Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής στα ελληνικά	**Ανάλυση Δικτύων Αναφορών με Χρήση Αλγορίθμων Γένεσης Τεχνητών Δικτύων**
Τίτλος Διατριβής στα αγγλικά	**Citation Network Analysis With Artificial Network Generation Algorithms**
Ονοματεπώνυμο Φοιτητή	**Νικόλαος Αντώνιος Γραμματικός**
Πατρώνυμο	**Αναστάσιος**
Αριθμός Μητρώου	**ΜΠΚΕΔ21010**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2023**

Τριμελής Εξεταστική Επιτροπή

Δημήτριος Σωτηρόπουλος Επίκουρος Καθηγητής	Δημήτριος Αποστόλου Καθηγητής	Δρ. Γρηγόριος Κωρονάκος Διδάσκων ΠΜΣ

Περίληψη

Ένα κοινωνικό δίκτυο πραγματεύεται πληθώρα πληροφοριών σχετικά με κοινές ή μη ιδιότητες. Το co-authorship είναι μια από τις πιο απτές μορφές ερευνητικής συνεργασίας. Ένα δίκτυο συν-συγγραφέων είναι ένα κοινωνικό δίκτυο στο οποίο οι συγγραφείς μέσω της συμμετοχής τους σε μία ή περισσότερες δημοσιεύσεις μέσω μιας έμμεσης διαδρομής έχουν συνδεθεί μεταξύ τους. Η παρούσα έρευνα, χρησιμοποιώντας την ανάλυση κοινωνικού δικτύου, μελέτησε το δίκτυο συν-συγγραφέων 10300 άρθρων που δημοσιεύθηκαν στο paper με τίτλο «**A survey and analysis of the first 40 years of scholarly literature in DEA: 1978-2016»** όπου αναφέρονται μελέτες που σχετίζονται με την ανάλυση φακέλων δεδομένων (DEA).

Η μελέτη διεξήχθη με την χρήση της γλώσσας προγραμματισμού Python και χρησιμοποιώντας την ανάλυση δικτύου co-authorship των συγγραφέων. Η τοπολογία του δικτύου συν-συγγραφέων 10300 δημοσιεύσεων αναλύθηκε χρησιμοποιώντας δείκτες μετρήσεων μακρο-επιπέδου ανάλυσης δικτύου, όπως αρθρωτότητα, συντελεστής ομαδοποίησης, συνιστώσες και μέση απόσταση. Επιπλέον, προκειμένου να αξιολογηθεί η απόδοση κάθε συγγραφέα και δημοσίευσης στο δίκτυο, χρησιμοποιήθηκαν οι δείκτες μετρήσεων μικρο-επιπέδου, όπως ο βαθμός κεντρικότητας, η κατανομή κεντρικότητας εγγύτητας και η εκκεντρότητα μεταξύ των συγγραφέων. Το λογισμικό Gephi χρησιμοποιήθηκε για να σχεδιαστεί και να αναλυθεί το δίκτυο συν-συγγραφέων.

Η αξιολόγηση της παραγωγικότητας έδειξε πως οι περισσότερες δημοσιεύσεις απέρρεαν από τα πρώτα paper του 1978. Τα συγκεκριμένα έργα όπως φαίνεται βρίσκονται στο κέντρο του γράφου που έχει σχεδιαστεί καθώς είναι αρχική πηγή για τα περισσότερα paper του paper. Συμπερασματικά καταλαβαίνουμε πως οι παλαιότεροι συγγραφείς θεωρούνται ηγέτες του δικτύου των συν-συγγραφέων και όσο προχωρούμε σε νεότερα έργα, τόσο οι αναφορές σε αυτούς πληθαίνουν.

Εκτελώντας αλγόριθμους γέννησης δικτύων, δημιουργήθηκε ένα τεχνητό δίκτυο, με παρόμοια δομή και αποτελέσματα τόσο σε δείκτες μετρήσεων μακρο-επιπέδου ανάλυσης, τόσο και μικρο-επιπέδου για περαιτέρω ανάλυση ομοιοτήτων στις μετρήσεις.

Το δίκτυο των συν-συγγραφέων του αρχικού paper αποτελεί ένα μικρό κοινωνικό δίκτυο αποτελούμενο από τους ίδιους τους συγγραφείς, με κοινά χαρακτηριστικά και συνδέσεις που αναλύονται στην παρούσα μελέτη.

Λέξεις Κλειδιά : κοινωνικό δίκτυο, δίκτυο συν-συγγραφέων, Python, δείκτες μετρήσεων μακρο-επιπέδου, δείκτες μετρήσεων μικρο-επιπέδου, Gephi, αλγόριθμοι γένεσης

Abstract

A social network deals with a wealth of information on common or non-common activities. Co-authorship is one of the most tangible forms of research collaboration. A co-authors network is a social network in which the authors are connected to each other through their participation in one or more publications through an indirect route. The present study, using social network analysis, studied the network of co-authors of 10,300 articles published in the paper entitled **"A survey and analysis of the first 40 years of scholarly literature in DEA: 1978-2016"** where studies related to data envelopment analysis (DEA).

The study was conducted using the Python programming language and co-authorship network analysis. The topology of the co-author network of 10,300 publications was analyzed using indicators of macro-level network analysis metrics, such as modularity, clustering coefficient, components and mean distance. In addition, in order to evaluate the performance of each author and paper in the network, micro-level metrics were used, such as the degree of centrality, the closeness centrality distribution and the eccentricity between the authors. The Gephi software was used to design and analyze the co-author network.

The productivity rating showed that most of the publications came from the first papers of 1978. The specific works seem to be in the center of the graph that has been designed as it is the initial source for most of the papers. In conclusion, we understand that older writers are considered leaders in the network of co-authors, and as we move on to newer projects, the references to them increase.

Using artificial network generation algorithms, a technical network was created with a similar structure and results both in macro-level analysis metrics and micro-level analysis, for further investigation regarding the metrics.

The network of co-authors of the initial paper is a small social network consisting of the authors themselves, with common features and connections that are analyzed in the present study.

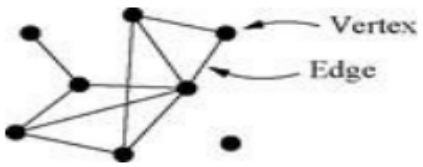
Keywords: social network analysis, co-authors network, Python, macro-level network analysis metrics, micro-level metrics, Gephi, generation algorithms

Περιεχόμενο

1. Εισαγωγή

1.1 Τι είναι το δίκτυο;

Ένα δίκτυο είναι μια συλλογή από αντικείμενα, που ονομάζουμε κόμβους, και τις διασυνδέσεις τους με άλλους κόμβους που ονομάζουμε ακμές. Ένα κοινωνικό δίκτυο είναι μια κοινωνική δομή που αποτελείται από ένα σύνολο κοινωνικών παραγόντων (όπως άτομα ή οργανώσεις), σύνολα δυαδικών δεσμών και άλλες κοινωνικές αλληλεπιδράσεις μεταξύ των κόμβων. Η προοπτική κοινωνικού δικτύου παρέχει ένα σύνολο μεθόδων για την ανάλυση της δομής ολόκληρων κοινωνικών οντοτήτων, καθώς και μια ποικιλία θεωριών που εξηγούν τα πρότυπα που παρατηρούνται σε αυτές τις δομές. Η μελέτη αυτών των δομών χρησιμοποιεί ανάλυση κοινωνικού δικτύου για τον εντοπισμό τοπικών και παγκόσμιων προτύπων, τον εντοπισμό σημαντικών οντοτήτων και την εξέταση της δυναμικής του δικτύου. Τα κοινωνικά δίκτυα και η ανάλυσή τους είναι ένα εγγενές διεπιστημονικό ακαδημαϊκό πεδίο που προέκυψε από την κοινωνική ψυχολογία, την κοινωνιολογία, τις στατιστικές και τη θεωρία γραφημάτων. Ο Georg Simmel έγραψε τις πρώιμες δομικές θεωρίες στην κοινωνιολογία, δίνοντας έμφαση στη δυναμική των τριάδων και του «ιστού των ομάδων». Ο Jacob Moreno πιστώνεται ότι ανέπτυξε τα πρώτα κοινωνιογραφήματα τη δεκαετία του 1930 για τη μελέτη διαπροσωπικών σχέσεων. Αυτές οι προσεγγίσεις τυποποιήθηκαν μαθηματικά στη δεκαετία του 1950 και οι θεωρίες και οι μέθοδοι των κοινωνικών δικτύων έγιναν διαδεδομένες στις κοινωνικές και συμπεριφορικές επιστήμες μέχρι τη δεκαετία του 1980. Η ανάλυση κοινωνικών δικτύων είναι πλέον ένα από τα σημαντικότερα πρότυπα στη σύγχρονη κοινωνιολογία και χρησιμοποιείται επίσης σε πολλές άλλες κοινωνικές και επίσημες επιστήμες[1]. Μαζί με άλλα πολύπλοκα δίκτυα, αποτελεί μέρος του δημιουργούμενου πεδίου της επιστήμης του δικτύου.

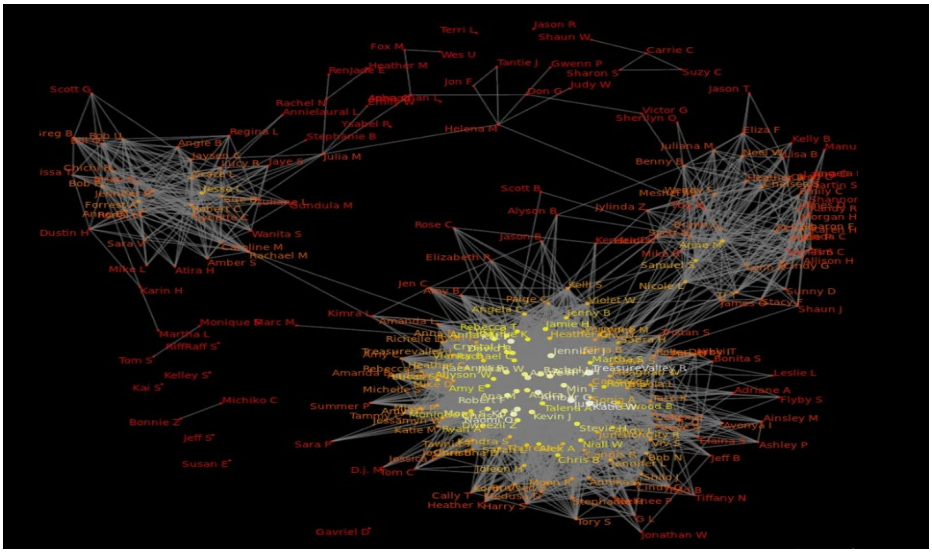


Εικόνα 1. Παράσταση κόμβου και ακμής

1.2 Τι είναι ένα κοινωνικό δίκτυο;

Το κοινωνικό δίκτυο είναι μια θεωρητική δομή χρήσιμη στις κοινωνικές επιστήμες για τη μελέτη σχέσεων μεταξύ ατόμων, ομάδων, οργανισμών ή ακόμη και ολόκληρων κοινωνιών (κοινωνικές μονάδες, δείτε διαφοροποίηση). Ο όρος χρησιμοποιείται για να περιγράψει μια κοινωνική δομή που καθορίζεται από τέτοιες αλληλεπιδράσεις. Οι δεσμοί μέσω των οποίων συνδέεται κάθε δεδομένη κοινωνική ενότητα αντιπροσωπεύουν τη σύγκλιση των διαφόρων κοινωνικών επαφών αυτής της μονάδας. Αυτή η θεωρητική προσέγγιση είναι, απαραίτητα, σχεσιακή. Ένα αξίωμα της προσέγγισης του κοινωνικού δικτύου για την κατανόηση της κοινωνικής αλληλεπίδρασης είναι ότι τα κοινωνικά φαινόμενα πρέπει να συλληφθούν και να διερευνηθούν κυρίως μέσω των ιδιοτήτων των σχέσεων μεταξύ και εντός των μονάδων, αντί των ιδιοτήτων αυτών των μονάδων. Έτσι, μια κοινή κριτική για τη θεωρία των κοινωνικών δικτύων είναι ότι η μεμονωμένη υπηρεσία συχνά αγνοείται, αν και αυτό δεν μπορεί να συμβαίνει στην πράξη (βλ. Μοντελοποίηση βάσει πρακτόρων). Ακριβώς επειδή πολλοί διαφορετικοί τύποι σχέσεων, μεμονωμένοι ή σε συνδυασμό, σχηματίζουν αυτές τις διαμορφώσεις δικτύου, τα αναλυτικά στοιχεία δικτύου είναι χρήσιμα για ένα ευρύ φάσμα ερευνητικών επιχειρήσεων[2].

Η ανάλυση κοινωνικού δικτύου (SNA) είναι η διερεύνηση των κοινωνικών δομών μέσω της χρήσης δικτύων και θεωρίας γραφημάτων. Χαρακτηρίζει δικτυωμένες δομές σε όρους κόμβων (μεμονωμένοι παράγοντες, άτομα ή πράγματα μέσα στο δίκτυο) και τους δεσμούς, τα άκρα ή τους συνδέσμους (σχέσεις ή αλληλεπιδράσεις) που τις συνδέουν. Παραδείγματα κοινωνικών δομών που εμφανίζονται συνήθως μέσω ανάλυσης κοινωνικών δικτύων περιλαμβάνουν δίκτυα κοινωνικών μέσων, memes spread, κυκλοφορία πληροφοριών, δίκτυα φιλίας και γνωριμιών, επιχειρηματικά δίκτυα, δίκτυα γνώσης, δύσκολες εργασιακές σχέσεις, κοινωνικά δίκτυα, γραφήματα συνεργασίας, συγγένεια, μετάδοση ασθενειών και σεξουαλικές σχέσεις. Αυτά τα δίκτυα συχνά απεικονίζονται μέσω κοινωνιογραμμάτων στα οποία οι κόμβοι αντιπροσωπεύονται ως σημεία και οι δεσμοί αντιπροσωπεύονται ως γραμμές. Αυτές οι απεικονίσεις παρέχουν ένα μέσο ποιοτικής αξιολόγησης των δικτύων μεταβάλλοντας την οπτική αναπαράσταση των κόμβων και των άκρων τους ώστε να αντικατοπτρίζουν χαρακτηριστικά ενδιαφέροντος. Η ανάλυση κοινωνικών δικτύων έχει αναδειχθεί ως βασική τεχνική στη σύγχρονη κοινωνιολογία. Έχει επίσης αποκτήσει σημαντική παρακολούθηση στην ανθρωπολογία, τη βιολογία, δημογραφία, μελέτες επικοινωνίας, οικονομία, γεωγραφία, ιστορία, επιστήμη της πληροφορίας, οργανωτικές μελέτες, πολιτική επιστήμη, δημόσια υγεία, κοινωνική ψυχολογία, μελέτες ανάπτυξης, κοινωνιογλωσσολογία και επιστήμη υπολογιστών και είναι πλέον συνήθως διαθέσιμη ως εργαλείο του καταναλωτή[4].

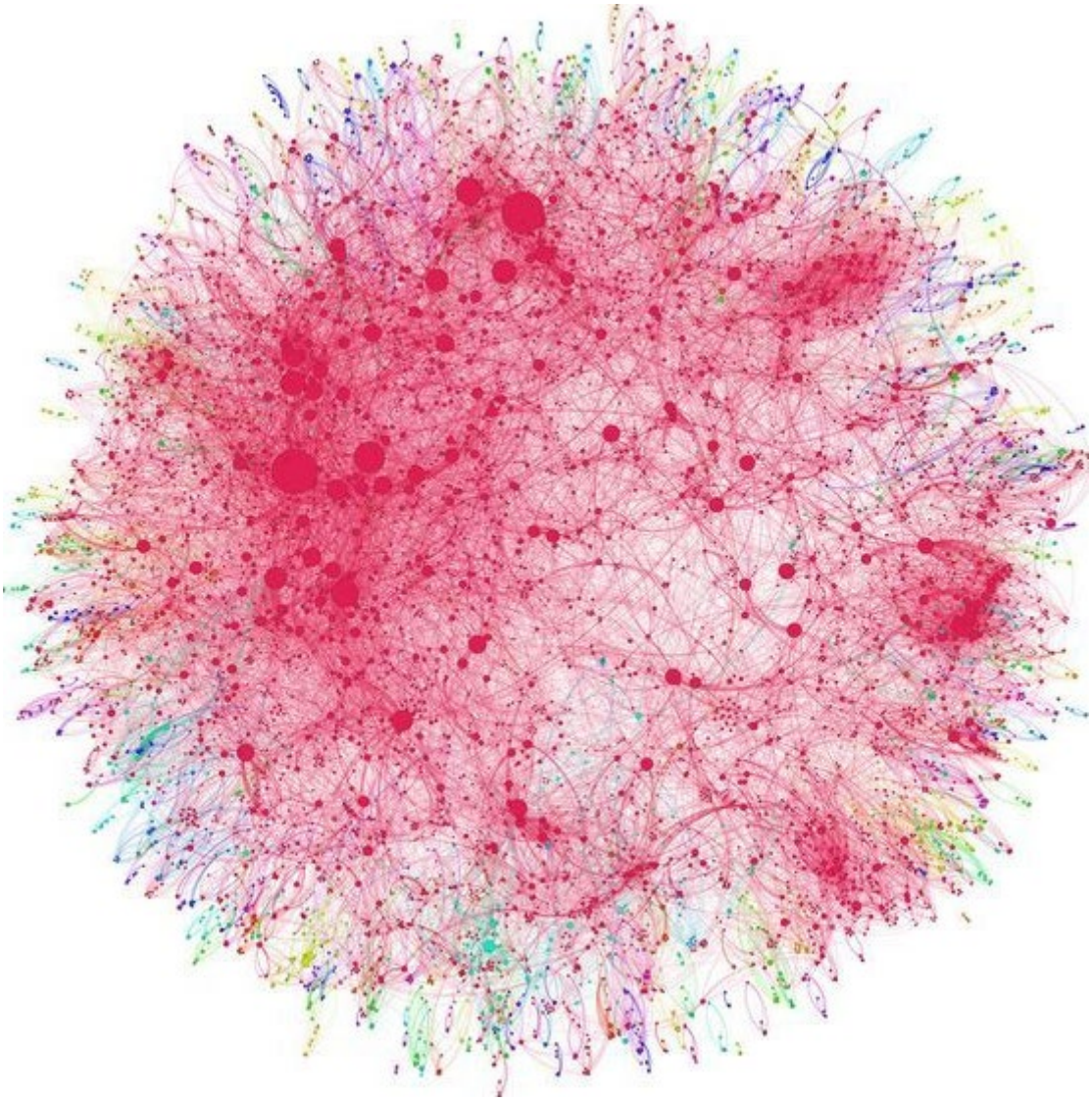


Εικόνα 2. Παράδειγμα κοινωνικού δικτύου

1.3 Co – Authorship Network

Σήμερα, η συζήτηση για τις ερευνητικές δραστηριότητες σχετίζεται με ορισμένα θέματα όπως εξαιρετικά εξειδικευμένες επιστήμες, υψηλή ταχύτητα τεχνολογικών αλλαγών, δυναμικότητα της γνώσης, μείωση των προϋπολογισμών έρευνας και εμφάνιση των διεπιστημονικών και διεπιστημονικών τομέων. Όταν σκεφτόμαστε αυτά τα θέματα, καταλαβαίνουμε ότι αυτές τις μέρες, ένα άτομο δεν μπορεί να είναι ειδικός σε όλες τις επιστήμες και τις τεχνικές όπως στο παρελθόν και να μην μπορεί να παρακολουθεί μόνο του την πορεία της γνώσης και της έρευνας[9]. Σήμερα, η αλληλεπίδραση των συγγραφέων αναγνωρίζεται ως βάση της ερευνητικής πρακτικής. Οι Acedo et al. επισημαίνοντας τις σπάνιες ακαδημαϊκές εργασίες με περισσότερους από έναν συγγραφείς στο πρώτο μισό του εικοστού αιώνα, ανέφερε την αυξανόμενη επιθυμία της συν-συγγραφέα σε επιστημονικές δημοσιεύσεις τις τελευταίες δεκαετίες. Στην πραγματικότητα, η συν-συγγραφέας είναι μια από τις πιο απτές μορφές ερευνητικής συνεργασίας. Η πολλαπλότητα και η ποικιλομορφία των ομαδικών κειμένων σε ένα πεδίο οδήγησαν στο σχηματισμό ενός κοινού δικτύου συγγραφέων ή συν-συγγραφέων, το οποίο έχει πολλές ομοιότητες με την επιστημονική κοινότητα και τη δομή της γνώσης στα ακαδημαϊκά περιβάλλοντα. Σε αυτό το δίκτυο , οι

