You can install these dependencies using `pip`:
-bash : pip install pandas psutil
"""

import pandas as pd
import time
import psutil
from collections import defaultdict
import time


input_file = "data_file.txt"
gene_output_file = "gene_to_all_diseases.tsv"
disease_output_file = "disease_to_all_genes.tsv"
gene_edges_file = "gene_edges_one_mode.tsv"
disease_edges_file = "disease_edges_one_mode.tsv"


CHUNK_SIZE = 50000

def timer(func):
    """Decorator to measure function execution time."""
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Function {func.__name__} finished in {end_time - start_time:.4f}
sec")
        return result
    return wrapper


start_time = time.time()
data_load_time = time.time()

disease_to_genes = defaultdict(set)
gene_to_diseases = defaultdict(set)

print("Reading file in chunks...")

try:
    total_rows = 0
    for chunk in pd.read_csv(input_file, sep='\t', header=0,
chunksize=CHUNK_SIZE):
        total_rows += len(chunk)
        for phenotype, gene in zip(chunk.iloc[:, 0], chunk.iloc[:, 1]):
            if pd.notna(phenotype) and pd.notna(gene):
                disease_to_genes[phenotype].add(gene)
                gene_to_diseases[gene].add(phenotype)
```

```
        mem_usage = psutil.virtual_memory().percent
        print(f"Processed {total_rows} rows... (Memory Usage: {mem_usage}%)")
except Exception as e:
    print(f"Error while reading file: {e}")

print(f"Data loaded in {time.time() - data_load_time:.2f} sec")


save_mappings_time = time.time()
print("Saving disease-gene and gene-disease mappings...")

with open(gene_output_file, "w", encoding="utf-8") as g_file:
    for gene, diseases in gene_to_diseases.items():
        g_file.write(f"{gene}\t{','.join(diseases)}\n")

with open(disease_output_file, "w", encoding="utf-8") as d_file:
    for disease, genes in disease_to_genes.items():
        d_file.write(f"{disease}\t{','.join(genes)}\n")

print(f"Mappings saved in {time.time() - save_mappings_time:.2f} sec")

gene_edges_time = time.time()
print("Generating gene-gene edges without the third column...")

genes = list(gene_to_diseases.keys())

with open(gene_edges_file, "w", encoding="utf-8") as edge_file:
    edge_file.write("Gene1\tGene2\n") # The header is now only two columns
    for i in range(len(genes)):
        g1 = genes[i]
        for j in range(i + 1, len(genes)):
            g2 = genes[j]
            # Check if there is at least one shared disease
            if gene_to_diseases[g1] & gene_to_diseases[g2]:
                edge_file.write(f"{g1}\t{g2}\n") # Writing only the two gene
names

print(f"Gene edges computed in {time.time() - gene_edges_time:.2f} sec")


disease_edges_time = time.time()
print("Generating disease-disease edges without the third column...")

diseases = list(disease_to_genes.keys())

with open(disease_edges_file, "w", encoding="utf-8") as edge_file:
    edge_file.write("Disease1\tDisease2\n") # The header is now only two columns
    for i in range(len(diseases)):
        d1 = diseases[i]
        for j in range(i + 1, len(diseases)):
            d2 = diseases[j]
            # Check if there is at least one shared gene
            if disease_to_genes[d1] & disease_to_genes[d2]:
                edge_file.write(f"{d1}\t{d2}\n") # Writing only the two disease
names

print(f"Disease edges computed in {time.time() - disease_edges_time:.2f} sec")


print(f"Total execution time: {time.time() - start_time:.2f} sec")
```

2. **Κατασκευή Μονομερούς Προβολής Διμερών Δικτύων με την Μέθοδο Πολλαπλασιασμού Πινάκων**

```
import numpy as np
from scipy import sparse
import argparse
from tqdm import tqdm
import gc
import time

def timer(func):
    """Decorator to measure function execution time."""
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Function {func.__name__} finished in {end_time - start_time:.4f}
sec")
        return result
    return wrapper


start_time = time.time()
data_load_time = time.time()

@timer
def parse_data_chunked(filename, chunk_size=100000):
    """Διαβάζει το αρχείο data.txt με chunking για μεγάλα αρχεία"""
    diseases = []
    genes = []

    with open(filename, 'r', encoding='utf-8') as file:
        chunk = []
        for i, line in enumerate(file):
            line = line.strip()
            if not line:
                continue
            chunk.append(line)

            if len(chunk) >= chunk_size:
                process_chunk(chunk, diseases, genes)
                chunk = []


        # Process remaining lines
        if chunk:
            process_chunk(chunk, diseases, genes)

    return diseases, genes

@timer
def process_chunk(chunk, diseases, genes):
    """Επεξεργάζεται ένα chunk δεδομένων με τη μορφή 'GENE<TAB>DISEASE'"""
    for line in chunk:
        parts = line.split('\t')
```

```
        if len(parts) >= 2:
            gene = parts[0].strip()
            disease = parts[1].strip()
            genes.append(gene)
            diseases.append(disease)


@timer
def create_sparse_matrix_optimized(diseases, genes):
    """Δημιουργεί sparse πίνακα με βελτιστοποιημένη διαχείριση μνήμης"""
    unique_diseases = sorted(list(set(diseases)))
    unique_genes = sorted(list(set(genes)))

    disease_to_idx = {disease: idx for idx, disease in
enumerate(unique_diseases)}
    gene_to_idx = {gene: idx for idx, gene in enumerate(unique_genes)}

    # Χρήση dok_matrix που είναι αποδοτικός για incremental construction
    sparse_matrix = sparse.dok_matrix((len(unique_diseases), len(unique_genes)),
dtype=np.int8)

    print("Δημιουργία sparse πίνακα...")
    for disease, gene in tqdm(zip(diseases, genes), total=len(diseases)):
        row_idx = disease_to_idx[disease]
        col_idx = gene_to_idx[gene]
        sparse_matrix[row_idx, col_idx] = 1

    # Μετατροπή σε CSR format για αποδοτικό πολλαπλασιασμό
    return sparse_matrix.tocsr(), unique_diseases, unique_genes

@timer
def calculate_projections_optimized(sparse_matrix, chunk_size=100000):
    """Υπολογίζει προβολές με chunking για μεγάλους πίνακες"""
    print("Υπολογισμός προβολής γονιδίων (G^T * G)...")


