National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering

Comparison of Community Detection methods for Botnet Detection

Diploma Thesis

of

IVI CHATZI

Supervisor: Symeon Papavassiliou
Professor

Athens, July 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering

Comparison of Community Detection methods for Botnet Detection

Diploma Thesis

of

IVI CHATZI

Supervisor: Symeon Papavassiliou
Professor

Approved by the examination committee on 5th July 2023.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Symeon Papavassiliou	Ioanna Roussaki	George Matsopoulos
Professor	Associate Professor	Professor

Athens, July 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Ivi Chatzi
Electrical Computer Engineering Graduate, NTUA

5th July 2023

Περίληψη

Τα botnets είναι ομάδες από παραβιασμένες συσκευές συνδεδεμένες στο Διαδίκτυο που ελέγχονται από κακόβουλο πρόσωπο. Χρησιμοποιούνται για την εκτέλεση κατανεμημένων επιθέσεων άρνησης υπηρεσιών (Distributed Denial-Of-Service, DDoS), κλοπή δεδομένων, ανεπιθύμητη αλληλογραφία ή απάτη κλικ. Η ευρεία εμφάνιση επιθέσεων οδηγούμενων από botnets προκάλεσε την ανάπτυξη μεθόδων ανίχνευσής τους. Μια τέτοια προσέγγιση συνδυάζει στατιστική ανίχνευση ανωμαλιών (anomaly detection) με ανίχνευση κοινοτήτων κοινωνικών δικτύων (community detection) για την ανακάλυψη δυσμενών κόμβων σε ένα δίκτυο. Το πρώτο στάδιο της μεθόδου χρησιμοποιεί καθαρή κίνηση δικτύου για εκμάθηση μιας εμπειρικής κατανομή φυσιολογικής κυκλοφορίας. Αυτή η κατανομή αναφοράς συγκρίνεται στη συνέχεια με νέα κίνηση, με μεγάλες αποκλίσεις να θεωρούνται ανώμαλες. Το δεύτερο στάδιο επεξεργάζεται την ανώμαλη κίνηση βασιζόμενο στην ιδέα ότι οι αλληλεπιδράσεις των κόμβων-bots συσχετίζονται μεταξύ τους και δημιουργείται ένα γράφημα κοινωνικής συσχέτισης (ΓΚΣ) (Social Correlation Graph, SCG). Στο ΓΚΣ τα bots είναι αρκετά πιθανό να σχηματίσουν κοινότητες, οπότε χρησιμοποιείται ανίχνευση κοινοτήτων για την αναγνώρισή τους.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η αξιολόγηση διαφορετικών αλγορίθμων ανίχνευσης κοινοτήτων στο τελικό στάδιο της μεθόδου, συμπεριλαμβανομένου του Hyperbolic Girvan-Newman, ενός αλγόριθμου που χρησιμοποιεί υπερβολική ενσωμάτωση για να επιταχύνει τους υπολογισμούς. Οι αλγόριθμοι συγκρίνονται με βάση την ακρίβειά τους στον εντοπισμό παραβιασμένων κόμβων από τρεις διαφορετικές επιθέσεις από botnets και αναλύονται τα οφέλη και τα μειονεκτήματα κάθε περίπτωσης.

Λέξεις Κλειδιά

Botnets, Ανίχνευση Ανωμαλιών, Ανίχνευση Κοινοτήτων, Ασφάλεια Δικτύων, Ενσωμάτωση Δικτύων, Υπερβολικός Χώρος

Abstract

Botnets are groups of compromised Internet-connected devices that are controlled by a malicious actor. They are used to perform Distributed Denial-of-Service (DDoS) attacks, data theft, spam or click fraud. The widespread popularity of botnet-led attacks caused the development of botnet detection methods. One such approach combines statistical anomaly detection with social network community detection in order to identify compromised nodes in a network. The first stage of the method uses clean network traffic to learn an empirical distribution of normal traffic. This reference distribution is then compared to new traffic, and large deviations are deemed anomalous. The second stage processes the anomalous traffic based on the idea that the interactions of bot nodes are correlated, and creates a Social Correlation Graph (SCG). In the SCG bots are likely to form communities, so community detection is used to identify them.

The aim of this thesis is to evaluate several different community detection algorithms on the final stage of the method, including Hyperbolic Girvan-Newman, an algorithm that utilises hyperbolic embedding in order to speed up calculations. The algorithms are compared based on their accuracy in identifying compromised nodes from three different botnet attacks, and the benefits and drawbacks of each case are analysed.