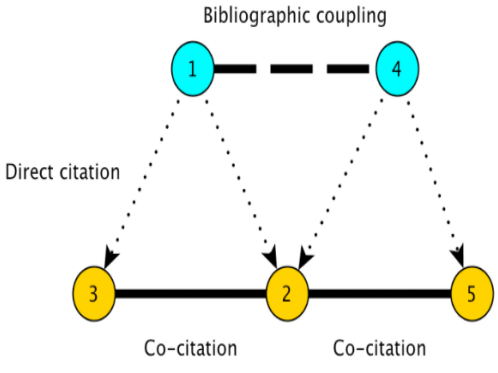
συγγραφείς ως συσχετισμένες οντότητες, αποτελούν το παγκόσμιο σύστημα παραγωγής γνώσης. Ένα δίκτυο συν-συγγραφέων είναι ένα κοινωνικό δίκτυο στο οποίο οι συγγραφείς μέσω της συμμετοχής τους σε μία ή περισσότερες δημοσιεύσεις μέσω μιας έμμεσης διαδρομής έχουν συνδεθεί μεταξύ τους. Σε ένα δίκτυο συν-συγγραφέων, οι συγγραφείς είναι οι κόμβοι του δικτύου και οι σύνδεσμοί τους είναι ο αριθμός των κοινών γραπτών τους που συνδέονται με μια γραμμή. Τα δίκτυα συν-συγγραφέων είναι οι καλύτεροι βιβλιομετρικοί δείκτες για την απεικόνιση διαφορετικών μοτίβων συν-συγγραφέων ακαδημαϊκών κλάδων, που μπορούν να εξετάσουν το χαρακτηριστικό αυτού του δικτύου χρησιμοποιώντας διάφορα μέτρα ανάλυσης κοινωνικού δικτύου[12]. Μέχρι στιγμής, πραγματοποιήθηκαν πολλές μελέτες στο δίκτυο συν-συγγραφέων χρησιμοποιώντας μέτρα SNA σε διαφορετικούς τομείς. Μερικές περιπτώσεις αυτών των τύπων είναι: Μερικές έρευνες στη νανοτεχνολογία, η μελέτη του δικτύου συν-συγγραφέων του Πανεπιστημίου Ιατρικών Επιστημών του Ιράν, η περιοχή της Ιρανικής Επείγουσας Ιατρικής και η οπτικοποίηση του δικτύου συν-συγγραφέων της Εφημερίδας της Σαηεντομετρικής. Τέτοιες μελέτες, εκτός από την οπτικοποίηση της κοινωνικής δομής των επιστημονικών αλληλεπιδράσεων, μπορούν να θεωρηθούν ως εργαλείο για την αυτο-αξιολόγηση των περιοδικών[10].



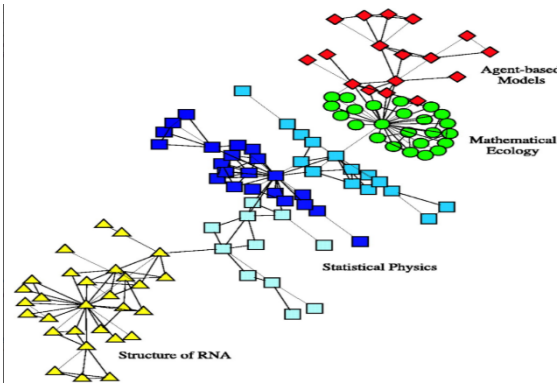
Εικόνα 3. Κοινωνικό δίκτυο Αμερικανών Γιατρών

1.4 Bibliographic Network

Η βιβλιογραφική ανάλυση δεδομένων μπορεί να εφαρμοστεί σε πολλά έργα σε διαφορετικούς τομείς. Υπάρχουν διάφοροι στόχοι, εκτός της ερευνητικής αξιολόγησης, όπως η κατανόηση της εξέλιξης της έρευνας. Η βιβλιομετρική ανάλυση βοηθά στην απόφαση για το ποια έργα ή ερευνητές θα πρέπει να λάβουν περισσότερη υποστήριξη, ποιος θα πρέπει να είναι αναθεωρητής, πώς να εξελίσσουν τα θέματα ενός συνεδρίου ή ενός ημερολογιακού χρόνου. Τα βιβλιογραφικά δεδομένα εξάγονται από διαδικτυακές βάσεις δεδομένων όπως DBLP, ACM, PubMed, NCBI κ.λπ. Γίνεται η συλλογή μεγάλων δεδομένων για επιστημονικές δημοσιεύσεις σε διάφορους τομείς, συμπεριλαμβανομένων πληροφοριών για συγγραφείς (π.χ. όνομα και ιδρύματα) και λεπτομέρειες δημοσιεύσεων (τίτλοι, συνέδρια, λέξεις-κλειδιά, ημερομηνία δημοσίευσης και παραπομπές). Είναι επομένως δυνατό να δημιουργηθούν δίκτυα όπως δίκτυο συν-συγγραφέων(co-authorship network), δίκτυο αναφορών και ούτω καθεξής. Το δίκτυο μπορεί να αναπαρασταθεί ως γράφημα που περιέχει κόμβους και άκρα, όπως ένα κοινωνικό δίκτυο. Τα βιβλιογραφικά δεδομένα έχουν χρησιμοποιηθεί ως βάση πολλών μελετών που εστιάζουν σε διαφορετικές προκλήσεις. Ο Muhlenbach et al προτείνει να ανακαλυφθούν ερευνητικές κοινότητες. Ο Gupta et al σχεδιάζει έναν αλγόριθμο ομαδοποίησης για την εξέλιξη του δικτύου. Οι «κόμβοι» τους στη συνέχεια ήταν έγγραφα, συγγραφείς, συνέδρια και όροι. Ο αλγόριθμος εξηγεί την εξέλιξη τόσο σε επίπεδο αντικειμένου όσο και σε επίπεδο ομαδοποίησης. Ο Huang et al εισήγαγε την ανίχνευση της εξέλιξης των σημασιολογικών κοινοτήτων που εξήχθησαν από τίτλους άρθρων. Δημιούργησαν ένα δίκτυο σύνδεσης λέξεων με βάση τις σχέσεις λέξεων σε τίτλους, χρησιμοποιώντας στατιστικές συχνότητες διανομής στα άκρα για να ταξινομήσουν δύο κοινότητες. Ο Deng et al παρουσίασε τρία μοντέλα προσεγγίσεων εξεύρεσης εμπειρογνωμόνων λαμβάνοντας υπόψη τις δημοσιεύσεις. Τα μοντέλα τους περιλάμβαναν το μοντέλο στατιστικής γλώσσας, το μοντέλο με βάση το θέμα και ένα υβριδικό μοντέλο. Οι Pham και Klamma παρείχαν μια οπτικοποίηση χρησιμοποιώντας ανάλυση παραπομπής. Το Social Network Analysis (SNA) χρησιμοποιείται για τον προσδιορισμό ζητημάτων ομαδοποίησης. Το αποτέλεσμα παρουσιάζεται σε επίπεδο ομαδοποίησης[6]. Αρκετές έρευνες μελέτησαν τις βάσεις δημοσιευμένων εφημερίδων προκειμένου να παρέχουν ένα εργαλείο ή μια διεπαφή του χρήστη για την παρακολούθηση και την εξερεύνηση αυτών των δεδομένων[22].



Εικόνα 4. Βιβλιογραφική σύζευξη



Εικόνα 5. Παράδειγμα δικτύου στην Φύση

1.5 Θεματολογία και Στόχοι της Διπλωματικής

Η εν λόγω διπλωματική επικεντρώνεται στην δημιουργία του αληθινού δικτύου των συν-συγγραφέων, αναλύοντας μετρικές του δικτύου και καταλήγοντας σε συμπεράσματα για τις συνδέσεις των γράφων μεταξύ τους και ενός τεχνητού δικτύου γράφων που αντικατοπτρίζει χαρακτηριστικά και δομή παρόμοια με το αληθινό δίκτυο. Για την επίτευξη αυτού του στόχου, χρησιμοποιούνται γενετικοί αλγόριθμοι, οι οποίοι εμπνέονται από τη φυσική εξέλιξη, προσφέροντας μια αυτοματοποιημένη διαδικασία βελτιστοποίησης της δομής του τεχνητού δικτύου.

Η δημιουργία τεχνητών δικτύων μέσω της χρήσης γενετικών αλγορίθμων αποτελεί μια εξελιγμένη προσέγγιση για τη μελέτη και την ανάλυση των δομών που παρατηρούμε στα πραγματικά δίκτυα. Με αφετηρία τα χαρακτηριστικά των πραγματικών δεδομένων, οι γενετικοί αλγόριθμοι προσπαθούν να αναπαράγουν τις δομές και την εξέλιξη των δικτύων σε ένα τεχνητό περιβάλλον. Η διαδικασία αυτή επιτρέπει τη δημιουργία προσαρμοσμένων τεχνητών δικτύων που μπορούν να χρησιμοποιηθούν για προβλέψεις, αναλύσεις και πειραματισμούς.

Η αντιστοίχιση του αληθινού δικτύου με ένα τεχνητό δίκτυο είναι μια πρόκληση στον τομέα της ανάλυσης δικτύων. Με την εφαρμογή γενετικών αλγορίθμων, εξερευνούμε διάφορες πιθανές διαμορφώσεις του τεχνητού δικτύου, προσεγγίζοντας σταδιακά τις διαστάσεις και τις συνδέσεις που χαρακτηρίζουν το πραγματικό δίκτυο. Με αυτόν τον τρόπο, μπορούμε να κατανοήσουμε τις αλληλεπιδράσεις μεταξύ των στοιχείων και τις δυναμικές που διέπουν την λειτουργία του αληθινού δικτύου. Οι γενετικοί αλγόριθμοι προσφέρουν μια εξελικτική προσέγγιση για την εύρεση βέλτιστων διαμορφώσεων, μετασχηματίζοντας και προσαρμόζοντας το τεχνητό δίκτυο. Η διαδικασία αυτή επιτρέπει την εμφάνιση νέων δομών και τη βελτίωση της απόδοσης του τεχνητού δικτύου, καθώς συγκλίνει προς μια πιο ρεαλιστική αναπαράσταση του αληθινού συστήματος.

Η ενσωμάτωση των γενετικών αλγορίθμων στη δημιουργία τεχνητών δικτύων παρέχει μια αντικειμενική προσέγγιση για την εξέταση της πολυπλοκότητας και της δομής των αληθινών δικτύων. Συμβάλλει στην αποτελεσματικότερη αξιοποίηση των παρατηρησιακών δεδομένων και της γνώσης που έχουμε για τα πραγματικά δίκτυα. Ως αποτέλεσμα, μπορούμε να αντιληφθούμε καλύτερα τις συνδέσεις και τις εναλλακτικές διαμορφώσεις που οδηγούν σε διαφορετικές συμπεριφορές των δικτύων. Τέτοιου είδους μελέτες μπορούν να ενισχύσουν την κατανόηση των βασικών διεργασιών που διαδραματίζονται σε πολύπλοκα συστήματα και να προσφέρουν νέες ευκαιρίες για τη βελτιστοποίηση και την πρόοδο των τεχνολογιών που βασίζονται στα δίκτυα.

Συνολικά, η παρούσα ερευνητική εργασία παρέχει μια προηγμένη μεθοδολογία για τη μελέτη και την ανάλυση δικτύων, συνδυάζοντας τις παρατηρήσεις από τα αληθινά δίκτυα με τη δημιουργία εξελιγμένων τεχνητών δικτύων. Οι γενετικοί αλγόριθμοι αποτελούν μια ισχυρή εργαλειοθήκη για την προσέγγιση της πολυπλοκότητας των δικτύων και την εξερεύνηση νέων ευκαιριών στο πεδίο της επιστήμης και της τεχνολογίας. Τα αποτελέσματα της έρευνας θα συνεισφέρουν σημαντικά στον εμπλουτισμό της γνώσης μας για τη λειτουργία των δικτύων και την αξιοποίηση τους σε εφαρμογές που καλύπτουν ποικίλους τομείς, από την επιστήμη και τη μηχανική, μέχρι την ιατρική και την κοινωνική δικτύωση. Επιπλέον, η προσέγγιση με γενετικούς αλγορίθμους μπορεί να προσφέρει νέες διαδικασίες βελτιστοποίησης και σχεδιασμού, οι οποίες είναι ιδιαίτερα επιτακτικές σε πολύπλοκα και μη γραμμικά προβλήματα που σχετίζονται με δίκτυα.

Κατά τη διεξαγωγή της διπλωματικής, εστιάζουμε στην ανάπτυξη μιας συστηματικής προσέγγισης για την εύρεση των κατάλληλων παραμέτρων και διαμορφώσεων του τεχνητού δικτύου που θα το προσεγγίσει όσο το δυνατόν περισσότερο στο αληθινό. Χρησιμοποιούμε πληθώρα μετρικών για την αξιολόγηση της ποιότητας της προσέγγισης, λαμβάνοντας υπόψη τόσο τις τοπολογικές όσο και τις λειτουργικές πτυχές του δικτύου.

Η ερευνητική προσπάθεια προσβλέπει στην αποτελεσματική εξέλιξη και βελτίωση των γενετικών αλγορίθμων, προκειμένου να επιτευχθεί ακόμα μεγαλύτερη ακρίβεια και αξιοπιστία στην προσέγγιση των δικτύων. Το επιτευχθέν αποτέλεσμα θα προσφέρει σημαντικές ευκαιρίες για την περαιτέρω κατανόηση, την ανάλυση και την αξιοποίηση των πολύπλοκων δικτύων που συναντούνται στη φύση και την τεχνολογία. Μέσω αυτής της προσπάθειας, ελπίζουμε να εμβαθύνουμε στις διαδικασίες της εξέλιξης και της διαμόρφωσης των δικτύων, αποκαλύπτοντας νέες πτυχές και προοπτικές που θα ενισχύσουν την επιστημονική μας κατανόηση του περίπλοκου κόσμου των δικτύων.

2. Ανάλυση Κώδικα, Δυσκολίες, Τεχνικές

2.1 Ανάλυση Αρχείου PDF και κώδικα

2.1.1 Αρχείο PDF και περιβάλλον εργασίας

Το dataset πάρθηκε από το pdf με τίτλο «**A survey and analysis of the first 40 years of scholarly literature in DEA: 1978-2016»** όπου αναφέρονται μελέτες που σχετίζονται με την ανάλυση φακέλων δεδομένων (DEA) από το αρχικό έργο των Charnes, Cooper και Rhodes [Μέτρηση της αποτελεσματικότητας των μονάδων λήψης αποφάσεων. European Journal of Operational Research 1978, 2 (6): 429–44] (αναφέρεται στο CCR). Αυτή η εργασία στοχεύει στην παροχή πλήρους λίστας εκδόσεων DEA από το 1978. Είναι γεγονός πως αυτή η λίστα άρθρων που σχετίζονται με την DEA είναι η πληρέστερη πηγή αναφορών στη θεωρία DEA και στις εφαρμογές της στη μέτρηση της αποτελεσματικότητας, της παραγωγικότητας ή της απόδοσης μονάδες λήψης αποφάσεων (DMU). Το περιεχόμενο του εγγράφου περιέχει εγγραφές διαφόρων άρθρων από το 1978 μέχρι το 2016 με την εξής μορφή:

17. Aghayi, N., M. Tavana and M. Ali Raayatpanah, Robust efficiency measurement with common set of weights under varying degrees of conservatism and data uncertainty. *European Journal of Industrial Engineering*, 2016. **10**(3): p. 385-405.

Εικόνα 6. Περιεχόμενο εγγραφής από το αρχείο PDF

Σκοπός της εργασίας είναι η άντληση της πληροφορίας των βιβλιογραφιών για κάθε εγγραφή μέσα στο pdf και η μεταξύ τους συσχέτιση, ώστε τελικά να δημιουργηθεί ένα κοινωνικό δίκτυο αποτελούμενο από τα papers και τις μεταξύ τους αναφορές. Για την άντληση της πληροφορίας έγινε χρήση της γλώσσας προγραμματισμού python[21] σε περιβάλλον spyder μέσω του anaconda[20].