    gene_projection = sparse_matrix.T.dot(sparse_matrix)

    print("Υπολογισμός προβολής ασθενειών (G * G^T)...")
    # Βελτιστοποιημένος υπολογισμός G * G^T
    disease_projection = sparse_matrix.dot(sparse_matrix.T)

    return gene_projection, disease_projection

def write_edges_optimized(projection_matrix, labels, output_filename,
threshold=0):
    """Γράφει ακμές με βελτιστοποιημένη διαχείριση μνήμης"""
    coo_matrix = projection_matrix.tocoo()

    print(f"Γράφω {output_filename}...")
    with open(output_filename, 'w', encoding='utf-8') as f:
        f.write("source\ttarget\tweight\n")

        written_edges = 0
        zero_edges = []   # για να αποθηκεύσουμε τα περίεργα edges

        for row, col, weight in zip(coo_matrix.row, coo_matrix.col,
coo_matrix.data):
            if row < col:
                if weight > threshold:
                    f.write(f"{labels[row]}\t{labels[col]}\t{weight}\n")
```

```
                    written_edges += 1
                elif weight == 0:
                    zero_edges.append((labels[row], labels[col]))

                if written_edges % 100000 == 0 and written_edges > 0:
                    print(f"Έγραψα {written_edges} ακμές...")

    print(f"Ολοκληρώθηκε! Σύνολο ακμών: {written_edges}")

    # Αν υπάρχουν παράσιτα, τύπωσε τα πρώτα 20
    if zero_edges:
        print(f"Βρέθηκαν {len(zero_edges)} ακμές με weight=0:")
        for edge in zero_edges[:20]:
            print(edge)

@timer
def main():
    parser = argparse.ArgumentParser(description='Network analysis of genes and
diseases')
    parser.add_argument('--input', default='data_file.txt', help='Input file')
    parser.add_argument('--genes_output', default='genes_edges.txt', help='Genes
edges output')
    parser.add_argument('--diseases_output', default='diseases_edges.txt',
help='Diseases edges output')
    parser.add_argument('--chunk_size', type=int, default=10000, help='Chunk
size for processing')

    args = parser.parse_args()

    try:
        print("Διαβάζω τα δεδομένα (chunked)...")
        diseases, genes = parse_data_chunked(args.input, args.chunk_size)

        print(f"Βρέθηκαν {len(diseases)} εγγραφές")
        print(f"Μοναδικές ασθένειες: {len(set(diseases))}")
        print(f"Μοναδικά γονίδια: {len(set(genes))}")

        # Έλεγχος αν τα δεδομένα είναι πολύ μεγάλα
        if len(set(genes)) > 50000 or len(set(diseases)) > 50000:
            print("ΠΡΟΕΙΔΟΠΟΙΗΣΗ: Πολύ μεγάλος αριθμός μοναδικών οντοτήτων!")
            print("Μπορεί να χρειαστούν ειδικές τεχνικές για Big Data")

        print("Δημιουργώ sparse πίνακα...")
        sparse_matrix, unique_diseases, unique_genes =
create_sparse_matrix_optimized(diseases, genes)

        # Απελευθερώνουμε μνήμη
        del diseases, genes
        gc.collect()

        print("Υπολογίζω προβολές...")
        gene_projection, disease_projection =
calculate_projections_optimized(sparse_matrix)

        # Απελευθερώνουμε μνήμη
        del sparse_matrix
        gc.collect()

        print("Γράφω αρχεία ακμών...")
        write_edges_optimized(gene_projection, unique_genes, args.genes_output)
```

```
        # Απελευθερώνουμε μνήμη
        del gene_projection
        gc.collect()

        write_edges_optimized(disease_projection, unique_diseases,
args.diseases_output)

        print("Ολοκληρώθηκε!")

    except MemoryError:
        print("ΣΦΑΛΜΑ ΜΝΗΜΗΣ: Τα δεδομένα είναι πολύ μεγάλα!")
        print("Προσπαθήστε:")
        print("1. Χρήση server με περισσότερη RAM")
        print("2. Μείωση του μεγέθους chunk_size")
        print("3. Χρήση distributed computing (Spark, Dask)")