Keywords

Botnets, Anomaly Detection, Community Detection, Network Security, Network Embedding, Hyperbolic Space

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Συμεών Παπαβασιλείου που μου έδωσε την ευκαιρία να εκπονήσω τη διπλωματική μου εργασία στο εργαστήριό του. Θα ήθελα επίσης να ευχαριστήσω τον κ. Βασίλη Καρυώτη και τον κ. Κωνσταντίνο Τσιτσεκλή για την πολύτιμη βοήθεια και το χρόνο που αφιέρωσαν για την καθοδήγηση της διπλωματικής μου εργασίας, καθώς και τα ερεθίσματα που μου έδωσαν ώστε να γνωρίσω πολλά ενδιαφέροντα θέματα. Τέλος, ευχαριστώ την οικογένεια και τους φίλους μου για την υποστήριξη καθ'όλη τη διάρκεια των σπουδών μου.

Αθήνα, Ιούλιος 2023

ΗΒΗ ΧΑΤΖΗ

Table of Contents

List of Figures

List of Tables

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Τα botnets είναι ομάδες συσκευών συνδεδεμένων στο Διαδίκτυο που έχουν παραβιαστεί και ελέγχονται από κακόβουλο πρόσωπο. Χρησιμοποιούνται για την εκτέλεση κατανεμημένων επιθέσεων άρνησης υπηρεσιών (Distributed Denial-of-Service, DDoS), κλοπή δεδομένων όπως στοιχεία λογαριασμού ή τράπεζας, ανεπιθύμητη αλληλογραφία ή απάτη κλικ. Οι επιθέσεις DDoS υπερφορτώνουν τον στόχο με περιττά αιτήματα, με αποτέλεσμα να αδυνατεί να διεκπεραιώσει φυσιολογικά αιτήματα και επομένως η υπηρεσία του να μην είναι διαθέσιμη. Λόγω της δημοτικότητας και της σοβαρότητας των επιθέσεων DDoS που οδηγούνται από botnets, ο εντοπισμός τους έχει γίνει ένα σημαντικό πρόβλημα κυβερνοασφάλειας. Προκειμένου να αποφευχθούν οι γνωστές τεχνικές ανίχνευσής τους, τα botnets εξελίσσονται διαρκώς, αλλά η συμπεριφορά τους συνεχίζει να επιδεικνύει ορισμένα μοτίβα που επιτρέπουν την ανίχνευσή τους. Σε γενικές γραμμές, είναι πιο δύσκολο να εντοπιστεί μια επίθεση και να αναγνωριστούν οι παραβιασμένοι κόμβοι κατά τα αρχικά στάδια της επίθεσης, αλλά όσο νωρίτερα ανακαλυφθεί το botnet, τόσο πιο εύκολη είναι η άμυνα απέναντί του.

Η εργασία αυτή εξετάζει μια υπάρχουσα μέθοδο ανίχνευσης botnets που συνδυάζει ανίχνευση ανωμαλιών και ανίχνευση κοινοτήτων. Αρχικά, η μέθοδος συγκεντρώνει φυσιολογική κίνηση κατά τη διάρκεια μιας περιόδου εκπαίδευσης, η οποία στη συνέχεια χρησιμοποιείται για να προσδιοριστεί εάν η νέα κίνηση είναι ανώμαλη βάσει της απόκλισής της από την κίνηση αναφοράς. Στη συνέχεια, η ανώμαλη κίνηση υποβάλλεται σε επεξεργασία προκειμένου να ομαδοποιηθούν τα bots με βάση τη συσχέτιση των αλληλεπιδράσεών τους. Η συμβολή της παρούσας διπλωματικής εργασίας είναι η εξέταση και σύγκριση πολλαπλών διαφορετικών αλγορίθμων ανίχνευσης κοινοτήτων στο τελικό στάδιο της μεθόδου.

Τρεις διαφορετικές επιθέσεις από botnets χρησιμοποιούνται για την αξιολόγηση των αλγορίθμων, οι οποίοι στη συνέχεια συγκρίνονται ως προς την ακρίβειά τους στον εντοπισμό των παραβιασμένων κόμβων. Στα πειράματα χρησιμοποιείται κυκλοφορία από τα πρώτα στάδια κάθε επίθεσης, καθώς τα επόμενα στάδια είναι εύκολα ανιχνεύσιμα λόγω του μεγάλου όγκου κυκλοφορίας. Πέντε αλγόριθμοι ανίχνευσης κοινότητας συγκρίνονται, συμπεριλαμβανομένου του Hyperbolic Girvan-Newman, ενός αλγόριθμου που ενσωματώνει το δίκτυο σε υπερβολικό χώρο για ταχύτητα υπολογισμών. Στο τέλος αναλύονται τα οφέλη και τα μειονεκτήματα κάθε αλγορίθμου.