2.1.2 WorkingPDFtoTEXT.py

Αρχικά διαβάζουμε το αρχείο pdf με την χρήση της βιβλιοθήκης pdfminer, δημιουργώντας ένα text αρχείο με το περιεχόμενο του pdf (WorkingPDFtoTEXT.py).

```
import io
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import os

#Converts pdf returning its text content as a string


def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)

    output = io.StringIO()
    manager = PDFResourceManager()
    converter = TextConverter(manager, output, laparams=LAParams(char_margin = 20, word_margin = 20))
    interpreter = PDFPageInterpreter(manager, converter)

    infile = open(fname, 'rb')
    for page in PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
    infile.close()
    converter.close()
    text = output.getvalue()
    output.close
    return text

def convertMultiple(pdfDir, txtDir):
    if pdfDir == "": pdfDir = os.getcwd() + "\\" #If no pdfDir passed in
    for pdf in os.listdir(pdfDir): #Iterate through pdfs in pdf directory
        fileExtension = pdf.split(".")[-1]
        if fileExtension == "pdf":
            pdfFilename = pdfDir + pdf
            text = convert(pdfFilename) #Get string of text content of pdf
            textFilename = txtDir + pdf + ".txt"
            textFile = open(textFilename, "w", encoding="utf-8") #Make text file
            textFile.write(text) #Write text to text file


#Set paths accordingly:
pdfDir = r'C:\Users\W. A. Grammatikos\Desktop\Desk\\'
txtDir = r'C:\Users\W. A. Grammatikos\Desktop\\'
convertMultiple(pdfDir, txtDir)
```

Εικόνα 7. Κώδικας αρχείου WorkingPDFtoText.py

2.1.3 NoSpacesText.py

Στην συνέχεια αφαιρούμε όλα τα κενά από το αρχείο κειμένου που δημιουργήσαμε προηγουμένως καθώς πρέπει να συλλέξουμε κάθε εγγραφή (NoSpacesText.py). Η αφαίρεση των κενών πραγματοποιήθηκε διότι, όπως θα δούμε στην συνέχεια, δεν θα μπορούσαμε να πάρουμε κάθε ένα match μέσω του regex.

```
#Editing the PDF text by removing all spaces

with open(r'C:\Users\N. A. Grammatikos\Desktop\dea2.txt', "r", encoding = 'utf-8') as read_file:
    with open(r'C:\Users\N. A. Grammatikos\Desktop\deaaac2.txt', "w", encoding = 'utf-8') as write_file:
        write_file.write(read_file.read().replace("\n", ''))
```

Εικόνα 8. Κώδικας αρχείου NoSpacesText.py

2.1.4 NonAsciiText.py

Παρόμοια αφαιρούμε όλους τους μη-Ascii χαρακτήρες, που δεν αναγνωρίζονται από τα συγκεκριμένα regex. Ως αποτέλεσμα της μη αφαίρεσης των χαρακτήρων αυτών είχαμε ένα μεγάλο mismatch στα papers του εγγράφου pdf, από 10300 σε περίπου 8000 (NonAsciiText.py)

```
#Editing the PDF text by removing all non-Ascii characters

with open(r'C:\Users\N. A. Grammatikos\Desktop\deaaac2.txt', "r", encoding = 'utf-8') as read_file:
    first_read_file = read_file.read()
    encoded_string = first_read_file.encode("ascii", "ignore")
    decode_string = encoded_string.decode()
    with open(r'C:\Users\N. A. Grammatikos\Desktop\deaac2NONACSCII.txt', "w", encoding = 'utf-8') as write_file:
        write_file.write(decode_string)
with open(r'C:\Users\N. A. Grammatikos\Desktop\deaac2NONACSCII.txt', "r", encoding = 'utf-8') as non_ascii:
    f_rfile = non_ascii.read()
    no_quotes = f_rfile.replace('"',' ')
    with open(r'C:\Users\N. A. Grammatikos\Desktop\deaac3NOQUOTES.txt', "w", encoding = 'utf-8') as w_nonascii:
        w_nonascii.write(no_quotes)
```

Εικόνα 9. Κώδικας αρχείου NonAsciiText.py

2.1.5 UsingRegex.py

Στην συνέχεια, χρησιμοποιώντας το regex προσπαθούμε να βρούμε μία κοινή ιδιότητα των εγγραφών ώστε να αντλήσουμε την πληροφορία από το έγγραφο pdf. Στο τμήμα αυτό της εργασίας παρουσιάστηκαν αρκετά προβλήματα που είχαν ως κύριο χαρακτηριστικό την μη άντληση όλων των εγγραφών για λόγους όπως:

1) Διαφορετική δομή:

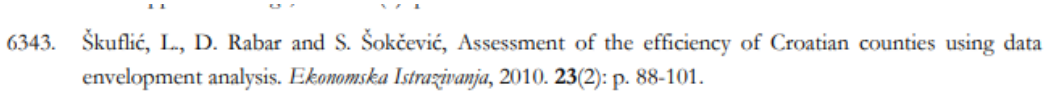
18. Agovino, M. and A. Rapposelli, Disability and Work: A Two-Stage Empirical Analysis of Italian Evidence at Provincial Level in Providing Employment for Disabled Workers. *Social Indicators Research*, 2016. **125**(2): p. 635-648.

19. Ahamed, S.K., M.M. Naidu and C. Subba Rami Reddy, Outliers/most influential observationsin variable returns to scale data envelopment analysis. *Indian Journal of Science and Technology*, 2016. **9**(2).

Εικόνα 10. Παρουσίαση δομής χαρακτήρων Ascii

Παρατηρούμε ότι η εγγραφή 18 αναφέρει τα ονόματα των συγγραφέων, τον τίτλο του paper, χρονολογία και τελειώνει με τις σελίδες. Αντιθέτως η εγγραφή 19 αν και αναφέρει παρόμοια τα ονόματα των συγγραφέων όπως και τον τίτλο, δεν τελειώνει με τις σελίδες, προκαλώντας την λανθασμένη άντληση του συγκεκριμένου match.

2) Non-Ascii Χαρακτήρες



6343. Škuflić, L., D. Rabar and S. Šokčević, Assessment of the efficiency of Croatian counties using data envelopment analysis. *Ekonomska Istrazivanja*, 2010. **23**(2): p. 88-101.

Εικόνα 11. Παρουσίαση δομής χαρακτήρων Non-Ascii

Κύριος λόγος και η δημιουργία του NonAsciiText.py, με τον τρόπο που δημιουργήθηκε το regex αγνοούσε τις εγγραφές με ονόματα συγγραφέων που ξεκινούσαν με χαρακτήρες non-Ascii. Αν και πραγματοποιήθηκε η πρόσθεση όλων των μη-Ascii χαρακτήρων δεν γινόταν η σωστή άντληση των εγγραφών με αποτέλεσμα να αφαιρεθούν όλοι οι χαρακτήρες Ascii σε ένα νέο αρχείο κειμένου.

Η συγκεκριμένη δομή του regex αποτελείται από:

14. Agasisti, T. and J. Wolszczak-Derlacz, Exploring efficiency differentials between Italian and Polish universities, 2001-11. *Science and Public Policy*, 2016. **43**(1): p. 128-142.

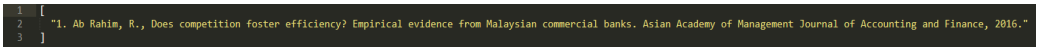
Εικόνα 12. Παρουσίαση χρωματισμένου κειμένου μετά από επεξεργασία μέσω regex

\d+ → Αριθμό με άπειρα ψηφία
\. → Μέχρι να συναντήσει . (τελεία)
\s → Ακριβώς μετά την τελεία έχουμε ένα κενό (white space)
[A-Za-z].*? → Μετά το κενό, το κείμενο ξεκινά με ένα κεφαλαίο γράμμα ή μικρό. Το σύμβολο .* σημαίνει πως κάνει match όλους τους χαρακτήρες μέχρι να συναντήσει τα στοιχεία μετά το ?
\d+ → Αριθμό με άπειρα ψηφία (Στοχεύοντας την χρονολογία)
\. → Να τελειώνει με . (τελεία)

Βρίσκοντας κάθε εγγραφή ξεχωριστά, δημιουργείται ένα αρχείο json, μοναδικό για κάθε εγγραφή, με περιεχόμενο την ίδια εγγραφή (UsingRegex.py)

```python
import re

#Acquiring the paper matches using regex

textfile = open(r'C:\Users\W. A. Grammatikos\Desktop\deaac3NOQUOTES.txt', "r", encoding = 'utf-8')

filetext = textfile.read()
textfile.close()
matches = re.findall(r'(\d+\.\s[A-Za-z].*?\d+\.)', filetext)

#\d+\.\s[A-Z].*?,\s\d\d\d\d  ---> τελειώνουμε σε χρονολογία
#A-ZàáâèéêëìíîöœùûüÿçÀÁÂÈÉÊËÌÍÎÖŒÙÛÜŸÇ ---> non-Ascii χαρακτήρες
#\d+\.\s[A-Z].*?p\.\s\d+-\d+\. ---> τελειώνουμε σε σελίδες και .

for i, match in enumerate(matches):
    with open("C:\\Users\\W. A. Grammatikos\\Desktop\\Matches\\match{0:04d}.json".format(i), 'w', encoding = 'utf-8') as nf:
        nf.write('[\n  "' +match+'"\n]')
```

Εικόνα 13. Κώδικας αρχείου UsingRegex.py



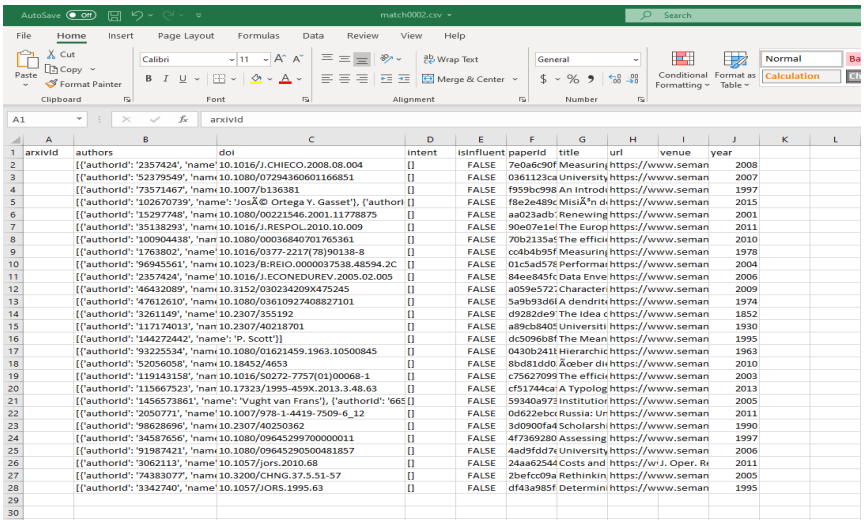
```
[
  "1. Ab Rahim, R., Does competition foster efficiency? Empirical evidence from Malaysian commercial banks. Asian Academy of Management Journal of Accounting and Finance, 2016."
]
```

Εικόνα 14. Παράδειγμα εγγραφής 0001 (match0001.json)

2.1.6 WholeCodeWithFunction.py

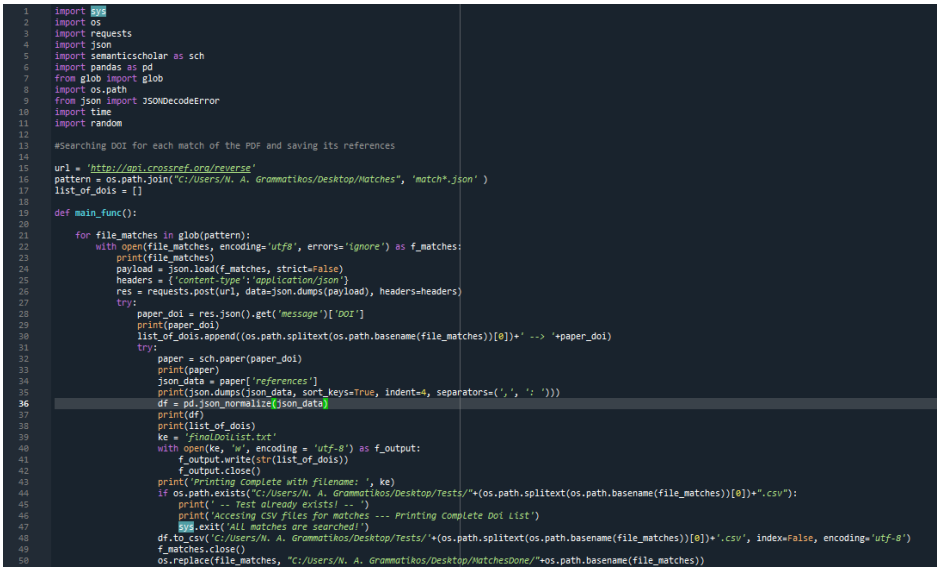
Έχοντας αντλήσει κάθε εγγραφή επιτυχώς από το κείμενο και έχοντας δημιουργήσει ένα αρχείο json για κάθε μία εγγραφή ξεχωριστά θα χρειαστεί για κάθε paper να βρεθεί το DOI του. Το DOI, ή Digital Object Identifier, είναι μια σειρά αριθμών, γραμμάτων και συμβόλων που χρησιμοποιούνται για την μόνιμη αναγνώριση ενός άρθρου ή εγγράφου και τη σύνδεσή του με αυτόν στον ιστό. Ένα DOI θα βοηθήσει τον αναγνώστη να εντοπίσει εύκολα ένα έγγραφο από τις αναφορές(βιβλιογραφία). Ενώ μια διεύθυνση ιστού (URL) ενδέχεται να αλλάξει, το DOI δεν θα αλλάξει ποτέ[13].

Για να βρεθεί το DOI του κάθε paper χρησιμοποιούμε το API του Crossref (πιο ειδικά http://api.crossref.org/reverse)[14]. Αρχικά φορτώνουμε σε ένα payload το αρχείο json για κάθε ένα match που έχουμε αποθηκεύσει. Συνεχίζοντας στέλνουμε στο συγκεκριμένο API το payload και λαμβάνουμε πίσω μία απάντηση της μορφής json, που περιέχει το DOI του paper. Στην συνέχεια απομονώνουμε το DOI, διαβάζοντας πρώτα το επιστρεφόμενο string. Το DOI είναι σημαντικό καθώς στην παρούσα εργασία χρειάζεται για κάθε ένα για τα papers να βρεθεί η βιβλιογραφία τους. Για το λόγο αυτό χρησιμοποιείται η βιβλιοθήκη του Semantic Scholar[18] στην python. Το Semantic Scholar[17] χρησιμοποιείται για την επιτάχυνση των επιστημονικών ανακαλύψεων βοηθώντας τους μελετητές να εντοπίσουν και να κατανοήσουν τη σωστή έρευνα, να κάνουν σημαντικές συνδέσεις και να ξεπεράσουν την υπερφόρτωση πληροφοριών. Με το DOI που βρέθηκε προηγουμένως και με την βοήθεια του Semantic Scholar καταφέρνουμε να αντλήσουμε την πληροφορία της βιβλιογραφίας για κάθε paper που εμπεριέχεται στο φάκελο των εγγραφών μας. Στο εξής σημείο, αξίζει να αναφερθεί πως, αν και η διαδικασία φαντάζει απλή η εκτέλεση παρέμενε αρκετά πολύπλοκη καθώς ορισμένα papers είτε ήταν κλειστά για το κοινό, έχοντας μόνο πρόσβαση στις πληροφορίες του με πληρωμή, είτε βρισκόντουσαν σε κλειστά πανεπιστημιακά περιβάλλοντα είτε ο κάτοχος της πληροφορίας έκανε γνωστό μέσω του TOS πως η χρήση των πληροφοριών των papers που είχε στην διάθεση του ήταν παράνομη χωρίς την έγκριση του εκάστοτε κατόχου. Επιπρόσθετα η βιβλιοθήκη του Semantic Scholar επέτρεπε την άντληση πληροφοριών για 100 papers ανά 5 λεπτά, κάνοντας την διαδικασία αρκετά χρονοβόρα. Σε ορισμένες περιπτώσεις το string που επέστρεφε ήταν κενό με αποτέλεσμα να χρειαστεί το πρόγραμμα να τρέξει για πληθώρα papers που αν και ήταν εφικτό να ληφθεί η πληροφορία είτε επειδή ο server υπερφορτωνόταν είτε επειδή το επιτρεπτό όριο είχε υπερβεί. Είναι αναγκαίο επίσης να σημειωθεί πως η ανάκτηση του DOI αλλά και η άντληση της βιβλιογραφίας για κάθε μία εγγραφή διαρκούσε από 1 εώς 3 λεπτά. Επικουρικά όλα τα DOI που βρέθηκαν, σώθηκαν σε μία λίστα που αργότερα θα μας βοηθήσει για την άντληση και των abstract, ενώ η βιβλιογραφία του κάθε paper σώθηκε με το όνομα του match του ως αρχείο .csv (Παράδειγμα του match0002.csv):



Εικόνα 15. Αρχείο αποτελεσμάτων του match0002.csv

Γίνεται η χρήση σύναρτησης της python με όνομα main_func κυρίως για τον λόγο που αν επιστραφεί ως αποτέλεσμα ένα error να ξαναρχίσει το πρόγραμμα από το τελευταίο match. Φαίνεται επίσης πως για κάθε match το οποίο έχει επεξεργαστεί και έχουν αντληθεί οι πληροφορίες μεταφέρονται στον φάκελο MatchesDone ώστε να σε περίπτωση λάθους να παραμείνουν στον φάκελο Matches μόνο οι εγγραφές που δεν έχουν ακόμα επεξεργαστεί (WholeCodeWithFunction.py).



```python
import sys
import os
import requests
import json
import semanticscholar as sch
import pandas as pd
from glob import glob
import os.path
from json import JSONDecodeError
import time
import random

#Searching DOI for each match of the PDF and saving its references

url = 'http://api.crossref.org/reverse'
pattern = os.path.join("C:/Users/N. A. Grammatikos/Desktop/Matches", 'match*.json' )
list_of_dois = []

def main_func():

    for file_matches in glob(pattern):
        with open(file_matches, encoding='utf8', errors='ignore') as f_matches:
            print(file_matches)
            payload = json.load(f_matches, strict=False)
            headers = {'content-type':'application/json'}
            res = requests.post(url, data=json.dumps(payload), headers=headers)
            try:
                paper_doi = res.json().get('message')['DOI']
                print(paper_doi)
                list_of_dois.append((os.path.splitext(os.path.basename(file_matches))[0])+' --> '+paper_doi)
                try:
                    paper = sch.paper(paper_doi)
                    print(paper)
                    json_data = paper['references']
                    print(json.dumps(json_data, sort_keys=True, indent=4, separators=(',', ': ')))
                    df = pd.json_normalize(json_data)
                    print(df)
                    print(list_of_dois)
                    ke = 'finalDoiList.txt'
                    with open(ke, 'w', encoding = 'utf-8') as f_output:
                        f_output.write(str(list_of_dois))
                        f_output.close()
                    print('Printing Complete with filename: ', ke)
                    if os.path.exists("C:/Users/N. A. Grammatikos/Desktop/Tests/"+(os.path.splitext(os.path.basename(file_matches))[0])+".csv"):
                        print(' -- Test already exists! -- ')
                        print('Accesing CSV files for matches --- Printing Complete Doi List')
                        sys.exit('ALL matches are searched!')
                    df.to_csv('C:/Users/N. A. Grammatikos/Desktop/Tests/'+(os.path.splitext(os.path.basename(file_matches))[0])+'.csv', index=False, encoding='utf-8')
                    f_matches.close()
                    os.replace(file_matches, "C:/Users/N. A. Grammatikos/Desktop/MatchesDone/"+os.path.basename(file_matches))
```