if __name__ == "__main__":
    main()
```

3. Κατασκευή Υπεργραφημάτων

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Circle
from matplotlib.collections import PatchCollection
import matplotlib.cm as cm

# --- Βήμα 1: Ανάγνωση δεδομένων και εύρεση κοινότητων ---
def read_hypergraph_data(file_path):
    """Διαβάζει το αρχείο και επιστρέφει τις μοναδικές υπερακμές"""
    G_components = nx.Graph()

    with open(file_path, 'r') as f:
        for line in f:
            parts = line.strip().split('\t')
            if len(parts) < 2:
                continue
            source = parts[0]
            targets = parts[1].split(',')

            for target in targets:
                G_components.add_edge(source, target)

    return list(nx.connected_components(G_components))


def print_largest_community(communities):
    """Εντοπίζει και εκτυπώνει τα γονίδια της μεγαλύτερης κοινότητας."""

    if not communities:
        print("\nΔεν βρέθηκαν κοινότητες για εκτύπωση.")
        return
```

```
    # Ταξινόμηση κοινοτήτων κατά μέγεθος (μεγαλύτερη πρώτη)
    communities_sorted = sorted(communities, key=len, reverse=True)

    # Η μεγαλύτερη κοινότητα είναι η πρώτη
    largest_community = communities_sorted[0]
    size = len(largest_community)

    print("\n=====================================")
    print(f"ΜΕΓΙΣΤΗ ΚΟΙΝΟΤΗΤΑ (Κοινότητα 0) - {size} ΓΟΝΙΔΙΑ")
    print("=====================================")

    # Μετατροπή σε λίστα και ταξινόμηση για ευκολία ανάγνωσης
    genes_list = sorted(list(largest_community))

    # Εκτύπωση των γονιδίων (μπορεί να εκτυπωθεί και σε γραμμές)
    print(', '.join(genes_list))
    print(f"\nΣυνολικό πλήθος γονιδίων: {size}")

# --- Βήμα 2: Δημιουργία γράφου και layout βάσει κοινότητων ---
def create_community_layout(hyperedges, max_communities=15):
    """Δημιουργεί layout όπου κάθε κοινότητα είναι σε κύκλο"""

    # Επιλογή των μεγαλύτερων κοινότητων
    communities_sorted = sorted(hyperedges, key=len, reverse=True)
[:max_communities]

    # Δημιουργία γράφου
    G = nx.Graph()
    community_colors = {}
    community_centers = {}

    # Χρώματα για κάθε κοινότητα
    colors = cm.tab20(np.linspace(0, 1, len(communities_sorted)))

    # Θέσεις κέντρων για κάθε κοινότητα σε κύκλο
    radius = 5
    angles = np.linspace(0, 2*np.pi, len(communities_sorted), endpoint=False)

    for i, (community, color) in enumerate(zip(communities_sorted, colors)):
        community_id = i
        community_colors[community_id] = color

        # Κέντρο της κοινότητας σε κύκλο
        center_x = radius * np.cos(angles[i])
        center_y = radius * np.sin(angles[i])
        community_centers[community_id] = (center_x, center_y)

        # Προσθήκη γονιδίων και ακμών
        for gene in community:
            G.add_node(gene, community=community_id)

        # Προσθήκη ακμών μεταξύ όλων των γονιδίων της κοινότητας
        genes_list = list(community)
        for j in range(len(genes_list)):
            for k in range(j+1, len(genes_list)):
                G.add_edge(genes_list[j], genes_list[k], weight=2.0)

    # Υπολογισμός θέσεων για κάθε κόμβο
    pos = {}
    for community_id, center in community_centers.items():
```

```
        community_genes = [node for node, attr in G.nodes(data=True)
                           if attr['community'] == community_id]

        # Τυχαία θέση γύρω από το κέντρο της κοινότητας
        for gene in community_genes:
            angle = np.random.uniform(0, 2*np.pi)
            distance = np.random.uniform(0, 1.5)  # Ακτίνα διασποράς
            pos[gene] = (
                center[0] + distance * np.cos(angle),
                center[1] + distance * np.sin(angle)
            )

    return G, pos, community_colors, community_centers, communities_sorted

# --- Βήμα 3: Οπτικοποίηση ---
def visualize_communities_circular(G, pos, community_colors, community_centers,
communities):
    """Οπτικοποίηση με κοινότητες σε κυκλική διάταξη"""

    plt.figure(figsize=(15, 15))
    ax = plt.gca()

    # Ζωγραφική κύκλων για κάθε κοινότητα
    patches = []
    for community_id, center in community_centers.items():
        circle = Circle(center, 2.0, alpha=0.2,
                        color=community_colors[community_id])
        patches.append(circle)

        # Προσθήκη label για την κοινότητα
        plt.text(center[0], center[1] + 2.2, f'K{community_id}',
                ha='center', va='center', fontsize=10, fontweight='bold',
                bbox=dict(boxstyle="round,pad=0.3",
facecolor=community_colors[community_id], alpha=0.7))

    # Προσθήκη των κύκλων στο plot
    p = PatchCollection(patches, match_original=True)
    ax.add_collection(p)

    # Ζωγραφική των κόμβων με χρώμα ανά κοινότητα
    node_colors = [community_colors[G.nodes[node]['community']]
                   for node in G.nodes()]

    nx.draw_networkx_nodes(G, pos, node_color=node_colors,
                           node_size=100, alpha=0.8, ax=ax)

    # Ζωγραφική μόνο των πιο σημαντικών ακμών (εσωτερικές της κοινότητας)
    nx.draw_networkx_edges(G, pos, alpha=0.2, edge_color='gray', ax=ax)

    # Ζωγραφική labels μόνο για μερικά γονίδια (για να μην μπερδευτεί)
    labels = {}
    for node in G.nodes():
        if np.random.random() < 0.3:  # 30% πιθανότητα να εμφανιστεί label
            labels[node] = node

    nx.draw_networkx_labels(G, pos, labels, font_size=6, ax=ax)

    plt.title('Οπτικοποίηση Κοινότητων Γονιδίων σε Κυκλική Διάταξη\n(Κάθε χρώμα
= Διαφορετική Κοινότητα)', fontsize=14)
    plt.axis('equal')
    plt.axis('off')
```

```
    plt.tight_layout()

    # Legend με πληροφορίες κοινότητων
    plt.figtext(0.02, 0.02, "Πληροφορίες Κοινότητων:", fontweight='bold')
    y_pos = 0.01
    for i, community in enumerate(communities[:10]):  # Πρώτες 10 κοινότητες
        if len(community) > 1:
            info = f"K{i}: {len(community)} γονίδια"
            if len(community) <= 8:
                info += f" ({', '.join(sorted(community))})"
            plt.figtext(0.02, y_pos, info, fontsize=8)
            y_pos -= 0.015

    return ax

# --- Βήμα 4: Εκτενέστερη ανάλυση ---
def print_community_stats(communities):
    """Εκτύπωση στατιστικών για τις κοινότητες"""
    print(f"\n=== ΣΤΑΤΙΣΤΙΚΑ ΚΟΙΝΟΤΗΤΩΝ ===")
    print(f"Συνολικές κοινότητες: {len(communities)}")
    print(f"Συνολικά γονίδια: {sum(len(he) for he in communities)}")

    sizes = [len(he) for he in communities]
    print(f"\nΚατανομή μεγέθους:")
    print(f"  - Μέγιστο: {max(sizes)} γονίδια")
    print(f"  - Ελάχιστο: {min(sizes)} γονίδια")
    print(f"  - Μέσος όρος: {np.mean(sizes):.2f} γονίδια")

    print(f"\n10 ΜΕΓΑΛΥΤΕΡΕΣ ΚΟΙΝΟΤΗΤΕΣ:")
    communities_sorted = sorted(communities, key=len, reverse=True)
    for i, community in enumerate(communities_sorted[:10]):
        print(f"{i+1}. Κοινότητα {i}: {len(community)} γονίδια")
        if len(community) <= 12:
            print(f"   Γονίδια: {', '.join(sorted(community))}")
        print()

# --- Βήμα 5: Κυρίως πρόγραμμα ---
if __name__ == "__main__":
    file_path = "hypergraph_disease_edges.txt"

    print("Διαβάζω αρχείο...")
    hyperedges = read_hypergraph_data(file_path)

    print(f"Βρέθηκαν {len(hyperedges)} κοινότητες")
    print(f"Συνολικά γονίδια: {sum(len(he) for he in hyperedges)}")

    # Εκτύπωση στατιστικών
    print_community_stats(hyperedges)

    # Δημιουργία οπτικοποίησης για τις 15 μεγαλύτερες κοινότητες
    print("Δημιουργία οπτικοποίησης...")
    G, pos, community_colors, community_centers, selected_communities =
create_community_layout(
        hyperedges, max_communities=15
    )
    print_largest_community(hyperedges)
    # Οπτικοποίηση
    visualize_communities_circular(G, pos, community_colors, community_centers,
selected_communities)

    # Αποθήκευση
```

```
    plt.savefig('circular_communities.png', dpi=300, bbox_inches='tight')
    print("Οπτικοποίηση ολοκληρώθηκε! Αποθηκεύτηκε ως
'circular_communities.png'")

    plt.show()
```