0.2 Θεωρητικό Υπόβαθρο

0.2.1 Θεωρία Γράφων και Δικτύων

Ένας γράφος ή δίκτυο $G = \{V, E\}$, είναι μια μαθηματική οντότητα που περιέχει ένα σύνολο από κορυφές (ή κόμβους) V και ένα σύνολο ακμών E. Μια ακμή $(u, v) \in E$ αναπαριστά μια διασύνδεση μεταξύ δύο κόμβων $u, v \in E$. Οι γράφοι χρησιμοποιούνται σε ποικιλία εφαρμογών όπως οδικά δίκτυα, το Διαδίκτυο, συνδέσεις κοινωνικών μέσων δικτύωσης, κλπ. Ένα παράδειγμα γράφου φαίνεται στο σχήμα 1.

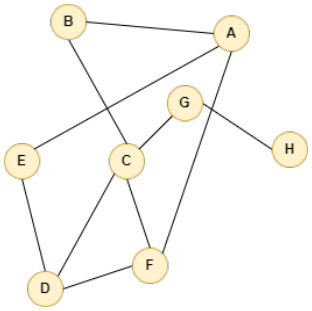

Figure 1. *Παράδειγμα γράφου με 8 κόμβους και 10 ακμές*

Δύο κόμβοι λέγονται γειτονικοί αν είναι συνδεδεμένοι με ακμή. Η γειτονιά N_u ενός κόμβου u είναι το σύνολο των γειτονικών κόμβων του: $N_u = \{v \in V : (u, v) \in E\}$, ενώ το μέγεθος της γειτονιάς λέγεται βαθμός του κόμβου. Μια συνηθισμένη αναπαράσταση γράφων είναι οι πίνακες γειτνίασης. Ο πίνακας γειτνίασης A ενός γράφου με n κόμβους είναι ένας τετραγωνικός πίνακας $n \times n$, όπου κάθε γραμμή και στήλη αντιστοιχεί σε έναν κόμβο. Αν υπάρχει η ακμή (i, j) τότε $A_{ij} = 1$, διαφορετικά $A_{ij} = 0$.

0.2.2 Ανίχνευση Κοινοτήτων

Σύνθετα δίκτυα είναι γράφοι με πολύπλοκη δομή των οποίων οι ιδιότητες εξαρτώνται σε μεγάλο βαθμό από τον τρόπο που οι κόμβοι συνδέονται και αλληλεπιδρούν μεταξύ τους [7]. Πολλά δίκτυα στην πραγματική ζωή είναι σύνθετα, όπως τα κοινωνικά δίκτυα ή τα δίκτυα υπολογιστών [8], [9].

Τα σύνθετα δίκτυα έχουν την τάση να σχηματίζουν κοινότητες, δηλαδή ομάδες κόμβων που είναι πιο πυκνά συνδεδεμένοι μεταξύ τους συγκριτικά με το υπόλοιπο δίκτυο [10]. Η γνώση των κοινοτήτων αυτών επιτρέπει καλύτερη κατανόηση της δομής του δικτύου και της διάδοσης πληροφορίας μέσα σε αυτό, οπότε η ανίχνευσή τους είναι ένα πρόβλημα με ποικίλες εφαρμογές. Για παράδειγμα, πολλά συστήματα συστάσεων βασίζονται στο γεγονός ότι χρήστες στην ίδια κοινότητα είναι πιο πιθανό να έχουν κοινά ενδιαφέροντα [11]. Η γνώση κοινοτήτων σε δίκτυα μπορεί επίσης να χρησιμοποιηθεί στη μελέτη διάδοσης επιδημιών όπως του COVID-19 [12], [13]. Στην ενότητα αυτή θα παρουσιάσουμε τις τεχνικές ανίχνευσης κοινοτήτων που χρησιμοποιούνται στη διπλωματική εργασία.