Εικόνα 16. Κώδικας αρχείου WholeCodeWithFunction.py (1)

```python
                except:
                    if os.path.exists("C:/Users/N. A. Grammatikos/Desktop/Tests/"+(os.path.splitext(os.path.basename(file_matches))[0])+".csv"):
                        f_matches.close()
                        print(' -- Test already exists! -- ')
                        print('Accesing CSV files for matches --- Printing Complete Doi List')
                        sys.exit('ALL matches are searched!')
                    f = open("C:/Users/N. A. Grammatikos/Desktop/Tests/"+(os.path.splitext(os.path.basename(file_matches))[0])+".csv",'w')
                    f.write('References Not Found!')
                    f.close()
                    print('References Not Found!')
                    ke = 'finalDoiList.txt'
                    with open(ke, 'w', encoding = 'utf-8') as f_output:
                        f_output.write(str(list_of_dois))
                        f_output.close()
                    f_matches.close()
                    os.replace(file_matches, "C:/Users/N. A. Grammatikos/Desktop/MatchesDone/"+os.path.basename(file_matches))
                    pass

            except (requests.exceptions.ConnectionError, JSONDecodeError, ValueError):
                    f_matches.close()
                    print('Returning to Start -- Probably a JSON empty String')
                    time.sleep(2 + random.random()*0.01)
                    return main_func()

main_func()

print('Accesing CSV files for matches --- Printing Complete Doi List')
sys.exit('ALL matches are searched!')
```

Εικόνα 17. Κώδικας αρχείου WholeCodeWithFunction.py (2)

2.1.7 FinalResultsCSV.py

Συνεχίζοντας, διαβάζεται ως λίστα το περιεχόμενο του αρχείου TestList1.txt και δημιουργείται το αρχείο AllignedList.txt που περιέχει όλα τα match με το ανάλογο DOI (AllignedList.txt).

```
match0000 --> 10.1016/s0038-0121(17)30333-6
match0001 --> 10.7176/rjfa/11-15-04
match0002 --> 10.1016/j.techfore.2015.10.007
match0003 --> 10.1108/h-09-2015-0058
match0004 --> 10.1016/j.dss.2016.06.003
match0005 --> 10.1108/md-09-2015-0413
match0006 --> 10.2298/fil1610653a
match0007 --> 10.4102/jef.v9i2.52
match0008 --> 10.1186/s13012-016-0434-2
match0009 --> 10.1016/j.ejor.2015.10.064
match0010 --> 10.1080/00036846.2015.1088145
match0011 --> 10.1007/s00291-016-0465-8
match0012 --> 10.1108/bij-07-2015-0074
match0013 --> 10.1007/s12597-015-0229-2
match0014 --> 10.1093/scipol/scv026
match0015 --> 10.17654/ms099111633
match0016 --> 10.1016/j.energy.2016.06.086
match0017 --> 10.1504/ejie.2016.076386
match0018 --> 10.1007/s11205-014-0851-z
match0019 --> 10.17485/ijst/2016/v9i2/80361
match0020 --> 10.1016/j.ejor.2015.07.045
match0021 --> 10.17010/ijf/2016/v10i2/87243
match0022 --> 10.1108/ijqrm-11-2014-0174
match0023 --> 10.1504/ijmef.2016.076475
match0024 --> 10.1016/j.cie.2016.10.003
match0025 --> 10.1142/s021962201550042x
match0026 --> 10.3390/su8020148
match0027 --> 10.22495/cocv13i4c3p6
match0028 --> 10.1108/ajems-01-2014-0007
match0029 --> 10.1007/s10644-015-9174-6
```

Εικόνα 18. Αποτελέσματα αρχείου AllignedList.txt

Από αυτό το αρχείο και κάνοντας ξανά χρήση του regex ψάχνουμε το DOI και το πρόγραμμα μας επιστρέφει το ανάλογο match. Τελικά δημιουργείται ένα τελικό αρχείο .csv με περιεχόμενο ένα source και ένα target που αναφέρει όλα τα κοινά βιβλιογραφικά στοιχεία(αναφορές) μεταξύ των matches (FinalResultsCSV.py)

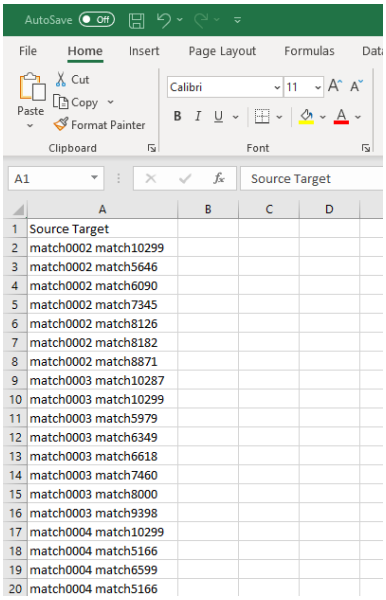
```
import csv
import glob
import re
import ast
import time
import os

#Creating the Source Target model for each match

with open('C:/Users/N. A. Grammatikos/Desktop/TestList1.txt', 'r', encoding = 'utf-8') as list1:
    read_list = ast.literal_eval(list1.read())
    list1.close()

myTestList = read_list

with open('C:/Users/N. A. Grammatikos/Desktop/AllignedList.txt', 'w', encoding = 'utf-8') as asList:
    for match in myTestList:
        asList.write(match + '\n')
    asList.close()

no_matches = re.sub(r'(match\d+\s-->\s)', '', str(myTestList)) # Removing all matches, leaving only the DOIs

with open('C:/Users/N. A. Grammatikos/Desktop/TestList1WITHOUTMATCHES.txt', 'w', encoding = 'utf-8') as list2:
    list2.write(no_matches)
    list2.close()

time.sleep(2)

def findDoi(doi_input):


    doi_input = doi_input.replace('(','\(')

    doi_input = doi_input.replace(')','\)')

    doiLookup = re.compile(doi_input)


    with open('C:/Users/N. A. Grammatikos/Desktop/AllignedList.txt', 'r', encoding = 'utf-8') as asListRead:
        for line in asListRead:
            for match in re.finditer(doiLookup, line):
                print(line)
                pat2 = re.findall('match\d+', line)
                print(str(*pat2))
                return str(*pat2)
                asListRead.close()

with open('C:/Users/N. A. Grammatikos/Desktop/TestList1WITHOUTMATCHES.txt', 'r', encoding = 'utf-8') as list3:
    doi_list = ast.literal_eval(list3.read())
    list3.close()
```

Εικόνα 19. Κώδικας αρχείου FinalResultsCSV.py (1)

```
write_header = True
output_csv = 'FinalResult.csv'

with open(output_csv, 'w', newline='', encoding = 'utf-8') as f_output:
    csv_output = csv.writer(f_output)

    for csv_filename in glob.glob('C:/Users/N. A. Grammatikos/Desktop/Tests/*.csv'):
        for doi in doi_list:
            if csv_filename != output_csv:
                with open(csv_filename, encoding = 'utf-8') as f_input:
                    csv_input = csv.reader(f_input)
                    header = next(csv_input)
                    if write_header:
                        csv_output.writerow(header)
                        write_header = False
                    for row in csv_input:
                        if doi in row[2].lower():
                            print(doi)
                            f_output.write((os.path.splitext(os.path.basename(csv_filename))[0]) +' '+str(findDoi(doi)))
                            print('Match Found')
                            csv_output.writerow('')
```

Εικόνα 20. Κώδικας αρχείου FinalResultsCSV.py (2)

Τελικά δημιουργείται το αρχείο FinalResult.csv:



	A	B	C	D
1	Source Target			
2	match0002 match10299			
3	match0002 match5646			
4	match0002 match6090			
5	match0002 match7345			
6	match0002 match8126			
7	match0002 match8182			
8	match0002 match8871			
9	match0003 match10287			
10	match0003 match10299			
11	match0003 match5979			
12	match0003 match6349			
13	match0003 match6618			
14	match0003 match7460			
15	match0003 match8000			
16	match0003 match9398			
17	match0004 match10299			
18	match0004 match5166			
19	match0004 match6599			
20	match0004 match5166			

Εικόνα 21. Αποτελέσματα αρχείου FinalResult.csv

Για παράδειγμα γίνεται η εξής παρατήρηση: Η εγγραφή με όνομα match0002 περιέχει στις αναφορές της το DOI της εγγραφής με όνομα match10299 και ούτω το καθεξής για τις υπόλοιπες εγγραφές.

2.1.8 AbstractsCode.py

Τελικά γίνεται η λήψη και των abstract των εγγραφών τρέχοντας τις συναρτήσεις gettingAbstract και reSearching όπου η πρώτη, όπως και με την βιβλιογραφία, με την βοήθεια του Semantic Scholar γίνεται η άντληση της πληροφορίας για κάθε DOI στην αρχική λίστα. Αν το αποτέλεσμα της απάντησης από τον server είναι κενό («{}») τότε γράφεται στον κείμενο του αρχείου το κείμενο «Information cannot be received», ενημερώνοντας την λίστα για το DOI που μόλις έγινε η επεξεργασία. Η δεύτερη συνάρτηση πραγματοποιεί μία επαλήθευση των πληροφοριών των matches, αν δηλαδή το περιεχόμενο των αρχείων κειμένου περιέχει το εξής κείμενο «Information cannot be received» τότε επαναλαμβάνει την διαδικασία για την ανάκτηση της πληροφορίας. Είναι σημαντικό να αναφερθεί ότι έχουν προστεθεί χρονικά διαλλείματα μεταξύ του κάθε σφάλματος καθώς είναι πολύ πιθανό να έχει προηγηθεί υπερφόρτωση του server ή να έχει ξεπεραστεί το όριο των επιτρεπτών εντολών σε αυτόν. (AbstractsCode.py)

```python
import os
import semanticscholar as sch
from glob import glob
import os.path
import time
import random
import ast
import requests
import json

#Acquiring abstract for each paper


pattern = os.path.join("C:/Users/N. A. Grammatikos/Desktop/MatchesDone", 'match*.json' )
pattern2 = os.path.join("C:/Users/N. A. Grammatikos/Desktop/Abstracts", 'match*.txt' )
doi_not_info = []
doi_not_info2 = []

with open('C:/Users/N. A. Grammatikos/Desktop/TestList1WITHOUTMATCHES.txt', 'r', encoding = 'utf-8') as list1:
    doi_list = ast.literal_eval(list1.read())
    list1.close()

with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List.txt', 'r', encoding = 'utf-8') as list1:
    doi_list = json.loads(list1.read())
    list1.close()

with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List_RE2.txt', 'r', encoding = 'utf-8') as list2:
    doi_list2 = json.loads(list2.read())
    list2.close()
```

Εικόνα 22. Κώδικας αρχείου AbstractsCode.py (1)

```python

def gettingAbstract():
    counter = 0
    j = 50
    for matches in glob(pattern):
        if counter == j:
            print('\nTime cooldown\n')
            time.sleep(30)
            j = j + 50
        else:
            for doi in doi_list:
                counter = counter + 1
                print(len(doi_list))
                try:
                    print(matches)
                    print(doi)
                    paper = sch.paper(doi, timeout=2)
                    if str(paper) == '{}':
                        print('Information cannot be received')
                        doi_not_info.append(doi)
                        with open('C:/Users/N. A. Grammatikos/Desktop/NoInfoDois.txt', 'w', encoding='utf-8') as apDoi:
                            apDoi.write(json.dumps(doi_not_info))
                            apDoi.close()
                        time.sleep(20 + random.random()*0.01)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Abstracts/'+(os.path.splitext(os.path.basename(matches))[0])+'.txt', 'w', encoding='utf-8') as wrAb1:
                            wrAb1.write('Information cannot be received')
                            wrAb1.close()
                        doi_list.remove(doi)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List.txt', 'w', encoding='utf-8') as delDoi:
                            delDoi.write(json.dumps(doi_list))
                            delDoi.close()
                        os.replace(matches, "C:/Users/N. A. Grammatikos/Desktop/MatchesDoneABSTRACT/"+os.path.basename(matches))
                        gettingAbstract()
                    else:
                        dataPaper = paper['abstract']
                        with open('C:/Users/N. A. Grammatikos/Desktop/Abstracts/'+(os.path.splitext(os.path.basename(matches))[0])+'.txt', 'w', encoding='utf-8') as wrAb2:
                            wrAb2.write(str(dataPaper))
                            wrAb2.close()
                        os.replace(matches, "C:/Users/N. A. Grammatikos/Desktop/MatchesDoneABSTRACT/"+os.path.basename(matches))
                        doi_list.remove(doi)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List.txt', 'w', encoding='utf-8') as delDoi:
                            delDoi.write(json.dumps(doi_list))
                            delDoi.close()
                        gettingAbstract()
                except(ValueError, KeyError, TimeoutError, requests.exceptions.Timeout):
                    print('\nRetrying..\n')
                    time.sleep(30 + random.random()*0.01)
                    gettingAbstract()


#gettingAbstract()
```

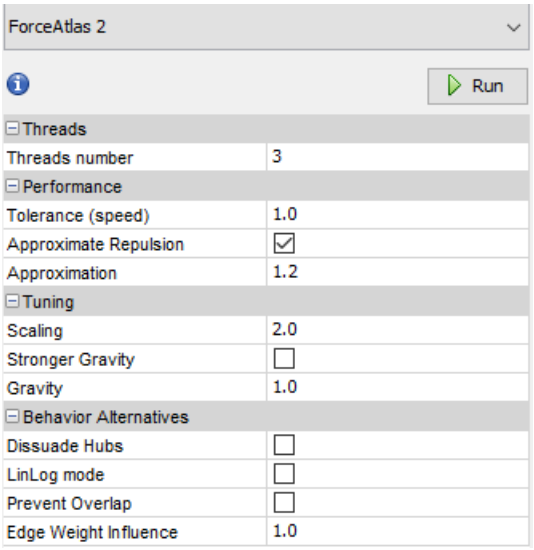
Εικόνα 23. Κώδικας αρχείου AbstractsCode.py (2)

```python
def reSearching():
    for abtext in glob(pattern2):
        for doi2 in doi_list2:
            with open(abtext, 'r', encoding = 'utf-8') as abOpen:
                abRead = abOpen.read()
                abOpen.close()
            if str(abRead) == 'Information cannot be received':
                try:
                    print(abtext)
                    print(doi2)
                    paper2 = sch.paper(doi2, timeout=2)
                    if str(paper2) == '{}':
                        print('Information cannot be received')
                        doi_not_info2.append(doi2)
                        with open('C:/Users/N. A. Grammatikos/Desktop/NoInfoDoisRE2.txt', 'w', encoding='utf-8') as apDoi2:
                            apDoi2.write(json.dumps(doi_not_info2))
                            apDoi2.close()
                        time.sleep(5 + random.random()*0.01)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Abstracts/'+(os.path.splitext(os.path.basename(abtext))[0])+'.txt', 'w', encoding='utf-8') as wrAb1:
                            wrAb1.write('Information cannot be received -- Retried Paper')
                            wrAb1.close()
                        doi_list2.remove(doi2)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List_RE2.txt', 'w', encoding='utf-8') as delDoi2:
                            delDoi2.write(json.dumps(doi_list2))
                            delDoi2.close()
                        reSearching()
                    else:
                        dataPaper2 = paper2['abstract']
                        with open('C:/Users/N. A. Grammatikos/Desktop/Abstracts/'+(os.path.splitext(os.path.basename(abtext))[0])+'.txt', 'w', encoding='utf-8') as wrAb2:
                            wrAb2.write(str(dataPaper2))
                            wrAb2.close()
                        doi_list2.remove(doi2)
                        with open('C:/Users/N. A. Grammatikos/Desktop/Remove_Doi_List_RE2.txt', 'w', encoding='utf-8') as delDoi2:
                            delDoi2.write(json.dumps(doi_list2))
                            delDoi2.close()
                        reSearching()
                except(ValueError, KeyError, TimeoutError, requests.exceptions.Timeout):
                        print('\nRetrying..\n')
                        time.sleep(30 + random.random()*0.01)
                        reSearching()
            else:
                pass

#reSearching()
```

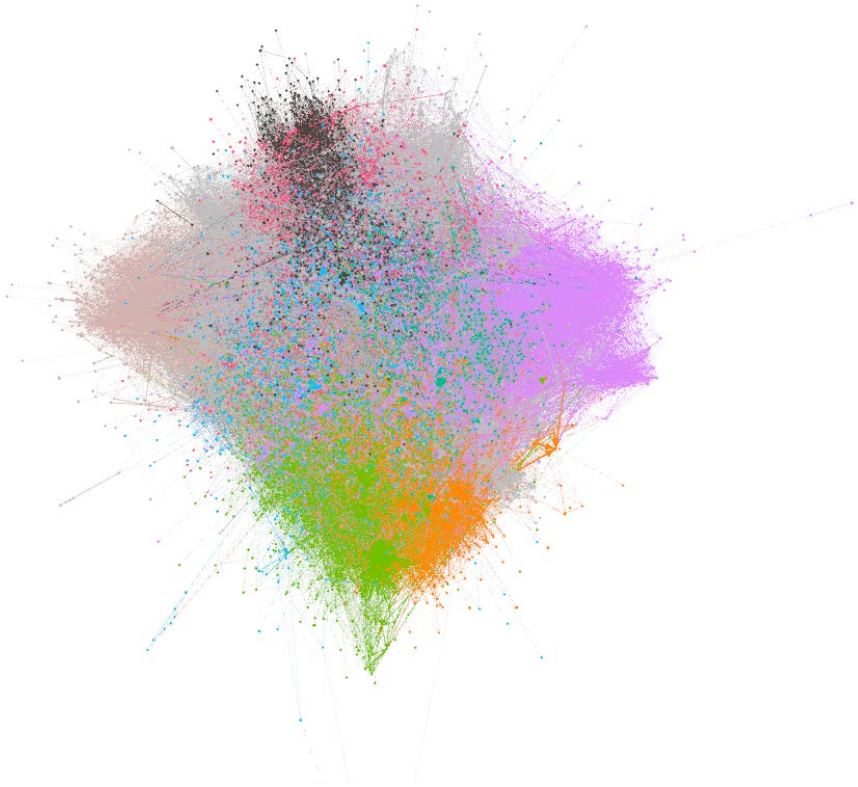
Εικόνα 24. Κώδικας αρχείου AbstractsCode.py (3)

2.2 Γραφήματα και στατιστικά από Gephi

Από το Gephi παρατηρούμε το εξής αποτέλεσμα τρέχοντας τον αρχείο FinalResult.csv με τον αλγόριθμο Force Atlas 2 και τις εξής ρυθμίσεις:



Εικόνα 25. Παράμετροι αλγορίθμου ForceAtlas 2



Εικόνα 26. Οπτικοποίηση δικτύου με χρωματισμένες γειτονίες

Η συμπλήρωση των χρωμάτων έχει πραγματοποιηθεί με βάση το Modularity Class όλων των εγγραφών. Παρατηρείται η διάσπαση των εγγραφών σε «γειτονιές» ανάλογα με το χρώμα τους και την θέση τους στο γράφημα. Επικουρικά παρατηρείται ότι οι πιο κεντρικά σε θέση κόμβοι έχουν περισσότερες ακμές και μεταβατικά η σημασία τους στο γράφημα είναι μεγαλύτερη από τους κόμβους που βρίσκονται στην άκρη του γράφου.