4. Κατασκευή της Μεθόδου Disparity Filter

```
import pandas as pd
from scipy.stats import beta
import warnings
import os
from tqdm import tqdm
import psutil

warnings.filterwarnings("ignore")

class DisparityFilter:
    def __init__(self, alpha=0.05):
        self.alpha = alpha
        self.beta_cache = {}


    # p: normalized weight, k: degree
    def get_beta_sf(self, p, k):
        if k <= 1:
            return 1.0
        if (p, k) not in self.beta_cache:
            self.beta_cache[(p, k)] = beta.sf(p, 1, k - 1)
        return self.beta_cache[(p, k)]

    def process_chunk(self, chunk, strength, degree):
        results = []
        for _, row in chunk.iterrows():
            u, v, weight = row['Node1'], row['Node2'], row['Weight']

            try:
                # Skip nodes with zero strength
                if strength.get(u, 0) == 0 or strength.get(v, 0) == 0:
                    continue

                pij_u = weight / strength[u]
                alpha_ij_u = self.get_beta_sf(pij_u, degree[u])

                pij_v = weight / strength[v]
                alpha_ij_v = self.get_beta_sf(pij_v, degree[v])

                alpha_ij = min(alpha_ij_u, alpha_ij_v)

                if alpha_ij < self.alpha:
                    results.append((u, v, weight, alpha_ij))
            except Exception as e:
                continue
        return results

def calculate_network_stats(filename, chunksize=100000):
    print("Calculating network statistics...")

    strength = {}
    degree = {}

    # Count total lines for progress bar
```

```
    with open(filename) as f:
        total_rows = sum(1 for line in f) - 1  # Subtract header if exists

    chunk_reader = pd.read_csv(filename, sep="\t", header=None,
                               names=["Node1", "Node2", "Weight"],
                               chunksize=chunksize)

    for chunk in tqdm(chunk_reader, total=total_rows/chunksize, desc="Processing
network"):
        chunk['Weight'] = pd.to_numeric(chunk['Weight'], errors='coerce')
        chunk = chunk.dropna()

        for _, row in chunk.iterrows():
            u, v, weight = row['Node1'], row['Node2'], row['Weight']

            # Update strength
            strength[u] = strength.get(u, 0) + weight
            strength[v] = strength.get(v, 0) + weight

            # Update degree
            degree[u] = degree.get(u, 0) + 1
            degree[v] = degree.get(v, 0) + 1

    return strength, degree

def filter_large_file(filename, output_file, strength, degree,
chunksize=100000):
    print("Applying disparity filter...")

    disparity = DisparityFilter()

    with open(filename) as f:
        total_rows = sum(1 for line in f) - 1

    with open(output_file, 'w') as f_out:
        f_out.write("Source,Target,Weight,Alpha\n")

        chunk_reader = pd.read_csv(filename, sep="\t", header=None,
                                   names=["Node1", "Node2", "Weight"],
                                   chunksize=chunksize)

        for chunk in tqdm(chunk_reader, total=total_rows/chunksize,
desc="Filtering edges"):
            chunk['Weight'] = pd.to_numeric(chunk['Weight'], errors='coerce')
            chunk = chunk.dropna()

            results = disparity.process_chunk(chunk, strength, degree)

            for u, v, weight, alpha in results:
                f_out.write(f"{u},{v},{weight},{alpha:.6f}\n")

            # Memory check
            if psutil.virtual_memory().percent > 90:
                warnings.warn("High memory usage! Consider reducing chunksize.")

def main():
    print("="*50)
    print("LARGE NETWORK DISPARITY FILTER".center(50))
    print("="*50)

    filename = input("Enter input filename: ").strip()
```

```
    output_file = input("Enter output filename: ").strip()
    chunksize = int(input(f"Enter chunksize (recommended 100000) [100000]: ") or
100000)

    try:
        # Step 1: Calculate network statistics
        strength, degree = calculate_network_stats(filename, chunksize)

        print(f"\nFound {len(degree):,} nodes in the network")
        print(f"Total edges to process: {sum(degree.values())//2:,}")

        # Step 2: Filter and save results
        filter_large_file(filename, output_file, strength, degree, chunksize)

        print(f"\nResults saved to: {output_file}")
        print(f"Output file size: {os.path.getsize(output_file)/1024/1024:.2f}
MB")

    except Exception as e:
        print(f"\nERROR: {str(e)}")

if __name__ == "__main__":
    main()
```