2.3 Ανάλυση Μετρικών Δικτύου

Degree Distribution: Η μέση βαθμίδα του γραφήματος μας δείχνει τον μέσο αριθμό συνδέσεων που έχει ένας κόμβος με έναν άλλον κόμβο. Παρατηρούμε πως τα περισσότερα papers έχουν μεταξύ τους 0 – 500 ενώσεις κατά κύριο λόγο συνδέονται με άλλα 10 papers κατά μέσο όρο. Είναι φανερό πως τα παλιότερα papers αναφέρονται στα καινούργια με μικρότερη πιθανότητα εμφάνισης όσο προχωρούμε σε πιο νεότερα. Εάν το γράφημα έχει άκρα, έχουμε τη δυνατότητα να προχωρήσουμε πέρα από την απλούστερη μέτρηση βαθμού και να το χωρίσουμε σε μετρήσεις In-Degree και Out-Degree. Αυτά τα δύο μέτρα μπορούν να μας πληροφορήσουν εάν ένα δίκτυο είναι πολύ αμοιβαίο (τα μέτρα In-Degree και Out-Degree είναι αρκετά παρόμοια) ή όχι[3].

Μέση Βαθμίδα - Average Degree: 10.854

Μέση Βαθμίδα Βάρους – Average Weighted Degree: 11.237

In-Degree Distribution: Για ένα κατευθυνόμενο γράφημα η ακολουθία In-Degree είναι μια ακολουθία που λαμβάνεται με την ταξινόμηση των βαθμών όλων των κορυφών σε αυξανόμενη σειρά. Συνεπώς στο γράφημα παρατηρούμε πως ένα μεγάλο ποσοστό των papers κατευθύνονται στο αρχικό σε πυκνότητα.
Out-Degree Distribution: Για ένα κατευθυνόμενο γράφημα η ακολουθία Out-Degree είναι μια ακολουθία που λαμβάνεται με την ταξινόμηση όλων των κορυφών σε αυξανόμενη σειρά. Συνεπώς στο γράφημα παρατηρείται πως από ένα μεγάλο αριθμό paper κατευθύνονται σε άλλα. Ένα άλλο σημαντικό μέτρο για κατευθυνόμενα γραφήματα είναι το μέτρο του αριθμού των Out-Degree, που ορίζονται ως συνδέσεις που ρέουν από έναν επιλεγμένο κόμβο σε μια σειρά άλλων μελών του δικτύου. Έχουμε ήδη σημειώσει ότι ένα υψηλό επίπεδο Out-Degree σε σχέση με In-Degree συχνά μας λέει ότι ένας συγκεκριμένος κόμβος δεν θεωρείται άμεση πηγή πληροφοριών. Αυτός ο ίδιος κόμβος μπορεί ωστόσο να παρέχει διαδρομές σε μια ποικιλία κρίσιμων πηγών, που χρησιμεύουν ως συγκεντρωτής πληροφοριών ή κόμβος για άλλους[3].
Connected Components: Όταν έχουμε ένα δίκτυο όπου υπάρχουν περισσότερα από ένα στοιχεία, το εργαλείο Συνδεδεμένα στοιχεία(Connected Components) μπορεί να χρησιμοποιηθεί για να μας παρέχει μια απλή ανάγνωση του αριθμού των διακριτών στοιχείων στο δίκτυο. Όταν το δίκτυό μας είναι πλήρως συνδεδεμένο, θα επιστραφεί μια τιμή 1, οπότε δεν υπάρχει ανάγκη για αυτόν τον υπολογισμό. Ωστόσο, σε πολύ μεγάλα δίκτυα μπορεί να είναι δύσκολο να προσδιοριστεί οπτικά εάν το δίκτυο είναι πλήρως συνδεδεμένο, επομένως μπορούμε να χρησιμοποιήσουμε αυτήν τη λειτουργία για να εξακριβώσουμε τον αριθμό των στοιχείων[5].

Number of Weakly Connected Components: 11
Number of Strongly Connected Components: 8471

Graph Density: Η πυκνότητα γραφήματος είναι ένα μέτρο του επιπέδου των συνδεδεμένων ακμών εντός ενός δικτύου σε σχέση με τη συνολική πιθανή τιμή και επιστρέφεται ως δεκαδική τιμή μεταξύ μηδέν και ενός. Τα γραφήματα με τιμές πιο κοντά στο ένα συνήθως θεωρούνται πυκνά γραφήματα, ενώ αυτά που πλησιάζουν το μηδέν ονομάζονται αραιά γραφήματα. Αυτό που συνιστά πυκνό σε σχέση με αραιό θα ποικίλει ανάλογα με τον τύπο των δεδομένων δικτύου που μελετάται, επομένως ίσως δεν είναι κατάλληλη η σύγκριση αριθμών μεταξύ πολύ διαφορετικών θεμάτων δικτύου. Ωστόσο, αυτό είναι ένα σημαντικό μέτρο για την κατανόηση ενός δομημένου δικτύου και μπορεί να βοηθήσει στον εντοπισμό κενών ή «τρυπών» στο γράφημα[3].

Parameters: Network Interpretation: directed
Results: Density: 0.001

HITS Metrics: Η συνάρτηση Hyperlink-Induced Topic Search (HITS) βασίζεται στην εργασία του Kleinberg και υπολογίζει δύο ξεχωριστές τιμές για κάθε κόμβο. Το πρώτο, που ονομάζεται Authority, παρέχει ένα μέτρο για το πόσο πολύτιμες πληροφορίες αποθηκεύονται από έναν συγκεκριμένο κόμβο, ενώ ο αριθμός Hub μετρά την ποιότητα των συνδέσμων προς και από

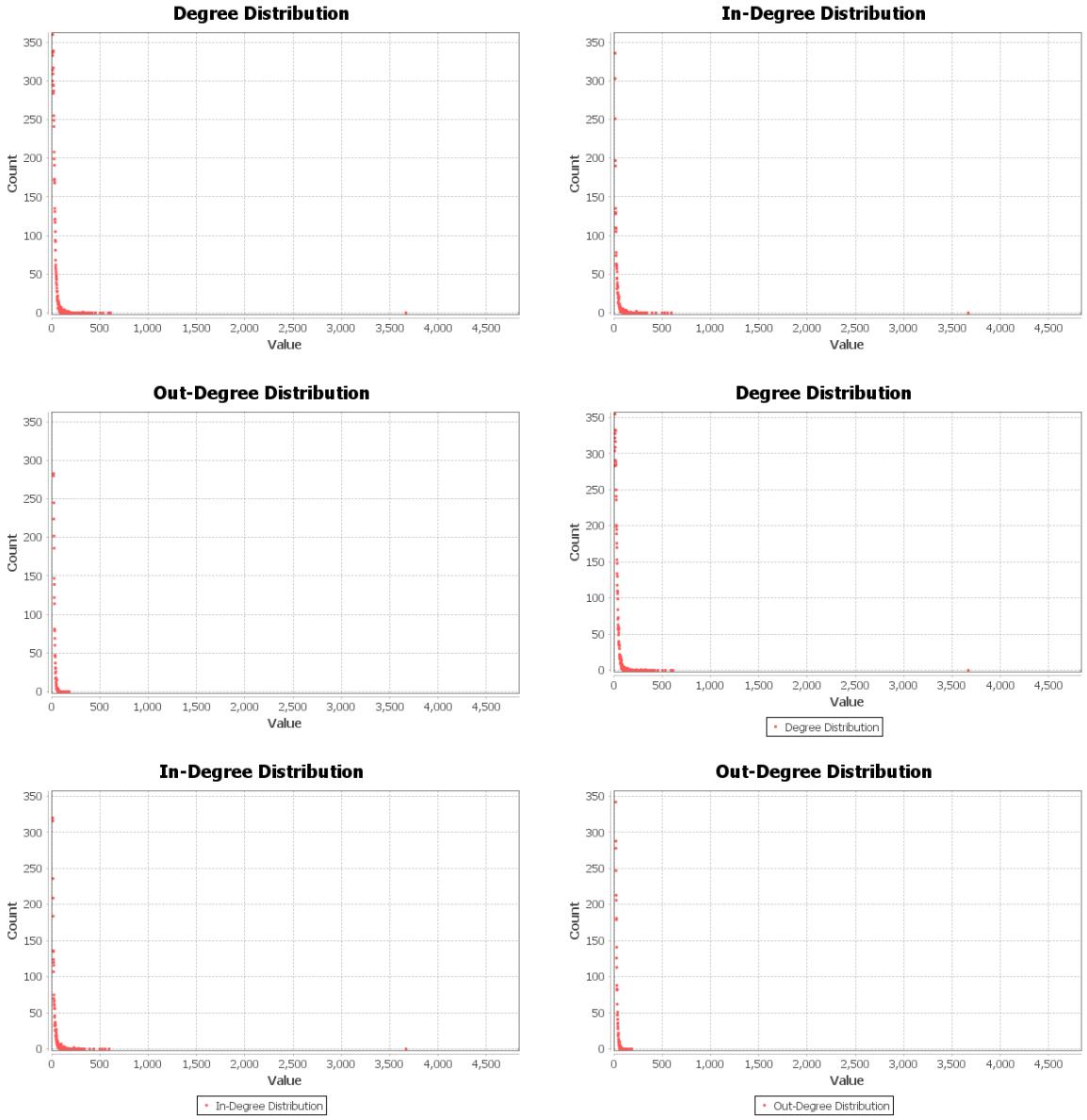
αυτόν τον συγκεκριμένο κόμβο. Αυτά τα μέτρα μπορούν να βοηθήσουν στον εντοπισμό ή την επιβεβαίωση των ρόλων που διαδραματίζουν τα κρίσιμα μέλη του δικτύου[5].
Network Diameter: Η έννοια της διαμέτρου στην ανάλυση δικτύου είναι πολύ απλή και αναφέρεται στον μέγιστο αριθμό συνδέσεων που απαιτούνται για να διασχιστεί το γράφημα. Ένας άλλος τρόπος για να γίνει αντιληπτό είναι να γνωρίζουμε πόσα βήματα χρειάζεται για να φτάσουν οι δύο πιο απομακρυσμένοι κόμβοι ο ένας τον άλλον στο δίκτυο. Στην πράξη, θα υπάρχουν περισσότερα από ένα ζεύγος κόμβων που ταιριάζουν σε αυτήν την περιγραφή, αλλά η ιδέα παραμένει η ίδια[5].

Diameter: 24 – Απόσταση από τους 2 πιο μακρινούς κόμβους
Radius: 0
Average Path length: 4.86629815391144

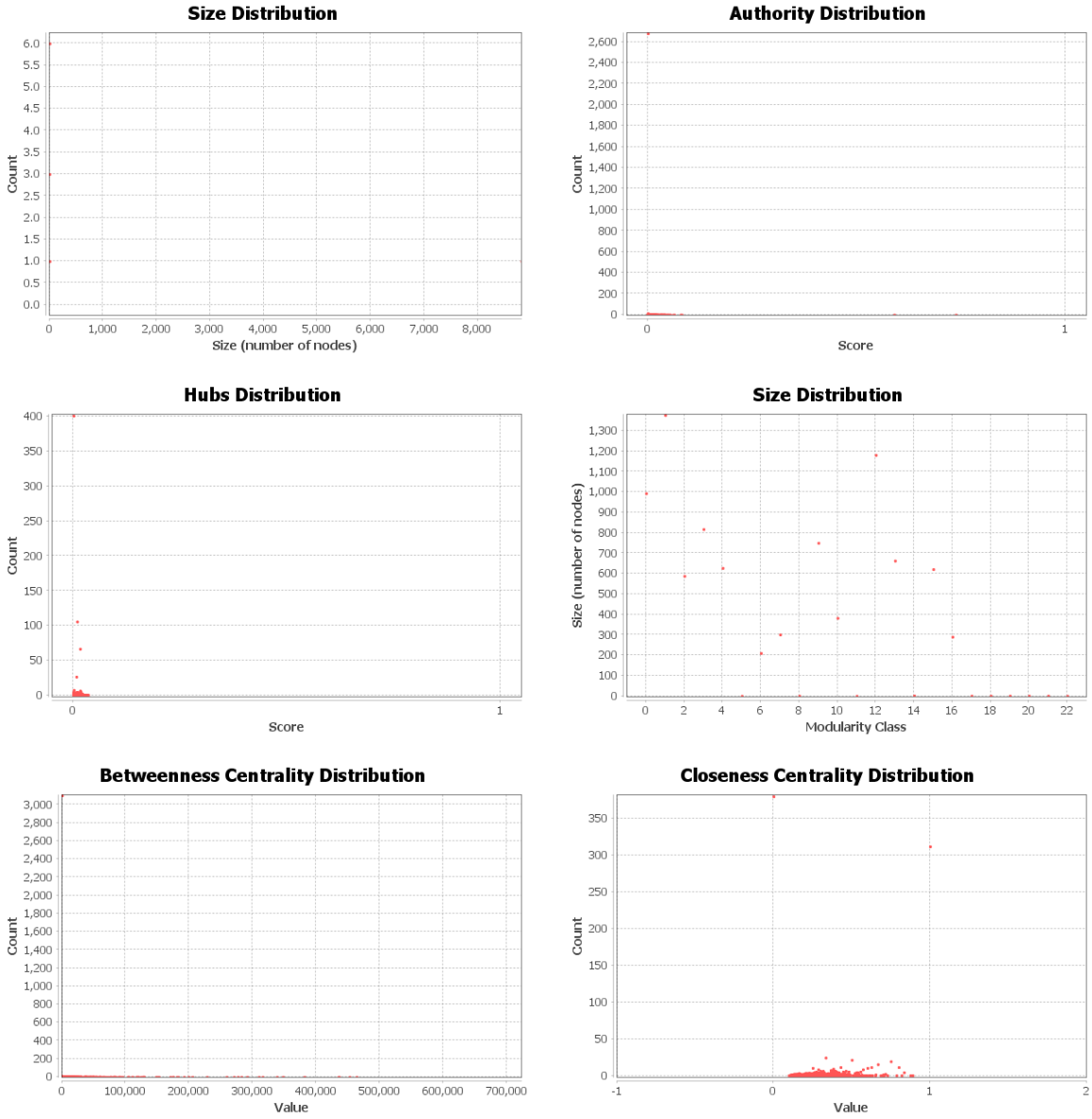
Betweenness Centrality Distribution: Η Betweenness Centrality είναι ένα μέτρο κεντρικότητας σε ένα γράφημα που βασίζεται σε συντομότερες διαδρομές. Για κάθε ζεύγος κορυφών σε ένα συνδεδεμένο γράφημα, υπάρχει τουλάχιστον μία συντομότερη διαδρομή μεταξύ των κορυφών έτσι ώστε είτε ο αριθμός των άκρων που περνά η διαδρομή (για μη σταθμισμένα γραφήματα) είτε το άθροισμα των βαρών των άκρων (για σταθμισμένα γραφήματα ελαχιστοποιείται). Η κεντρική απόσταση μεταξύ κάθε κορυφής είναι ο αριθμός αυτών των συντομότερων διαδρομών που διέρχονται από την κορυφή.
Closeness Centrality Distribution: Σε ένα συνδεδεμένο γράφημα, η Closeness Centrality (ή εγγύτητα) ενός κόμβου είναι ένα μέτρο κεντρικότητας σε ένα δίκτυο, υπολογιζόμενο ως το αμοιβαίο άθροισμα του μήκους των μικρότερων διαδρομών μεταξύ του κόμβου και όλων των άλλων κόμβων στο γράφημα. Έτσι, όσο πιο κεντρικός είναι ένας κόμβος, τόσο πιο κοντά είναι σε όλους τους άλλους κόμβους. Η κεντρικότητα της εγγύτητας αντιπροσωπεύει μια ενδιαφέρουσα περίπτωση, στην οποία ο επιλεγμένος κόμβος μπορεί στην πραγματικότητα να είναι κακώς συνδεδεμένος με μια άμεση έννοια, αλλά εξακολουθεί να έχει μεγάλη επιρροή λόγω της εγγύτητας καλώς συνδεδεμένων γειτόνων[3].
Eccentricity Centrality Distribution: Η εκκεντρότητα αναφέρεται στον αριθμό των βημάτων που απαιτούνται για τη διέλευση ενός δικτύου από έναν κόμβο. Αυτός ο αριθμός περιορίζεται από τη διάμετρο του γραφήματος. Για παράδειγμα, ένα γράφημα με διάμετρο επτά θα έχει κόμβους με μέγιστο επίπεδο εκκεντρότητας επτά. Όταν χρησιμοποιείται για τη σύγκριση κόμβων, η εκκεντρότητα μπορεί να βοηθήσει στην παροχή κάποιου περιβάλλοντος για την εκτίμηση της σχετικής θέσης και επιρροής των κόμβων μέσα σε ένα δίκτυο. Αν και δεν είναι υποκατάστατο των διαφόρων κεντρικών μέτρων, η εκκεντρότητα μπορεί ωστόσο να παρέχει κάποιες ενδείξεις σχετικά με τη σχετική σημασία των μεμονωμένων κόμβων εντός του δικτύου[3].
Harmonic Centrality Distribution: Σε ένα γράφημα (όχι απαραίτητα συνδεδεμένο), η αρμονική κεντρικότητα αντιστρέφει το άθροισμα και τις αμοιβαίες λειτουργίες στον ορισμό της κεντρικής προσέγγισης. Παρατηρείται πως η αξία κυμαίνεται λογικά μεταξύ του 0 και 1.
Clustering Coefficient Distribution: Με τον συντελεστή Clustering, το Gephi μας παρέχει τη δυνατότητα να μετρήσουμε το επίπεδο στο οποίο οι κόμβοι ομαδοποιούνται μεταξύ τους, σε αντίθεση με την ισότιμη ή τυχαία σύνδεση στο δίκτυο. Οι βαθμολογίες σε αυτό το μέτρο θα έχουν μια αντίστροφη συσχέτιση με άλλα στατιστικά στοιχεία, συμπεριλαμβανομένων πολλών από τους υπολογισμούς κεντρικότητας, ιδιαίτερα όταν μιλάμε σε παγκόσμιο επίπεδο (ολόκληρο το γράφημα). Μπορούμε επίσης να μετρήσουμε αυτό το στατιστικό σε τοπικό επίπεδο, για να κατανοήσουμε την επίδραση ενός μόνο κόμβου στη γειτονιά του[5].
Eigenvector Centrality Distribution: Η κεντρικότητα του Eigenvector (ονομάζεται επίσης Eigencentrality) είναι ένα μέτρο της επιρροής ενός κόμβου σε ένα δίκτυο. Εκχωρεί σχετικές βαθμολογίες σε όλους τους κόμβους του δικτύου με βάση την ιδέα ότι οι συνδέσεις με κόμβους υψηλής βαθμολογίας συμβάλλουν περισσότερο στην βαθμολογία του εν λόγω κόμβου από τις ίσες συνδέσεις με κόμβους χαμηλής βαθμολογίας. Παρατηρείται πως λίγοι κόμβοι ασκούν επιρροή στο δίκτυο που είναι και οι παλαιότεροι, καθώς αυτοί αναφέρονται στα νεότερα papers. Όταν οι κόμβοι συνδέονται σε μεγάλο βαθμό με άλλους κόμβους με υψηλά επίπεδα επιρροής, το αποτέλεσμα θα είναι ένα υψηλό επίπεδο κεντρικότητας του Eigenvector. Σε αυτήν την περίπτωση, δεν είναι απλώς η σύνδεση με πολλούς άλλους κόμβους που είναι κρίσιμης σημασίας, αλλά η σύνδεση με τους πιο σημαντικούς κόμβους είναι υψίστης σημασίας[3].

Average Length Path: Το μέσο μήκος μονοπατιού μεταξύ όλων των ζευγαριών των κόμβων. Συνδεδεμένοι κόμβοι έχουν μήκος γράφου ένα.

2.4 Γραφήματα δικτύου



Εικόνα 27. Κατανομή Κεντρικότητας Βαθμού (Με και χωρίς βάρος)



Εικόνα 28. Κατανομές Μεγέθους, μετρικών HITS, Κεντρικότητας Ενδιαμεσότητας, Κεντρικότητας Εγγύτητας

Εικόνα 29. Κατανομή μετρικών ακμών δικτύου

3. Τεχνητό Δίκτυο

3.1 Αλγόριθμοι δημιουργίας Τεχνητού Δικτύου

Με την δημιουργία των αλγορίθμων, μεταφράζοντας τις οδηγίες τους σε γλώσσα python, πραγματοποιήθηκαν δοκιμές δημιουργίας του τεχνητού δικτύου ώστε να κατοπτρίζει τα αποτελέσματα των μετρήσεων αλλά και της δομής με το αληθινό δίκτυο συν-συγγραφέων. Με βάση την διαδικασία προσέγγισης του πραγματικού δικτύου, μέσω των αλγόριθμων γένεσης τεχνητών δικτύων καταφέραμε να αποκαλύψουμε τον μηχανισμό δημιουργίας του πραγματικού δικτύου. Συγκεκριμένα, ρυθμίζοντας τις εσωτερικές παραμέτρους του εκάστοτε αλγόριθμου προκειμένου οι μακροσκοπικές μετρικές ομοιότητας των δύο δικτύων να μεγιστοποιηθούν προσδιορίσαμε τις τιμές αυτών των παραμέτρων για το πραγματικό δίκτυο.

Επιπλέον προσδιορίσαμε το **Kullback–Leibler divergence**, μεταξύ του πραγματικού δικτύου και του τεχνητού για τον κάθε αλγόριθμο. Σκοπός για την εύρεση της απόκλισης των δικτύων είναι να προσδιοριστεί η μικρότερη απόκλιση, συνεπώς ο καλύτερος αλγόριθμος που δημιουργεί το δίκτυο με την μεγαλύτερη ομοιότητα ως προς τις μετρικές.

Οι αλγόριθμοι που χρησιμοποιήθηκαν και οι οδηγίες τους ανά κατηγορία:

Κατηγορία: Μοντέλα εξέλιξης δυναμικών δικτύων (Δυναμικά NEMs). Τρία μοντέλα βασισμένα σε τριαδική κλειστότητα και οικουμενικές συνδέσεις.

Παράμετροι	Μηχανισμοί
DEB (Davidsen et al., 2002) 2 free, N,p	I. Επέλεξε ένα κόμβο i τυχαία, και (a) Αν ο i έχει λιγότερες από 2 ακμές, σύνδεσέ τον σε έναν τυχαίο κόμβο (b) αλλιώς, επέλεξε 2 γείτονες του i και σύνδεσέ τους αν δεν ήταν ήδη συνδεδεμένοι II. Επέλεξε έναν τυχαίο κόμβο με πιθανότητα p και διέγραψε όλες τις συνδέσεις του.
MVS (Marsili et al., 2004) 3 free, N,ξ, η, $(\lambda = 0.001)$	I. Επέλεξε ένα κόμβο i τυχαία, και (a) Σύνδεσε τον i σε έναν τυχαίο κόμβο με πιθανότητα η. (b) Επέλεξε τον φίλο του φίλου του i (ομοιόμορφη τυχαία αναζήτηση) με πιθανότητα ξ και σύνδεσέ τον στον i αν δεν είναι ήδη συνδεδεμένοι. II. Επέλεξε μία τυχαία σύνδεση και διέγραψέ την με πιθανότητα λ
KOSKK (Kumpula et al., 2007) 3 free, N, p_Δ, p_r $(\mathrm{w}_0 = 1, \mathrm{p}_d = 0.001, \delta = 0.5)$	I. Επέλεξε ένα κόμβο i τυχαία, και (a) Επέλεξε τον φίλου του φίλου του k (μέσω αναζήτησης βάρους) και σύνδεσέ τον με τον i με πιθανότητα p_Δ (με αρχική δύναμη σύνδεσης w_0) αν δεν είναι ήδη συνδεδεμένοι. Αύξησε την δύναμη της σύνδεσης κατά δ κατά μήκος του μονοπατιού αναζήτησης, όπως και στην ακμή l_{ik}, αν ήταν ήδη παρούσα. (b) Επιπλέον, με πιθανότητα p_r (ή με πιθανότητα 1 αν ο i δεν έχει συνδέσεις), σύνδεσε τον i σε έναν τυχαίο κόμβο j (με δύναμη σύνδεσης w_0). II. Επέλεξε έναν τυχαίο κόμβο και με πιθανότητα p_d διέγραψε όλες τις συνδέσεις του.

Πίνακας 1. Μοντέλα Δυναμικών NEMs

Κατηγορία: Μοντέλα εξέλιξης αυξανόμενων δικτύων (αυξανόμενα NEMs). Δύο μοντέλα βασισμένα σε τριαδική κλειστότητα και οικουμενικές συνδέσεις.

Παράμετροι	Μηχανισμοί
TOSHK (Toivonen et al., 2006) 3 free, N, p, k (simplified)	I. Πρόσθεσε ένα νέο κόμβο i στο δίκτυο, συνδέοντας το σε μία τυχαίο αρχική επαφή με πιθανότητα p, ή 2 επαφές με πιθανότητα $1-p$. II. Για κάθε τυχαία αρχική επαφή j, επέλεξε έναν αριθμό m_{sec} από δευτερεύουσες συνδέσεις από την κατανομή $U[0,k]$ και σύνδεσε το i με τους γείτονες m_{sec} του j αν είναι διαθέσιμοι.
Váz (Vázquez, 2003) 2 free, N,u	I. Με πιθανότητα $1-u$, πρόσθεσε έναν νέο κόμβο στο δίκτυο, συνδέοντας το σε έναν τυχαίο κόμβο i. Οι πιθανές ακμές που θα δημιουργηθούν ανάμεσα στον νέο κόμβο n και τους γείτονες j του i (μία πιθανή ακμή σημαίνει ότι ο n και ο j έχουν κοινό γείτονα, i, αλλά όχι απευθείας σύνδεση μεταξύ τους). II. Με πιθανότητα u, μετάτρεψε μία τέτοια πιθανή ακμή που δημιουργήθηκε κατά το προηγούμενο βήμα σε μία ακμή. Οι πιθανές ακμές που δημιουργήθηκαν μέσω της μετατροπής μίας ακμής αγνοούνται.

Πίνακας 2. Μοντέλα αυξανόμενων NEMs

Τελικώς δημιουργήθηκε ένας κατευθυνόμενος άκυκλος γράφος (DAG model) με την εξής οδηγία:

Το μοντέλο λαμβάνει ως είσοδο μια ταξινομημένη ακολουθία βαθμού που αποτελείται από τον εισερχόμενο βαθμό (in-degree) K_i^{in} και τον εξερχόμενο βαθμό (out-degree) K_i^{out}i για κάθε κορυφή $i = 1 \dots n$, όπου n είναι ο συνολικός αριθμός κορυφών στο δίκτυο. Οι κατευθυνόμενες ακμές στο μοντέλο επιτρέπεται να πηγαίνουν μόνο από κορυφές με μεγαλύτερους δείκτες σε κορυφές με μικρότερους δείκτες και αυτός ο περιορισμός επιβάλλει την ακυκλική φύση του δικτύου. Έτσι, μπορούμε να έχουμε μια ακμή που πηγαίνει στην κορυφή i από την κορυφή j μόνο αν $i < j$.

Με την χρήση του τελευταίου αλγόριθμου, επιτεύχθηκε η δημιουργία ενός τεχνητού δικτύου, παρόμοιο με το αληθινό, με ισόνομες μετρήσεις.

3.2 Ανάλυση Κώδικα Αλγορίθμων

3.2.1 DEB (Davidsen et al., 2002)

```python
import random


# Function to introduce a new node to the network
def introduce_node(network, i):
    # If node i has fewer than two ties, connect it to a random node
    if len(network[i]) < 2:
        j = random.choice(list(network.keys()))
        while j == i or j in network[i]:
            j = random.choice(list(network.keys()))
        network[i].add(j)
        network[j].add(i)
    # Otherwise, pick two neighbors of i and connect them if they are not already connected
    else:
        neighbors = list(network[i])
        random.shuffle(neighbors)
        for j in neighbors[:2]:
            for k in neighbors[:2]:
                if j != k and k not in network[j]:
                    network[j].add(k)
                    network[k].add(j)

```

Εικόνα 30. Κώδικας αλγορίθμου DEB (1)

Αρχικά δημιουργείται μία συνάρτηση για την εισαγωγή ενός νέου κόμβου στο δίκτυο. Στην συνέχεια εκτελούνται τα βήματα του αλγόριθμου. Ειδικότερα, αν ο κόμβος i έχει λιγότερες από 2 συνδέσεις, συνδέεται με έναν τυχαίο κόμβο, αλλιώς επιλέγονται 2 γείτονες του i και συνδέονται μεταξύ τους αν δεν είναι ήδη συνδεδεμένοι.

```python
# Function to remove all ties of a random node with probability p
def remove_node_ties(network, p):
    # Select a random node from the network
    i = random.choice(list(network.keys()))
    # Remove all of its ties with probability p
    if random.random() < p:
        for j in network[i]:
            network[j].remove(i)
        del network[i]
```

Εικόνα 31. Κώδικας αλγορίθμου DEB (2)

Στην συνέχεια, δημιουργείται μία συνάρτηση για την διαγραφή όλων των συνδέσεων ενός τυχαίου κόμβου με πιθανότητα p. Επιλέγεται ο τυχαίος κόμβος από το δίκτυο και διαγράφονται οι συνδέσεις του.

```python
# Function to write the network to a file
def write_network_to_file(network, filename):
    with open(filename, 'w') as f:
        f.write(f"Source Target\n")
        for i in network:
            for j in network[i]:
                f.write(f"{i} {j}\n")


# Initialize the network as an empty dictionary
network = {}

# Add the first node to the network
network[0] = set()

# Loop through all nodes from 1 to 12999
for i in range(1, 8830):
    # Randomly select a node to connect to
    j = random.randint(0, i - 1)

    # Add node i to the network and introduce it to a random node or two neighbors of i
    network[i] = set()
    introduce_node(network, i)

    # Remove ties of a random node with probability p=0.05
    remove_node_ties(network, 0.05)

# Print the resulting network
print(network)

# Write the resulting network to a file
write_network_to_file(network, "network_1a.csv")
```

Εικόνα 32. Κώδικας αλγορίθμου DEB (3)

Έπειτα, δημιουργείται η συνάρτηση για την εγγραφή του δικτύου σε ένα νέο αρχείο κειμένου. Αρχικά, δημιουργείται το δίκτυο σε ένα κενό dictionary, προσθέτοντας τον πρώτο κόμβο. Μέσω ενός for loop με τους κόμβους του πραγματικού δικτύου, δημιουργούμε το νέο τεχνητό δίκτυο, διαγράφοντας τυχαία συνδέσεις κόμβων με πιθανότητα p.

```python
while adef_p_init < 0.9990:
    a_def(adef_p_init)
    print(adef_p_init)
    adef_p_list.append(adef_p_init)
    degree_dist1 = read_data("network_1a.csv", False)
    degree_prob_dist.append(degree_dist1)
    if len(degree_dist2) > len(degree_dist1):
        zeros_to_add = len(degree_dist2) - len(degree_dist1)
        zeros = np.zeros(zeros_to_add, dtype=degree_dist2.dtype)
    degree_dist1 = np.concatenate([degree_dist1, zeros])
    # compute the KL divergence between the two degree distributions
    kl_divergence = entropy(degree_dist1, degree_dist2)
    kl_list.append(kl_divergence)
    print(kl_divergence)
    adef_p_init = round(adef_p_init + adef_p_step, 6)
```

Εικόνα 33. Κώδικας αλγορίθμου DEB (4)

Για κάθε loop όπου η παράμετρος του αλγορίθμου είναι μικρότερη από 0.999, με βήμα 0.00001 (από την αρχική θέση 0.0001), προσδιορίστηκε η μικρότερη τιμή της KL απόκλισης για τον αλγόριθμο.

Min KL Απόκλιση	adef_p Παράμετρος
1.9362336978472388	0.0002

Πίνακας 3. Ελάχιστη απόκλιση KL και παράμετρος adef_p

3.2.2 MVS (Marsili et al., 2004)

```python
import csv
import random

# define the parameters
n_nodes = 8830
h_prob = 0.3
x_prob = 0.3
l_prob = 0.00000000001


# initialize the list of nodes
nodes = [[] for _ in range(n_nodes)]


# define helper functions
def connect(i, j):
    nodes[i].append(j)
    nodes[j].append(i)


def introduce(i):
    friends = nodes[i]
    if len(friends) > 0:
        friend = random.choice(friends)
        friends_of_friend = nodes[friend]
        if len(friends_of_friend) > 0:
            new_friend = random.choice(friends_of_friend)
            if new_friend not in friends:
                connect(i, new_friend)


def delete_connection():
    i = random.randint(0, n_nodes - 1)
    if len(nodes[i]) > 0:
        j = random.choice(nodes[i])
        if random.random() < l_prob:
            nodes[i].remove(j)
            nodes[j].remove(i)
```

Εικόνα 30. Κώδικας αλγορίθμου MVS (1)

Στην αρχή, ορίζουμε τις αρχικές τιμές των παραμέτρων, και δημιουργούμε ορισμένες συναρτήσεις βοήθειας για την εκτέλεση του συγκεκριμένου αλγορίθμου.

```python
# run the algorithm for a certain number of steps
for step in range(100000):
    # select a node i randomly
    i = random.randint(0, n_nodes - 1)
    # connect i to another random node with probability h_prob
    if random.random() < h_prob:
        j = random.randint(0, n_nodes - 1)
        if i != j:
            connect(i, j)
    # select a friend's friend of i (by uniformly random search) with probability x_prob
    if random.random() < x_prob:
        introduce(i)
    # select a random tie and delete it with probability l_prob
    delete_connection()

# write the resulting connections to a CSV file
with open('network_1b.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=' ')
    writer.writerow(['Source', 'Target'])
    for i in range(n_nodes):
        for j in nodes[i]:
            if i < j:
                writer.writerow([i, j])
```

Εικόνα 31. Κώδικας αλγορίθμου MVS (2)

Εκτελούμε τον αλγόριθμο για έναν συγκεκριμένο αριθμό βημάτων, με το να επιλέγουμε τυχαία έναν κόμβο i, να τον συνδέοουμε με έναν τυχαίο κόμβο με πιθανότητα h_prob, να συνδέοουμε τους φίλους των φίλων του i μέσω τυχαίας αναζήτησης αλλά και να διαγράφουμε συνδέσεις με πιθανότητα l_prob. Τέλος γράφουμε το δίκτυο σε ένα αρχείο κειμένου.

```
while l_prob < 1:
    while hmax_prob > h1:
        while xmax_prob > x1:
            b_def(n_nodes, h1, x1, l_prob)
            h1_list.append(h1)
            x1_list.append(x1)
            l_list.append(l_prob)
            print(h1, x1, l_prob)
            degree_dist1 = read_data("network_1b.csv", False)
            # np.array2string(read_data("network_1b.csv")[0], separator=', ', formatter={'float_kind': lambda x:
            # "%.10f" % x})
            degree_prob_dist.append(degree_dist1)
            if len(degree_dist2) > len(degree_dist1):
                zeros_to_add = len(degree_dist2) - len(degree_dist1)
                zeros = np.zeros(zeros_to_add, dtype=degree_dist2.dtype)
                degree_dist1 = np.concatenate([degree_dist1, zeros])
            # compute the KL divergence between the two degree distributions
            kl_divergence = entropy(degree_dist1, degree_dist2)
            kl_list.append(kl_divergence)
            x1 = round(x1 + d_x_prob_b, 2)
        h1 = round(h1 + d_h_prob_b, 2)
        x1 = 0.1
    l_prob = round(l_prob * 10, 15)
    h1 = 0.1
    x1 = 0.1
```

Εικόνα 32. Κώδικας αλγορίθμου MVS (3)

Για τις παραμέτρους l_prob, h_prob και x_prob βρίσκουμε τις optimal τιμές για την ελάχιστη απόκλιση KL και συνεπώς την μεγαλύτερη ομοιότητα στις μετρικές με το αρχικό δίκτυο.