5. Κατασκευή της Μεθόδου Doubly Stochastic Filter

```
import numpy as np
import pandas as pd
import networkx as nx
from scipy.sparse import coo_matrix, csr_matrix
from collections import defaultdict
from tqdm import tqdm

class TrueMultiscaleBackbone:
    def __init__(self, max_iter=200, tol=1e-6, smoothing_power=2,
                 significance_level=0.05, sparsity_adjustment=True,
                 early_stop_percentile=5, min_edges=None):
        """
        Parameters for flexible backbone extraction:
        - early_stop_percentile: Stop when edge weight < Xth percentile
        - min_edges: Minimum edges to keep before checking stopping criteria
        """
        self.max_iter = max_iter
        self.tol = tol
        self.smoothing_power = smoothing_power
        self.significance_level = significance_level
        self.sparsity_adjustment = sparsity_adjustment
        self.early_stop_percentile = early_stop_percentile
        self.min_edges = min_edges

    def _make_doubly_stochastic(self, matrix):
        """Exact doubly stochastic normalization"""
        matrix = matrix.tocsr()
        for _ in range(self.max_iter):
            # Row normalization
            row_sums = matrix.sum(axis=1).A.ravel()
```

```
            row_sums[row_sums == 0] = 1
            matrix = matrix.multiply(1 / row_sums[:, None])

            # Column normalization
            col_sums = matrix.sum(axis=0).A.ravel()
            col_sums[col_sums == 0] = 1
            matrix = matrix.multiply(1 / col_sums)

            if np.max(np.abs(1 - matrix.sum(axis=1).A.ravel())) < self.tol and \
               np.max(np.abs(1 - matrix.sum(axis=0).A.ravel())) < self.tol:
                break
        return matrix.power(self.smoothing_power)

    def _tarjan_strongly_connected(self, adj_list):
        """Tarjan's algorithm for strong connectivity check"""
        index = 0
        indices = {}
        lowlinks = {}
        on_stack = set()
        stack = []
        components = []

        def strongconnect(node):
            nonlocal index
            indices[node] = index
            lowlinks[node] = index
            index += 1
            stack.append(node)
            on_stack.add(node)

            for neighbor in adj_list.get(node, []):
                if neighbor not in indices:
                    strongconnect(neighbor)
                    lowlinks[node] = min(lowlinks[node], lowlinks[neighbor])
                elif neighbor in on_stack:
                    lowlinks[node] = min(lowlinks[node], indices[neighbor])

            if lowlinks[node] == indices[node]:
                component = []
                while True:
                    w = stack.pop()
                    on_stack.remove(w)
                    component.append(w)
                    if w == node:
                        break
                components.append(component)

        for node in adj_list:
            if node not in indices:
                strongconnect(node)

        return components

    def _build_backbone(self, matrix, node_names):
        """Complete hierarchical backbone construction"""
        coo = matrix.tocoo()
        edges = sorted(zip(coo.row, coo.col, coo.data), key=lambda x: -x[2])

        # Pre-calculate thresholds
        weights = [w for _,_,w in edges]
```

```
        percentile_threshold = np.percentile(weights,
self.early_stop_percentile)
        significance_threshold = self.significance_level if
self.significance_level else 0

        G = nx.DiGraph()
        G.add_nodes_from(node_names)
        adj_list = defaultdict(list)
        backbone_edges = []

        for i, j, w in tqdm(edges, desc="Building backbone"):
            # Apply significance filter
            if w < significance_threshold:
                continue

            G.add_edge(node_names[i], node_names[j], weight=w)
            adj_list[i].append(j)
            backbone_edges.append((i,j,w))

            # Check minimum edges requirement
            if self.min_edges and len(backbone_edges) < self.min_edges:
                continue

            # Check percentile stopping
            if w < percentile_threshold:
                print(f"Stopping: edge weight {w:.4f} <
{percentile_threshold:.4f} percentile")
                break

            # Check strong connectivity periodically
            if len(backbone_edges) % 1000 == 0:
                components = self._tarjan_strongly_connected(adj_list)
                if len(components) == 1:
                    print("Stopping: achieved strong connectivity")
                    break

        return G

    def extract_backbone(self, edge_list_path, output_path, mode='strict',
nrows=None): # Προσθέτουμε την παράμετρο nrows
        """
        Flexible extraction with modes:
        - 'strict': Original method (early stopping + significance)
        - 'balanced': Keep more edges but maintain structure
        - 'full': Keep maximum edges passing significance
        - nrows: Optional. Number of rows to read from the input file.
        """
        if mode == 'strict':
            self.early_stop_percentile = 20   # πιο αυστηρό
            self.significance_level = 0.05    # κόβει περισσότερες ακμές
            self.min_edges = None
        elif mode == 'balanced':
            self.early_stop_percentile = 1
            self.significance_level = 0.001
            self.min_edges = 100
        elif mode == 'full':
            self.early_stop_percentile = 0
            self.significance_level = 0
            self.min_edges = None

        print("Loading network...")
```

```
        try:
            # Χρησιμοποιούμε την παράμετρο nrows εδώ
            df = pd.read_csv(edge_list_path, sep="\t",
                             names=["source", "target", "weight"], skiprows=1,
nrows=nrows)

            print(f"  Αρχικές γραμμές που διαβάστηκαν: {len(df)}") # Διαγνωστική
εκτύπωση

            if df.empty:
                print(f" Προειδοποίηση: Το αρχείο '{edge_list_path}' είναι κενό
ή δεν μπόρεσε να αναλυθεί. Δεν φορτώθηκαν δεδομένα.")
                return nx.DiGraph() # Επιστρέφουμε άδειο γράφημα με χάρη

            df['weight'] = pd.to_numeric(df['weight'], errors='coerce')
            df = df.dropna()
            df = df.query("source != target")

            if df.empty:
                print(" Προειδοποίηση: Δεν βρέθηκαν έγκυρες ακμές μετά τον
καθαρισμό και το φιλτράρισμα. Επιστρέφεται ένα κενό γράφημα.")
                return nx.DiGraph()

        except FileNotFoundError:
            print(f"Σφάλμα: Το αρχείο δεν βρέθηκε στη διαδρομή
'{edge_list_path}'. Παρακαλώ ελέγξτε τη διαδρομή και το όνομα του αρχείου.")
            return nx.DiGraph()
        except Exception as e:
            print(f"Προέκυψε ένα απροσδόκητο σφάλμα κατά την ανάγνωση του
αρχείου: {e}")
            return nx.DiGraph()

        nodes = sorted(set(df['source']) | set(df['target']))
        node_to_idx = {n:i for i,n in enumerate(nodes)}

        if not nodes:
            print(" Προειδοποίηση: Δεν βρέθηκαν κόμβοι μετά την επεξεργασία του
αρχείου. Επιστρέφεται ένα κενό γράφημα.")
            return nx.DiGraph()

        sparse_matrix = coo_matrix(
            (df['weight'],
             ([node_to_idx[s] for s in df['source']],
              [node_to_idx[t] for t in df['target']])),
            shape=(len(nodes), len(nodes))
        ).tocsr()

        print(f"Δίκτυο: {len(nodes)} κόμβοι, {sparse_matrix.nnz} ακμές")

        if sparse_matrix.nnz == 0:
            print("Προειδοποίηση: Ο sparse matrix είναι κενός (καμία ακμή μετά
τη μετατροπή). Δεν είναι δυνατή η κανονικοποίηση.")
            return nx.DiGraph()

        print("Normalizing matrix...")
        normalized = self._make_doubly_stochastic(sparse_matrix)

        print("Extracting backbone...")
        backbone = self._build_backbone(normalized, nodes)

        print("\nResults:")
```

```
        print(f"Nodes: {backbone.number_of_nodes()}")
        print(f"Edges: {backbone.number_of_edges()}")
        print(f"Strongly connected: {nx.is_strongly_connected(backbone)}")

        nx.write_weighted_edgelist(backbone, output_path)
        print(f"Saved to {output_path}")

        return backbone

if __name__ == "__main__":
    extractor = TrueMultiscaleBackbone()

    # Εδώ αλλάζουμε το αρχείο εισόδου και προσθέτουμε την παράμετρο nrows=10000
    strict_backbone = extractor.extract_backbone(
        "weighted_gene_edges_one_mode.tsv", # Το νέο αρχείο εισόδου
        "strictgenes_ds_backbone_10k.edgelist", # Αλλάζουμε το όνομα αρχείου
εξόδου για να αντικατοπτρίζει τον περιορισμό
        mode='strict',
        nrows=200000 # Διαβάζουμε μόνο τις πρώτες 10.000 γραμμές
    )
    """
    # For balanced approach (more edges)
    balanced_backbone = extractor.extract_backbone(
        "weighted_gene_edges_one_mode.tsv",
        "balanced_ds_backbone.edgelist",
        mode='balanced'
    )

    # For maximum edges
    full_backbone = extractor.extract_backbone(
        "gene_edges_one_mode.tsv",
        "full_ds_backbone.edgelist",

        mode='full'
    )
    """
```

5. Κατασκευή της Μεθόδου Locally Adaptive Network Sparsification Filter

```
"""
Required Dependencies:
----------------------
- pandas
- scipy
- numpy
- psutil
- tqdm

Description:
------------
This script implements the LANS (Locally Adaptive Network Sparsification) method
for extracting the backbone of dense weighted networks.
The corrected version ensures proper edge processing and avoids duplicate edges
in the output.
"""
```

```
import pandas as pd
from scipy.sparse import csr_matrix, lil_matrix, coo_matrix, triu
import multiprocessing as mp
from functools import partial
import psutil
import os
import gc
import time
from tqdm import tqdm
import warnings
from typing import Dict, List, Tuple, Union
import resource
import numpy as np
import sys

warnings.filterwarnings('ignore')

# Settings
CHUNK_SIZE = 50000
ALPHA = 1.0 # More reasonable significance level
N_CORES = max(1, mp.cpu_count() - 1)
MAX_MEMORY = 16000  # Increased memory limit
TEST_MODE = True # Process entire file
TEST_ROWS = 10000

def increase_memory_limit():
    """Attempts to increase memory limits (Linux/Unix only)."""
    try:
        resource.setrlimit(resource.RLIMIT_AS, (resource.RLIM_INFINITY,
resource.RLIM_INFINITY))
    except Exception as e:
        print(f"Warning: Could not increase memory limit: {e}")

def memory_usage() -> float:
    """Returns current memory usage of the process in MB."""
    return psutil.Process(os.getpid()).memory_info().rss / (1024 ** 2)

def check_memory():
    """Raises an error if memory usage exceeds MAX_MEMORY."""
    if memory_usage() > MAX_MEMORY:
        raise MemoryError(f"Memory limit exceeded {MAX_MEMORY}MB")

def sparse_row_normalize(matrix: csr_matrix, row_index: int) -> csr_matrix:
    """Normalizes a sparse matrix row so that the sum of its elements is 1."""
    row = matrix.getrow(row_index)
    row_sum = row.sum()
    return row / row_sum if row_sum > 0 else csr_matrix(row.shape)

def process_chunk(chunk: pd.DataFrame, gene_map: Dict[str, int]) -> coo_matrix:
    """Processes a chunk of data and returns a COO matrix without duplicate
edges."""
    row = chunk['Gene1'].map(gene_map).values
    col = chunk['Gene2'].map(gene_map).values
    data = pd.to_numeric(chunk['Shared_Diseases'],
errors='coerce').fillna(0).values
    data = np.clip(data, 0, None)

    # Ensure we don't create duplicate edges
    n_genes = len(gene_map)
    return coo_matrix((data, (row, col)), shape=(n_genes, n_genes))
```

```
def build_sparse_matrix(input_file: str) -> Tuple[csr_matrix, Dict[str, int]]:
    """Builds a sparse weight matrix from a TSV file."""

    print("\nPhase 1: Discovering genes...")
    genes = set()

    # First pass: discover all unique genes
    for chunk in tqdm(pd.read_csv(input_file, sep='\t', header=None,
                    names=['Gene1', 'Gene2', 'Shared_Diseases'],
                    chunksize=CHUNK_SIZE),
                    desc="Scanning file"):
        genes.update(chunk['Gene1'].unique())
        genes.update(chunk['Gene2'].unique())
        check_memory()

        if TEST_MODE:
            if len(genes) >= TEST_ROWS:
                break

    gene_map = {gene: idx for idx, gene in enumerate(genes)}
    n_genes = len(gene_map)
    print(f"Found {n_genes} unique genes | Memory: {memory_usage():.2f}MB")

    print("\nPhase 2: Building matrix...")
    matrix = lil_matrix((n_genes, n_genes), dtype='float32')
    rows_processed = 0

    # Second pass: build the matrix
    for chunk in tqdm(pd.read_csv(input_file, sep='\t', header=None,
                                names=['Gene1', 'Gene2', 'Shared_Diseases'],
                                chunksize=CHUNK_SIZE),
                    desc="Loading data"):
        chunk_matrix = process_chunk(chunk, gene_map)
        # Add only upper triangular to avoid duplicates
        matrix += triu(chunk_matrix, format='lil')
        gc.collect()

        if memory_usage() > MAX_MEMORY * 0.7:
            matrix = matrix.tocsr()
            gc.collect()

        if TEST_MODE:
            rows_processed += len(chunk)
            if rows_processed >= TEST_ROWS:
                break

    # Make matrix symmetric by adding its transpose
    matrix = matrix + matrix.T
    return matrix.tocsr(), gene_map

def process_node(args):
    """Worker function to normalize one node row of the matrix."""
    matrix, i = args
    row = sparse_row_normalize(matrix, i)
    return (i, row) if row.nnz > 0 else None

def calculate_fractional_weights(matrix: csr_matrix) -> csr_matrix:
    """Calculates the normalized (fractional) edge weights for each node."""
    print("\nCalculating fractional weights...")
    n_nodes = matrix.shape[0]
    fractional = lil_matrix(matrix.shape, dtype='float32')
```

```
    with mp.Pool(processes=N_CORES) as pool:
        tasks = [(matrix, i) for i in range(n_nodes)]
        results = list(tqdm(pool.imap(process_node, tasks, chunksize=100),
                      total=n_nodes, desc="Parallel processing"))

    for result in results:
        if result:
            i, row = result
            fractional[i] = row

    return fractional.tocsr()

def extract_backbone_edges(args: Tuple[csr_matrix, csr_matrix, List[int],
float]) -> List[Tuple[int, int, float]]:
    """Extracts statistically significant edges for a subset of nodes."""
    matrix, fractional, nodes, alpha = args
    edges = []

    for i in nodes:
        row = fractional.getrow(i)
        if row.nnz == 0:
            continue

        weights = sorted(row.data, reverse=True)
        n_neighbors = len(weights)

        for j, p_ij in zip(row.indices, row.data):
            # Only process upper triangular to avoid duplicates
            if i < j:
                prob = sum(1 for w in weights if w > p_ij) / n_neighbors
                if prob < alpha:
                    edges.append((i, j, matrix[i, j]))

    return edges

def extract_backbone(matrix: csr_matrix, fractional: csr_matrix, alpha: float) -
> List[Tuple[int, int, float]]:
    print("\nExtracting network backbone...")
    n_nodes = matrix.shape[0]
    edges = []

    step = max(1, n_nodes // (N_CORES * 4) + 1)
    chunks = [(matrix, fractional, list(range(i, min(i + step, n_nodes))),
alpha)
              for i in range(0, n_nodes, step)]

    with mp.Pool(processes=N_CORES) as pool:
        for result in tqdm(pool.imap_unordered(extract_backbone_edges, chunks),
                           total=len(chunks), desc="Processing nodes"):
            edges.extend(result)
            check_memory()

    return edges

def save_results(edges: List[Tuple[int, int, float]],
                 gene_map: Dict[str, int],
                 fractional: csr_matrix,
                 output_file: str = "backbone_results.tsv",
                 gephi_file: str = "backbone_gephi.csv"):
    print("\nSaving results...")
```

```
    reverse_map = {idx: gene for gene, idx in gene_map.items()}

    # Save detailed TSV
    with open(output_file, 'w') as f:
        f.write("Gene1\tGene2\tWeight\tNormalized_Weight\n")
        for i, j, w in tqdm(edges, desc="Saving TSV"):
            f.write(f"{reverse_map[i]}\t{reverse_map[j]}\t{w}\
t{fractional[i,j]:.6f}\n")

    # Save Gephi format
    pd.DataFrame([(reverse_map[i], reverse_map[j], w) for i,j,w in edges],
                 columns=['Source', 'Target', 'Weight']).to_csv(gephi_file,
index=False)

    print(f"Results saved to:\n- {output_file}\n- {gephi_file}")
    print(f"Total edges exported: {len(edges)}")

def main(input_file: str):
    start_time = time.time()
    increase_memory_limit()

    try:
        # Verify input file
        print(f"\nProcessing file: {input_file}")
        df_test = pd.read_csv(input_file, sep='\t', header=None, nrows=5)
        print("Input file preview:")
        print(df_test.head())

        if TEST_MODE:
            print(f"\nTEST MODE - Processing first {TEST_ROWS} rows")

        matrix, gene_map = build_sparse_matrix(input_file)
        print(f"Matrix built ({matrix.shape[0]} nodes, {matrix.nnz} edges) |
Memory: {memory_usage():.2f}MB")

        fractional = calculate_fractional_weights(matrix)
        print(f"Fractional weights calculated | Memory: {memory_usage():.2f}MB")

        backbone_edges = extract_backbone(matrix, fractional, ALPHA)
        print(f"Found {len(backbone_edges)} significant edges (α={ALPHA})")

        save_results(backbone_edges, gene_map, fractional)

    except MemoryError as e:
        print(f"Error: {e}\nAdjust CHUNK_SIZE or MAX_MEMORY parameters")
    except Exception as e:
        print(f"Critical error: {str(e)}", file=sys.stderr)
        raise
    finally:
        print(f"\nTotal time: {(time.time()-start_time)/60:.2f} minutes")
        print(f"Final memory: {memory_usage():.2f} MB")

if __name__ == '__main__':
    mp.freeze_support()
    input_file = "weighted_disease_edges_one_mode.tsv"
    main(input_file)
```