Min KL Απόκλιση	l_prob Παράμετρος	h_prob Παράμετρος	x_prob Παράμετρος
0.33982751960657287	0.1	0.29	0.97

Πίνακας 4. Ελάχιστη απόκλιση KL και παράμετροι l_prob, h_prob και x_prob

3.2.3 KOSKK (Kumpula et al., 2007)

```
import random
import csv

# Parameters
pD = 0.1
pR = 0.05
w0 = 1
d = 0.5
pF = 0.0001
num_nodes = 13000

# Initialize network
network = {i: set() for i in range(num_nodes)}

```

Εικόνα 33. Κώδικας αλγορίθμου KOSKK (1)

Ορίζουμε τις παραμέτρους και κάνουμε initialize το τεχνητό δίκτυο.

```python
# Algorithm
for _ in range(num_nodes):
    # Select a node i randomly
    i = random.randint(0, num_nodes - 1)

    # (a) select a friend's friend k (by weighted search)
    if network[i]:
        friend = random.choice(list(network[i]))
        if network[friend]:
            k = random.choices(list(network[friend]), weights=[len(network[n]) for n in network[friend]])[0]

            # introduce k to i with probability pD (with initial tie strength w0) if not already acquainted
            if k not in network[i] and random.random() < pD:
                network[i].add(k)
                network[k].add(i)
                # Increase tie strengths by d along the search path, as well as on the link l if it was already present
                curr_node = friend
                while curr_node != k:
                    network[curr_node].add(k)
                    network[k].add(curr_node)
                    curr_node = random.choice(list(network[curr_node]))
                for j in network[i]:
                    network[j].add(k)
                    network[k].add(j)

    # (b) connect i to a random node j with probability pR (or with probability 1 if i has no connections)
    if not network[i] or random.random() < pR:
        j = random.randint(0, num_nodes - 1)
        if i != j and j not in network[i]:
            network[i].add(j)
            network[j].add(i)

    # (II) Select a random node and with probability pF remove all of its ties.
    if random.random() < pF:
        node_to_remove = random.choice(list(network.keys()))
        for j in list(network[node_to_remove]):
            network[j].remove(node_to_remove)
        network[node_to_remove] = set()

```

Εικόνα 34. Κώδικας αλγορίθμου KOSKK (2)

Επιλέγουμε τυχαία έναν κόμβο i. Έπειτα, επιλέγουμε τον φίλο του φίλου k μέσω αναζήτησης βάρους. Τον συνδέουμε με τον κόμβο i με πιθανότητα pD αν δεν είναι ήδη συνδεδεμένοι, αυξάνοντας την δύναμη της σύνδεσης κατά d. Συνδέουμε τον i με έναν τυχαίο κόμβο j με πιθανότητα pR ή με πιθανότητα 1 αν ο i δεν έχει συνδέσεις. Τέλος επιλέγουμε έναν τυχαίο κόμβο και με πιθανότητα pF του αφαιρούμε όλες τις συνδέσεις.

```python
# Write network to CSV file
with open('network_1c.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=' ')
    writer.writerow(['Source', 'Target'])
    for i in range(num_nodes):
        for j in network[i]:
            writer.writerow([i, j])
```

Εικόνα 35. Κώδικας αλγορίθμου KOSKK (3)

Γράφουμε το νέο τεχνητό δίκτυο σε αρχείο κειμένου.

```
while pF <= 1:
    while d <= 1:
        while w0 <= 1:
            while pD <= 1:
                while pR <= 1:
                    c_def(pD, pR, w0, d, pF, num_nodes)
                    pD_list.append(pD)
                    pR_list.append(pR)
                    w0_list.append(w0)
                    d_list.append(d)
                    pF_list.append(pF)
                    print(pD, pR, w0, d, pF)
                    degree_dist1 = read_data("network_1c.csv", False)
                    degree_prob_dist.append(degree_dist1)
                    if len(degree_dist2) > len(degree_dist1):
                        zeros_to_add = len(degree_dist2) - len(degree_dist1)
                        zeros = np.zeros(zeros_to_add, dtype=degree_dist2.dtype)
                        degree_dist1 = np.concatenate([degree_dist1, zeros])
                    # compute the KL divergence between the two degree distributions
                    kl_divergence = entropy(degree_dist1, degree_dist2)
                    print("KL:", kl_divergence)
                    kl_list.append(kl_divergence)
                    pR = round(pR + pR_step, 4)
                pD = round(pD + pD_step, 4)
                pR = 0.05
            w0 = round(w0 + w0_step, 4)
            pD = 0.1
            pR = 0.05
        d = round(d + d_step, 4)
        w0 = 0.1
        pD = 0.1
        pR = 0.05
    pF = round(pF * 10, 6)
    d = 0.1
    w0 = 0.1
    pD = 0.1
    pR = 0.05
```

Εικόνα 40. Κώδικας αλγορίθμου KOSKK (4)

Για κάθε μία από τις παραμέτρους του αλγόριθμου, δημιουργείτε ένα loop για εξέταση όλων των τιμών. Για κάθε ένα που φτάνει στο προσδιορισμένο όριο, οι προηγούμενες παράμετροι αρχικοποιούνται.

Τελικώς, για την ελάχιστη απόκλιση KL και τις παραμέτρους pF, pD, pR, w0 και d παρατηρούμε:

Min KL Απόκλιση	pF Παράμετρος	pD Παράμετρος	pR Παράμετρος	w0 Παράμετρος	d Παράμετρος
0.961049684406875	0.0001	1.0	1.0	0.4	0.7

Πίνακας 5. Ελάχιστη απόκλιση KL και παράμετροι pF, pD, pR, w0 και d

3.2.4 TOSHK (Toivonen et al., 2006)

```python
import random
import csv

# define parameters
p = 0.5   # probability of connecting to one initial contact
k = 10    # maximum number of secondary connections

# initialize the network
network = {0: []}   # the first node has no connections yet

```

Εικόνα 41. Κώδικας αλγορίθμου TOSHK (1)

Ορίζουμε τις παραμέτρους και κάνουμε initialize το τεχνητό δίκτυο, προσθέτοντας τον πρώτο κόμβο, ο οποίος δεν έχει ακόμα συνδέσεις.

```python
# add new nodes and their connections
for i in range(1, 13000):
    initial_contacts = []
    if random.random() < p:
        initial_contacts.append(random.choice(list(network.keys())))
    else:
        initial_contacts = random.choices(list(network.keys()), k=2)
    network[i] = []
    for j in initial_contacts:
        msec = random.randint(0, k)
        available_neighbors = list(set(network.keys()) - set(network[j]) - {i})
        selected_neighbors = random.sample(available_neighbors, min(msec, len(available_neighbors)))
        network[i].extend(selected_neighbors)
        network[j].append(i)
```

Εικόνα 42. Κώδικας αλγορίθμου TOSHK (2)

Ακολουθώντας τα βήματα του αλγορίθμου, προσθέτουμε νέους κόμβους και τις συνδέσεις μεταξύ τους.

```python
# write the network to a csv file
with open('network_2a.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile, delimiter=' ')
    writer.writerow(['Source', 'Target'])
    for i in network:
        for j in network[i]:
            writer.writerow([i, j])

```

Εικόνα 36. Κώδικας αλγορίθμου TOSHK (3)

Τέλος, γράφουμε το νέο τεχνητό δίκτυο σε αρχείο κειμένου.

```
while k_start < 50:
    while p_start < 1:
        a_def_2(p_start, k_start)
        print(p_start, k_start)
        p_list.append(p_start)
        k_list.append(k_start)
        degree_dist1 = read_data("network_2a.csv", False)
        degree_prob_dist.append(degree_dist1)
        if len(degree_dist2) > len(degree_dist1):
            zeros_to_add = len(degree_dist2) - len(degree_dist1)
            zeros = np.zeros(zeros_to_add, dtype=degree_dist2.dtype)
        degree_dist1 = np.concatenate([degree_dist1, zeros])
        # compute the KL divergence between the two degree distributions
        kl_divergence = entropy(degree_dist1, degree_dist2)
        print("KL:", kl_divergence)
        kl_list.append(kl_divergence)
        p_start = round(p_start + p_step, 4)
    k_start = k_start + k_step
    p_start = 0.1
```

Εικόνα 37. Κώδικας αλγορίθμου TOSHK (4)

Για τις παραμέτρους p και k δημιουργείται ένα loop με αποτέλεσμα την εύρεση της ελάχιστης τιμής της απόκλισης KL.

Min KL Απόκλιση	p Παράμετρος	k Παράμετρος
0.12498725521239953	0.7	12

Πίνακας 6. Ελάχιστη απόκλιση KL και παράμετροι k και p

3.2.5 Váz (Vázquez, 2003)

```
import random
import csv


class Node:
    def __init__(self, id):
        self.id = id
        self.neighbors = set()

    def add_neighbor(self, node):
        self.neighbors.add(node)
        node.neighbors.add(self)

```

Εικόνα 38. Κώδικας αλγορίθμου Váz (1)

Δημιουργούμε μία κλάση για τους κόμβους και τους γείτονές τους.

```python
class Network:
    def __init__(self, u):
        self.nodes = []
        self.u = u

    def add_node(self):
        new_node = Node(len(self.nodes))
        self.nodes.append(new_node)
        if len(self.nodes) > 1:
            # With probability 1 - u, connect the new node to a random node i.
            i = random.choice(self.nodes[:-1])
            i.add_neighbor(new_node)
            # Create potential edges between the new node and i's neighbors.
            for neighbor in i.neighbors:
                if new_node not in neighbor.neighbors and new_node != neighbor:
                    neighbor.add_neighbor(new_node)
                    # With probability u, convert potential edge to edge.
                    if random.random() < self.u:
                        new_node.add_neighbor(neighbor)

    def get_adjacency_list(self):
        adjacency_list = {}
        for node in self.nodes:
            adjacency_list[node.id] = [neighbor.id for neighbor in node.neighbors]
        return adjacency_list

    def write_to_csv(self, filename):
        with open(filename, 'w', newline='') as csvfile:
            writer = csv.writer(csvfile, delimiter=' ')
            writer.writerow(['Source', 'Target'])
            for node in self.nodes:
                for neighbor in node.neighbors:
                    if neighbor.id > node.id:
                        writer.writerow([node.id, neighbor.id])

```

Εικόνα 39. Κώδικας αλγορίθμου Váz (2)

Έπειτα, δημιουργούμε την κλάση για το δίκτυο και εκτελούμε τον αλγόριθμο, όπου πραγματοποιούνται οι συνδέσεις μεταξύ των νέων κόμβων και των γειτόνων τους. Τέλος, το τεχνητό δίκτυο, γράφεται σε αρχείο κειμένου.

```python
network = Network(u=0.5)
for i in range(13000):
    network.add_node()

adjacency_list = network.get_adjacency_list()
print(adjacency_list)
network.write_to_csv('network_2b.csv')
```

Εικόνα 40. Κώδικας αλγορίθμου Váz (3)

Με σκοπό τον κατοπτρισμό του τεχνητού δικτύου με το αληθινό (actual), τρέχουμε τον αλγόριθμο για 13000 κόμβους.

Οι τιμές της KL divergence του συγκεκριμένου αλγορίθμου έτειναν προς το άπειρο συνεπώς μπορούμε να απορρίψουμε την σχέση ομοιότητας μεταξύ του τεχνητού δικτύου, αποτέλεσμα γένεσης του αλγορίθμου, με το πραγματικό δίκτυο.

3.2.6 DAG Model

```python
import csv


def create_network(degree_sequences, filename):
    n = len(degree_sequences)
    out_degrees = [degree[1] for degree in degree_sequences]
    in_degrees = [degree[0] for degree in degree_sequences]

    # create the network
    network = []
    for i in range(n):
        for j in range(i + 1, n):
            if in_degrees[i] > 0 and out_degrees[j] > 0:
                network.append((i + 1, j + 1))
                in_degrees[i] -= 1
                out_degrees[j] -= 1

    # write the network to a csv file
    with open(filename, 'w') as csvfile:
        writer = csv.writer(csvfile, delimiter=' ')
        writer.writerow(['Source', 'Target'])
        for connection in network:
            writer.writerow(list(connection))

```

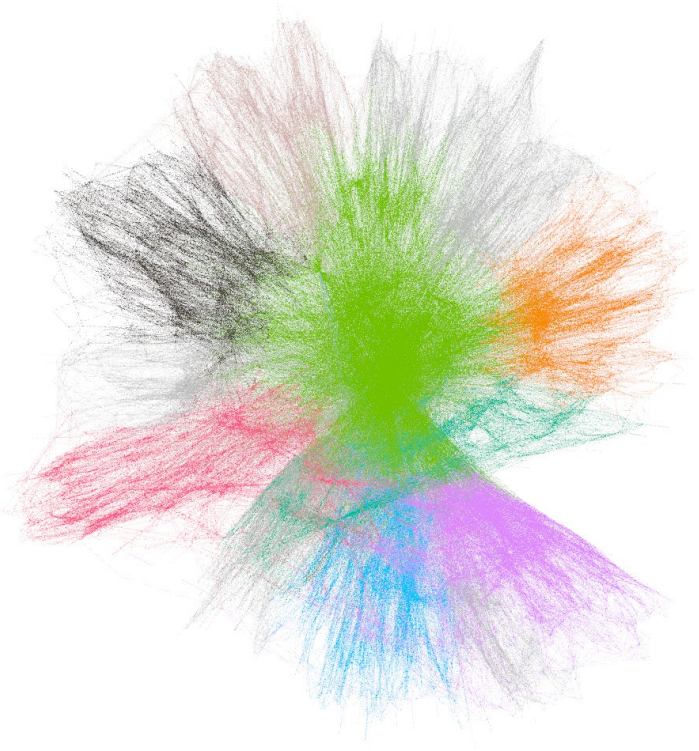
Εικόνα 41. Κώδικας αλγορίθμου DAG (1)

Με βάση τα in-degrees του πραγματικού δικτύου, δημιουργούμε το τεχνητό εκτελώντας τον αλγόριθμο DAG (Directional Acyclic Graph), γράφοντας το νέο τεχνητό δίκτυο σε αρχείο κειμένου.

```python
df = pd.read_csv('network_actual.csv', delimiter=' ')
# print(df.head())
g = nx.from_pandas_edgelist(df, source='Source', target='Target', create_using=nx.DiGraph)
# nx.draw(g)

nodes = g.number_of_nodes()
# compute the degree distribution
degrees = [g.degree(n) for n in g.nodes()]

# get the in-degree and out-degree of each node as tuples
in_out_degrees = [(g.in_degree(node), g.out_degree(node)) for node in g.nodes()]
```

Εικόνα 42. Κώδικας αλγορίθμου DAG (2)

Στην συγκεκριμένη περίπτωση του αλγορίθμου, πραγματοποιήθηκε η χρήση των in-degree και out-degree μετρικών κόμβων του πραγματικού δικτύου, για να δημιουργηθεί το τεχνητό.



Εικόνα 50. Οπτικό αποτέλεσμα του τεχνητού δικτύου, με χρωματισμένες γειτονίες

3.3 Ανάλυση Μετρικών Τεχνητού Δικτύου

Μέση βαθμίδα – Average Degree : 10.845
Μέση βαθμίδα βάρους – Average Weighted Degree: 10.845

Συνδεδεμένα Στοιχεία

Number of Weakly Connected Components: 1
Number of Strongly Connected Components: 8780

Πυκνότητα γραφήματος

Parameters: Network Interpretation: directed
Results: Density: 0.001

Διάμετρος Δικτύου

Diameter: 9 – Απόσταση από τους 2 πιο μακρινούς κόμβους
Radius: 0
Average Path length: 3.6264032345334267

Μέσος Συντελεστής Ομαδοποίησης

Average Clustering Coefficient: 0.043

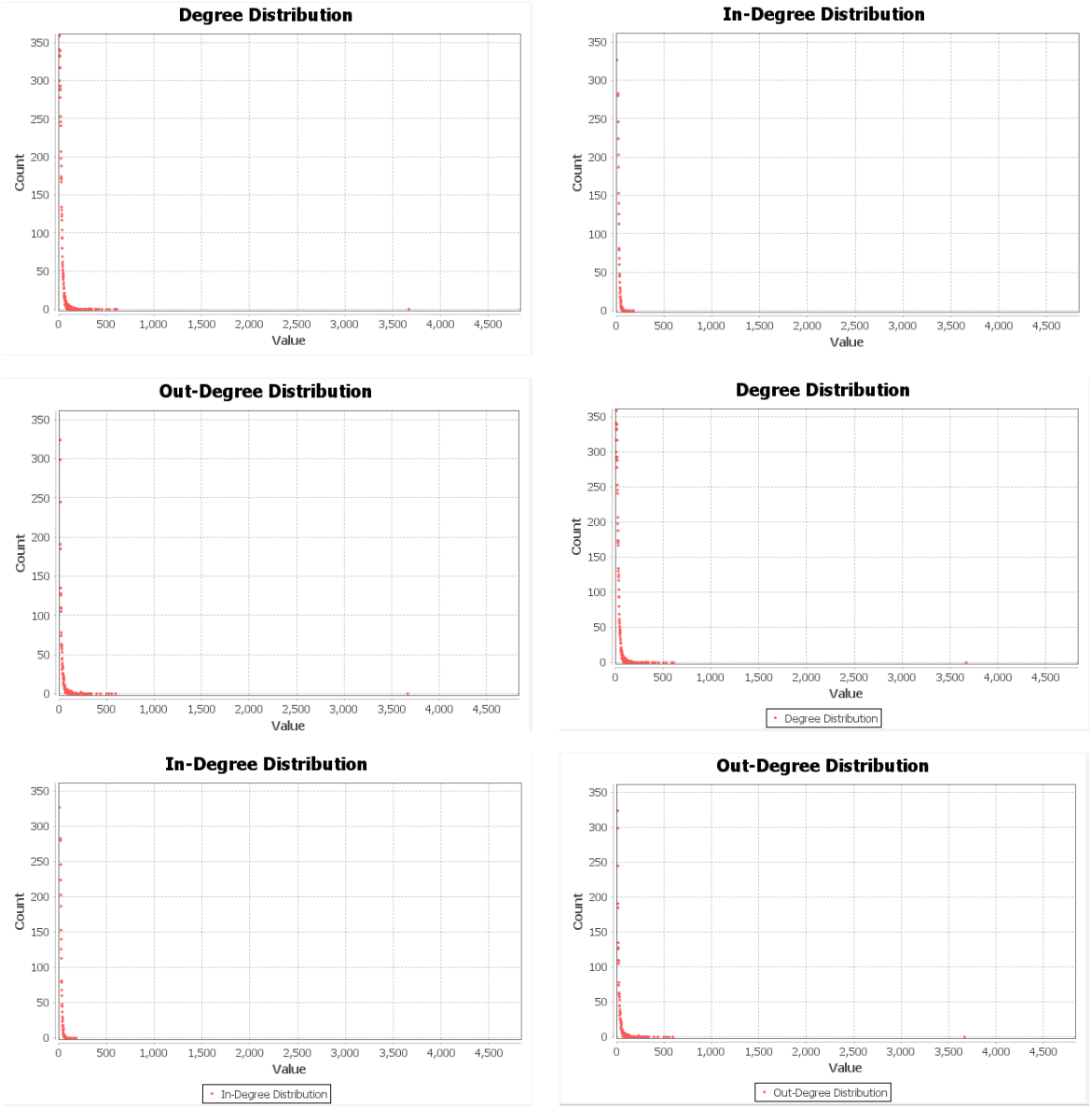
EigenVector Centralities

Network Interpretation: directed
Number of iterations: 100
Sum change: 0.4290879222910288