5. Κατασκευή της Μεθόδου Marginall Likelihood Filter

```
"""
This code implements the Marginal Likelihood Filter (MLF) for pruning edges in
weighted complex networks,
as described in the article:
"Unwinding the hairball graph: Pruning algorithms for weighted complex networks"
by Navid Dianati (PHYSICAL REVIEW E 93, 012304 (2016)).

Purpose:
The primary goal is to identify the statistically most significant edges in a
weighted network
(where edge weights are integers and count the occurrences of an "event"
relating the nodes)
and extract a subgraph consisting of those edges. This helps in "unwinding"
"hairball" graphs,
removing "noise" and revealing the most important structures.

How it works:
1.  **Null Model:** It assumes that unit edges are randomly assigned between
nodes
    with probabilities proportional to the product of their weighted degrees
(strengths). In other words,
    nodes with higher degrees are more likely to be connected to one another
purely by chance.
2.  **p-value Calculation:** For each edge, a p-value is calculated using the
binomial distribution.
    This p-value expresses the probability of observing the specific edge weight
(or a larger one)
    by chance, given the weighted degrees of the two connected nodes and the
total weight of the network.
    A low p-value indicates that the edge is statistically significant and less
likely to have occurred randomly.
3.  **Filtering:** All edges whose p-value is higher than a predefined threshold
`alpha` (e.g., 0.05) are
    removed. The remaining edges are considered the most significant.
4.  **Chunk Processing:** For efficient handling of large datasets, the code
reads and processes data
    in chunks, thus avoiding memory overload.
5.  **Parallel Computation:** The p-value calculation is performed in parallel
using all available CPU cores,
    significantly speeding up the process for large networks.
6.  **Sparse Matrix Output:** The filtered graph is optionally saved as a sparse
matrix for memory efficiency.
7.  **Test Mode:** Includes a test mode to process only a subset of the data for
quick debugging and testing.
"""


from scipy.stats import binom, poisson
import pandas as pd
import numpy as np
```

```
from scipy.stats import binom
from scipy.sparse import dok_matrix, save_npz
from multiprocessing import Pool, cpu_count
import warnings
import os
from math import ceil

# Global settings
warnings.filterwarnings("ignore")
os.environ["PYTHONHASHSEED"] = "0"

class MLFProcessor:
    def __init__(self, alpha=0.05, chunk_size=50000, test_mode=False,
test_size=10000):
        """
        Initializes the MLFProcessor with parameters for significance, chunking,
and testing.

        Args:
            alpha (float): The significance threshold for filtering edges (p-
value < alpha).
            chunk_size (int): The number of rows (edges) to process at once from
the input file.
            test_mode (bool): If True, only processes a limited number of edges
defined by test_size.
            test_size (int): The maximum number of edges to process when
test_mode is True.
        """

        self.alpha = alpha
        self.chunk_size = chunk_size
        self.test_mode = test_mode
        self.test_size = test_size

    def load_data(self, file_path):
        """Load data with test mode support"""
        if self.test_mode:
            print(f"TEST MODE: Loading first {self.test_size} rows...")
            return pd.read_csv(file_path, sep="\t", nrows=self.test_size,
chunksize=self.chunk_size)
        else:
            print("Loading full dataset in chunks...")
            return pd.read_csv(file_path, sep="\t", chunksize=self.chunk_size)

    @staticmethod
    def calculate_p_value(args):
        """
        Calculates the p-value using the Poisson approximation for large
networks.
        """
        w_ij, s_i, s_j, T = args

        # Handle edge cases
        if T <= 0 or s_i <= 0 or s_j <= 0 or w_ij <= 0:
            return 1.0

        # Calculate probability according to MLF formula
        p_ij = (s_i * s_j) / (2 * T ** 2)

        # Calculate the mean (lambda) for the Poisson approximation
        lambda_val = T * p_ij
```

```
        # Use Poisson approximation for large T or small p
        # Poisson is a good approximation of Binomial when T is large and p is
small
        # A common rule of thumb is T*p < 10, but it works well even for larger
values.
        # Given your large T, this is the most robust approach.
        try:
            # Use survival function for P(X >= w_ij)
            p_val = poisson.sf(w_ij - 1, lambda_val)
            return p_val
        except Exception as e:
            print(f"Error in p-value calculation: {e}, args: {args}")
            return np.nan

    def process_graph(self, file_path, output_file):
        """
        Main method to process the graph
        """

        # First, check the column names in the file
        print("Reading file header to detect column names...")
        with open(file_path, 'r') as f:
            header = f.readline().strip().split('\t')
        print(f"Columns detected: {header}")

        # Assume first two columns are nodes, third is weight
        node1_col, node2_col, weight_col = header[0], header[1], header[2]
        print(f"Using columns: {node1_col}, {node2_col}, {weight_col}")

        # FIRST PASS: Calculate strengths
        print("\n=== FIRST PASS ===")
        node_index = {}
        current_idx = 0
        strength = []
        total_weight = 0
        edge_count = 0

        for chunk in self.load_data(file_path):
            # Use the detected weight column name
            chunk[weight_col] = pd.to_numeric(chunk[weight_col],
errors="coerce").fillna(1).astype(int)

            for _, row in chunk.iterrows():
                # Use the detected node and weight column names
                g1 = row[node1_col]
                g2 = row[node2_col]
                w = row[weight_col]

                # Skip zero or negative weights
                if w <= 0:
                    continue

                # Update node index
                if g1 not in node_index:
                    node_index[g1] = current_idx
                    strength.append(0)
                    current_idx += 1
                if g2 not in node_index:
                    node_index[g2] = current_idx
                    strength.append(0)
```

```
                    current_idx += 1

                # Update strengths
                idx1 = node_index[g1]
                idx2 = node_index[g2]
                strength[idx1] += w
                strength[idx2] += w
                total_weight += w
                edge_count += 1

                # Early stop in test mode
                if self.test_mode and edge_count >= self.test_size:
                    break

            if self.test_mode and edge_count >= self.test_size:
                break

        # Total weight should be sum of all edge weights
        total_weight /= 2
        strength = np.array(strength, dtype=np.float64)
        print(f"Processed {len(node_index)} nodes and {edge_count} edges")
        print(f"Total weight: {total_weight:.2f}")

        # SECOND PASS: Filter edges
        print("\n=== SECOND PASS ===")
        sparse_graph = dok_matrix((len(node_index), len(node_index)),
dtype=np.float32)
        results = []
        processed_count = 0

        for chunk in self.load_data(file_path):
            # Use the detected weight column name
            chunk[weight_col] = pd.to_numeric(chunk[weight_col],
errors="coerce").fillna(1).astype(int)

            args = []
            valid_rows = []
            for _, row in chunk.iterrows():
                g1 = row[node1_col]
                g2 = row[node2_col]
                w = row[weight_col]

                # Skip zero or negative weights
                if w <= 0:
                    continue

                # Skip nodes not found in first pass
                if g1 not in node_index or g2 not in node_index:
                    continue

                s_i = strength[node_index[g1]]
                s_j = strength[node_index[g2]]

                args.append((w, s_i, s_j, total_weight))
                valid_rows.append(row)

                if self.test_mode and processed_count >= self.test_size:
                    break

                processed_count += 1
```

```
            # Parallel processing
            if args:
                with Pool(cpu_count()) as pool:
                    p_values = pool.map(self.calculate_p_value, args)

                # Build sparse matrix and results
                for row, p_val in zip(valid_rows, p_values):
                    g1 = row[node1_col]
                    g2 = row[node2_col]
                    w = row[weight_col]

                    if p_val < self.alpha:
                        i, j = node_index[g1], node_index[g2]
                        sparse_graph[i, j] = w
                        sparse_graph[j, i] = w  # Undirected
                        results.append((g1, g2, w, p_val))

                    # Debug output for first few edges
                    if len(results) < 10:
                        print(f"Edge: {g1}-{g2}, weight: {w}, strength_i:
{strength[node_index[g1]]:.2f}, strength_j: {strength[node_index[g2]]:.2f}, p-
value: {p_val:.6f}")

            if self.test_mode and processed_count >= self.test_size:
                break

        # Save results
        print("\n=== SAVING RESULTS ===")
        print(f"Found {len(results)} significant edges out of {processed_count}
processed")

        if len(results) > 0:
            if not self.test_mode:
                save_npz("filtered_graph_sparse.npz", sparse_graph.tocsr())
                print("Saved sparse matrix: filtered_graph_sparse.npz")

            df_results = pd.DataFrame(results, columns=[node1_col, node2_col,
"Weight", "p_value"])
            df_results.to_csv(output_file, sep="\t", index=False)
            print(f"Saved {len(results)} filtered edges to {output_file}")
        else:
            print("No significant edges found! Try adjusting the alpha
threshold.")

        print(f"Sparse matrix stats: {sparse_graph.shape} shape,
{sparse_graph.nnz} non-zero elements")

if __name__ == "__main__":
    # Configuration
    input_file = "weighted_disease_edges_one_mode.tsv"
    output_file = "mlf_diseases_filtered.tsv" if True else
"filtered_edges_FULL.tsv"

    # Initialize processor in test mode (first 10,000 edges)
    processor = MLFProcessor(
        alpha=0.05,
        chunk_size=5000,  # Smaller chunks for testing
        test_mode=False,   # Enable test mode
        test_size=1000000   # Process only first 10,000 edges
    )
```

```
    # Run processing
    processor.process_graph(
        file_path=input_file,
        output_file=output_file)
```

Όλα τα αρχεία προγραμματισμού, καθώς και οι μέθοδοι εξαγωγής κορμού δικτύων είναι αναρτημένα στον σύνδεσμο https://github.com/datalysis-cmd/PRED-BiPN/tree/main.