Μέσο Μήκος Μονοπατιού

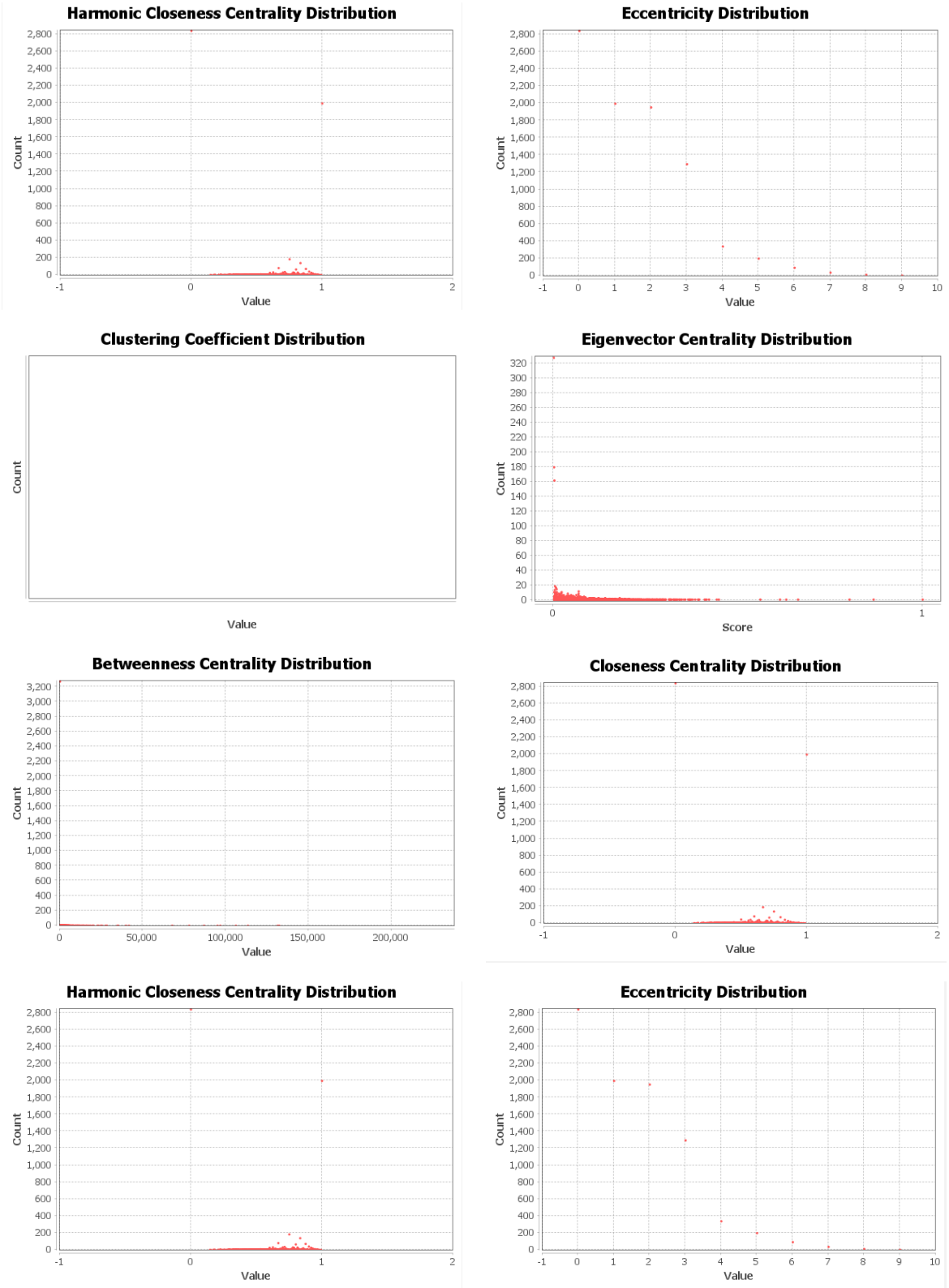
Diameter: 9
Radius: 0
Average Path length: 3.6264032345334267

3.4 Επισκόπηση Τεχνητού Δικτύου



Εικόνα 51. Κατανομή Κεντρικότητας Βαθμού (Με και χωρίς βάρος)

Εικόνα 52. Κατανομές Μεγέθους, μετρικών HITS, Κεντρικότητας Ενδιαμεσότητας, Κεντρικότητας Εγγύτητας



Εικόνα 53. Κατανομή μετρικών ακμών δικτύου

4. Διαφορές Πραγματικού και Τεχνητού Δικτύου

Παρατηρούμε παρόμοιες μετρικές στους βαθμούς των κορυφών και στα δύο δίκτυα. Παρατηρείται μία διαφορά σε In-Degree και Out-Degree, με αλλαγή της συμπεριφοράς του γραφήματος στα δύο δίκτυα. Στο πραγματικό το In-Degree Distribution έχει μικρότερες τιμές σε αντίθεση με το Out-Degree. Παράλληλα στο τεχνητό το Out-Degree έχει μικρότερες τιμές σε σχέση με το In-Degree. Συμπερασματικά καταλήγουμε στο γεγονός ότι στο τεχνητό δίκτυο, οι κόμβοι - papers - καταλήγουν σε άλλα papers σε αντίθεση με το αληθινό, όπου οι κόμβοι καταλήγουν κυρίως στο αρχικό. Επιπρόσθετα, παρατηρούμε πως οι συνδέσεις των κόμβων με άλλους κόμβους είναι όμοιοι σε τιμές και στα δύο δίκτυα.

Τα μεγέθη των δικτύων ακολουθούν παρόμοια μετρική με διαφορά στις μετρικές HITS. Το Authority Distribution στο τεχνητό δίκτυο είναι αυξημένο σε αντίθεση με το πραγματικό δίκτυο ενώ αντιθέτως, το Hubs Distribution είναι μειωμένο σε σχέση με το πραγματικό, όπου είναι εμφανώς αυξημένο. Καταλαβαίνουμε πως στο τεχνητό δίκτυο υπάρχουν περισσότεροι κόμβοι που αποθηκεύουν πολύτιμη πληροφορία για το δίκτυο ενώ η ποιότητα των συνδέσεων μεταξύ των κόμβων είναι χαμηλή σε αντίθεση με το πραγματικό.

Η εγγύτητα των δύο δικτύων παραμένει σε παρόμοιες μετρικές όπως και η κατανομή του μεγέθους, καθώς η κεντρικότητα, που αντιπροσωπεύει το πόσο κοντά είναι οι κόμβοι μεταξύ τους, είναι σημαντικός παράγοντας και στα δύο δίκτυα, με περισσότερες τιμές να καταλαμβάνει το πραγματικό δίκτυο. Παρόμοια μετρική παρατηρείται και στη κατανομή Harmonic Closeness Centrality.

Σημαντική διαφορά παρατηρείται στην κατανομή Eigenvector Centrality, με αυξημένες τιμές στο τεχνητό δίκτυο σε αντίθεση με το πραγματικό. Αυτό συμβαίνει καθώς στο πραγματικό δίκτυο, τα παλαιότερα papers ασκούν επιρρόη στο δίκτυο και στους άλλους κόμβους, με αποτέλεσμα να υπάρχουν χαμηλές τιμές. Αντιθέτως στο τεχνητό δίκτυο, οι περίσσοτεροι κόμβοι ασκούν εξίσου σημαντική επιρρόη στο δίκτυο, με αποτέλεσμα περισσότεροι κόμβοι να θεωρούνται σημαντικοί.

Συμπερασματικά, παρόλες τις διαφορές που παρατηρούνται στις μετρικές της Eigenvector Centrality Distribution και των HITS μετρικών μεταξύ του τεχνητού δικτύου και του αληθινού, εξακολουθεί να υφίσταται η δυνατότητα εκτέλεσης πειραμάτων και αναλύσεων στο τεχνητό δίκτυο. Η μοναδική αυτή απόκλιση δεν θα πρέπει να αποθαρρύνει την πειραματική μελέτη και την προσέγγιση που ακολουθήθηκε με τη χρήση γενετικών αλγορίθμων. Αντιθέτως, αποτελεί μια αφετηρία για περαιτέρω αναζήτηση και ανάπτυξη, προκειμένου να κατανοήσουμε καλύτερα τους παράγοντες που επηρεάζουν τη διαμόρφωση των δικτύων. Οι γενετικοί αλγόριθμοι αποτελούν έναν ισχυρό και πολλά υποσχόμενο τρόπο για την προσομοίωση και την αναπαράσταση περίπλοκων δικτύων, και η δημιουργία τεχνητών δικτύων με παρόμοιες μετρικές με τα πραγματικά δίκτυα μπορεί να αποτελέσει πολύτιμη προσέγγιση για την εξερεύνηση συγκεκριμένων χαρακτηριστικών και συμπεριφοράς που ενδέχεται να επηρεάζουν τη λειτουργία του δικτύου. Το τεχνητό δίκτυο μπορεί να λειτουργήσει ως εργαλείο για να εξετάσουμε ή να επιβεβαιώσουμε υποθέσεις και σενάρια που θα μπορούσαν να ισχύουν και σε πραγματικά δίκτυα. Ακόμη και με τις διαφορές που παρατηρούνται, το τεχνητό δίκτυο μπορεί να παράσχει ένα πειραματικό περιβάλλον για να εξετάσουμε την απόδοση αλγορίθμων, την επίδραση διαφόρων μεταβλητών στη δομή του δικτύου, και να αναδείξουμε πιθανές αιτίες πίσω από τις παρατηρούμενες διαφορές στις μετρικές.

5. Συμπέρασμα και μελλοντικά σχέδια

Συμπερασματικά καταλήγουμε στο γεγονός πως στο αρχικό γράφημα, όσο κεντρικός είναι ένας κόμβος τόσο πιο σημασιολογικά σημαντικός είναι για τους υπόλοιπους στον γράφο. Η απόσταση από όλους τους κόμβους είναι μειωμένη άρα καταλαβαίνουμε ότι η εγγραφή αυτή έχει μέγιστη βαρύτητα. Είναι φανερό πως όσο πιο παλιό είναι ένα paper τόσο περισσότερες εγγραφές θα αναφέρονται σε αυτό, καθώς αποτελεί κύρια βάση και για τα επόμενα χρονολογικά pa-pers. Το δίκτυο που έχει δημιουργηθεί αποτελεί μία εικόνα για το πως είναι δομημένο το αρχικό pdf, χωρίζοντας σε χρωματικές «γειτονιές». Υπάρχουν κόμβοι που βρίσκονται μακριότερα από τους κόμβους στα άκρα του γράφου, γεγονός που σηματοδοτεί είτε την παντελής έλλειψη στις αναφορές των γράφων αυτών ή λάθος της βιβλιοθήκης του Semantic Scholar που μέσω αυτής έγινε η άντληση των πληροφοριών είτε λάθος άντληση του αρχικού DOI για το κάθε paper ξεχωριστά. Γενικότερα η προσπάθεια συγκέντρωσης 8830 κόμβων και 95838 ακμών όπως και η επεξεργασία τους μέσω του αλγόριθμου Force Atlas 2 για εκτενέστερα οπτικά αποτελέσματα θα χρειαζόταν μεγαλύτερη υπολογιστική δύναμη, που λόγω της μεγάλης ποσότητας κόμβων και ακμών, υπάρχει και η πιθανότητα να μην μπορούσε να δημιουργηθεί ποτέ ένα τελικό αποτέλεσμα.

Η διπλωματική είχε ως σκοπό να δείξει τις σχέσεις μεταξύ των papers στο έγγραφο pdf, αντλώντας πληροφορίες με την χρήση των διαφόρων βιβλιοθηκών της python, αλλά και να αναλύσει οπτικά και αριθμητικά τις μεταξύ τους συνδέσεις, δημιουργώντας ένα τεχνητό δίκτυο που να κατοπτρίζει το αληθινό. Οι συνδέσεις αυτές δημιουργήθηκαν επεξεργάζοντας την πληροφορία σχετικά με την βιβλιογραφία της κάθε εγγραφής και συνδέοντας τις εγγραφές μέσω του κώδικα που αναλύθηκε παραπάνω. Με την ανάλυση των 10300 paper του εγγράφου pdf καθώς και η άντληση της πληροφορίας για το κάθε ένα από αυτά, κατέστησε δυνατή την αυτοματοποιημένη συσχέτιση των εγγραφών μεταξύ τους αλλά και την οπτική απεικόνιση τους σε ένα γράφο. Παράλληλα, η εκτέλεση των αλγορίθμων για την δημιουργία του τεχνητού δικτύου, μας έδωσε μία σαφή εικόνα οπτικών αποτελεσμάτων παρόμοια με του αληθινού δικτύου, με μικρές διαφορές σε ορισμένες μετρικές επιρροής κόμβων στο δίκτυο.

Συνολικά, η δημιουργία και η ανάλυση του τεχνητού δικτύου με την υποστήριξη γενετικών αλγορίθμων αποτελεί μια υποσχόμενη προσέγγιση για την πειραματική μελέτη των γράφων. Προσφέρει την ευκαιρία για την εξερεύνηση και την κατανόηση των χαρακτηριστικών τους, καθώς και για την ανάδειξη νέων ερευνητικών ερωτήσεων και προκλήσεων που αφορούν την μοντελοποίηση και την ανάλυση πολύπλοκων δικτύων. Μελλοντικές προσπάθειες μπορούν να επικεντρωθούν στη βελτίωση των αλγορίθμων και των μοντέλων που χρησιμοποιούνται, προσφέροντας μια ακόμη πιο ρεαλιστική αναπαράσταση των πραγματικών δικτύων και ενδυναμώνοντας τις δυνατότητες της ανάλυσης των γραφικών δομών.

Μελλοντικά, η αποτελεσματική άντληση της πληροφορίας αλλά και η διεύρυνση των εργαλείων που χρησιμοποιήθηκαν, καθώς και η μείωση του χρόνου που απαιτείται για την επεξεργασία της κάθε εγγραφής θα λειτουργήσουν θετικά σε μία προσπάθεια επανεκτίμησης των αποτελεσμάτων. Σίγουρα η ανάπτυξη των ορίων επικοινωνίας με τον server θα μπορέσει να επιφέρει πιο ακριβή αποτελέσματα και πληροφορίες σχετικές με τις εγγραφές. Είναι γεγονός πως η δωρεάν εξόρυξη πληροφοριών από τους κατόχους των άρθρων αλλά και η δημιουργία ενός νέου API, με σκοπό την άντληση πληροφορίας για όλα τα DOI των paper που υπάρχουν στο διαδίκτυο, χωρίς περιορισμούς, θα μπορούσε να επιφέρει γρήγορα και πιο ακριβή αποτελέσματα.

6. Βιβλιογραφία

1. Networks, An introduction, M. E. J. Newman, University of Michigan and Santa Fe Institute

2. Networks, Crowds, and Markets: Reasoning about a Highly Connected World, David Ea-sley, Jon Kleinberg, Cambridge University Press, 2010

3. Mastering Gephi Network Visualization, Ken Cherven, 2015 http://gephi.michal-novak.eu/Mastering%20Gephi%20Network%20Visualization.pdf

4. Blondel, V. D., Guillaume, J., & Lefebvre, E. (n.d.). Fast unfolding of communities in large network

5. GEPHI – Introduction to network analysis and visualization, http://www.mar-tingrandjean.ch/wp-content/uploads/2015/10/Gephi-introduction.pdf

6. OLAP on Information Networks: A New Framework for Dealing with Bibliographic Data, https://www.researchgate.net/publication/281328637_OLAP_on_Information_Net-works_A_New_Framework_for_Dealing_with_Bibliographic_Data

7. Infrastructure for a Bibliographic Network, https://www.indexdata.com/infrastructure-for-a-bibliographic-network/

8. Applying centrality measures to impact analysis: A co-authorship network analysis, https://doi.org/10.1002/asi.21128

9. Co-authorship network analysis in health research: method and potential use, https://health-policy-systems.biomedcentral.com/articles/10.1186/s12961-016-0104-5

10. Co-authorship networks and research impact: A social capital perspective, https://www.sciencedirect.com/science/article/pii/S0048733313001169

11. The role of social network analysis as a learning analytics tool in online problem based learning, https://bmcmededuc.biomedcentral.com/articles/10.1186/s12909-019-1599-6

12. Study of co-authorship network of papers in the Journal of Research in Medical Sciences using social network analysis, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3963322/

13. What is a DOI and how do I use them in citations?, https://library.uic.edu/help/arti-cle/1966/what-is-a-doi-and-how-do-i-use-them-in-citations

14. Crossref Metadata Retrieval, https://www.crossref.org/services/metadata-retrieval

15. OpenCitations, https://opencitations.net/

16. Reverse DOI lookups with Crossref, https://partiallyattended.com/2017/01/10/crossref-reverse-lookups/

17. Semantic Scholar, https://pages.semanticscholar.org/about-us

18. Semantic Scholar Python Library, https://github.com/danielnsilva/semanticscholar

19. Elsevier Developer Portal, https://dev.elsevier.com

20. Anaconda, https://www.anaconda.com/products/individual

21. Python, https://www.python.org/

22. Bibliographic Coupling, https://diging.github.io/tethne/doc/0.6.1-beta/tutorial.biblio-coupling.html

23. Karrer, Brian & Newman, M. (2009). Random graph models for directed acyclic networks. Physical review. E, Statistical, nonlinear, and soft matter physics. 80. 046110. 10.1103/PhysRevE.80.046110.

24. Yasui Y, Nakano J (2022) A stochastic generative model for citation networks among academic papers. PLoS ONE 17(6): e0269845. https://doi.org/10.1371/journal.pone.0269845

25. Joyce, J.M. (2011). Kullback-Leibler Divergence. In: Lovric, M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_327