Girvan-Newman

Ο αλγόριθμος Girvan-Newman [14] λειτουργεί αφαιρώντας ακμές από το γράφο μέχρι να αποσυνδεθεί σε συγκεκριμένο αριθμό κοινοτήτων που προσδιορίζεται εξαρχής. Η ακμή που αφαιρείται σε κάθε βήμα του αλγορίθμου είναι αυτή με τη μεγαλύτερη τιμή κεντρικότητας ενδιαμεσικότητας ακμής. Η μετρική αυτή εκφράζει το ποσοστό των ελάχιστων μονοπατιών μεταξύ όλων των ζευγών κορυφών του γράφου που περνάνε από μια συγκεκριμένη ακμή. Όσο μεγαλύτερη είναι η τιμή κεντρικότητας ενδιαμεσικότητας ακμής (edge-betweenness centrality), τόσο πιθανότερο η ακμή να συνδέει δύο διαφορετικές κοινότητες, οπότε ο αλγόριθμος αφαιρεί ακμές με ψηλή τιμή της μετρικής αυτής έως ότου ο γράφος φτάσει στον προκαθορισμένο αριθμό συνεκτικών συνιστωσών. Μειονέκτημα του αλγορίθμου είναι ότι οι συνεχείς υπολογισμοί κεντρικότητας ενδιαμεσικότητας ακμής είναι ιδιαίτερα κοστοβόροι οπότε δεν είναι κατάλληλος για δίκτυα μεγάλου μεγέθους.

Louvain

Ο αλγόριθμος Louvain [15] είναι μια άπληστη μέθοδος που αποσκοπεί στη μεγιστοποίηση της αρθρωτότητας των κοινοτήτων. Η αρθρωτότητα (modularity) είναι μια μετρική που εκφράζει πόσο καλή είναι η διαμέριση του δικτύου σε κοινότητες, σε σύγκριση με την περίπτωση που οι ακμές ήταν τυχαίες. Αρχικά κάθε κόμβος τοποθετείται στη δική του κοινότητα, έπειτα μετακινείται στην κοινότητα ενός γείτονά του με σκοπό τη μεγιστοποίηση της αρθρωτότητας, αν γίνεται. Στη συνέχεια κάθε κοινότητα ομαδοποιείται σε μια μοναδική κορυφή και ο αλγόριθμος επαναλαμβάνεται έως ότου σταματήσει να βελτιώνεται η αρθρωτότητα του δικτύου. Τα πλεονεκτήματα της μεθόδου αυτής είναι ότι δεν απαιτεί τη γνώση του αριθμού κοινοτήτων εξαρχής και είναι πιο γρήγορος από τον Girvan-Newman.

Walktrap

Ο Walktrap [16] είναι ένας αλγόριθμος ανίχνευσης κοινοτήτων που χρησιμοποιεί τυχαίους περιπάτους. Βασίζεται στην ιδέα ότι μικροί τυχαίοι περίπατοι σε ένα δίκτυο τείνουν να μένουν εντός της ίδιας κοινότητας.

Spectral Clustering

Ο Spectral clustering [17] είναι ένας αλγόριθμος που χρησιμοποιεί τα ιδιοδιανύσματα του Λαπλασιανού πίνακα του δικτύου. Ο πίνακας αυτός ορίζεται ως $L = D - A$, όπου A ο πίνακας γειτνίασης και D ένας διαγώνιος πίνακας που κάθε στοιχείο του αντιστοιχεί στο βαθμό ενός κόμβου. Ο αλγόριθμος υπολογίζει τα ιδιοδιανύσματα και τις ιδιοτιμές του Λαπλασιανού πίνακα και εφαρμόζει ομαδοποίηση K-means στα ιδιοδιανύσματα των k μεγαλύτερων ιδιοτιμών, όπου k ο προκαθορισμένος αριθμός κοινοτήτων.

Hyperbolic Girvan-Newman

Ο Hyperbolic Girvan-Newman [5] είναι μια παραλλαγή-προσέγγιση του αλγορίθμου Girvan-Newman. Βασίζεται στην ενσωμάτωση του δικτύου σε υπερβολική γεωμετρία μέσω ενσωμάτωσης Rigel [18], ώστε αντί να γίνεται ο κοστοβόρος υπολογισμός των κεντρικοτήτων

ενδιαμεσικότητας ακμής, να υπολογίζεται πολύ γρηγορότερα το υπερβολικό ανάλογο [19] με ελάχιστο κόστος ακρίβειας. Σε κάθε επανάληψη του αλγορίθμου ενσωματώνεται η μεγαλύτερη συνεκτική συνιστώσα του γράφου και αφαιρούνται οι k ακμές με τη μεγαλύτερη τιμή υπερβολικής κεντρικότητας ενδιαμεσιμότητας ακμής, όπου k το μέγεθος δέσμης.

0.2.3 Botnets

Τα botnets είναι ομάδες συσκευών συνδεδεμένες στο Διαδίκτυο, όπως υπολογιστές ή έξυπνες συσκευές, των οποίων η ασφάλεια έχει παραβιαστεί και ο έλεγχος γίνεται από κακόβουλη πηγή. Κάθε συσκευή μετατρέπεται σε bot αφού μολυνθεί από κακόβουλο λογισμικό και μετά ελέγχεται εξ αποστάσεως. Τα botnets χρησιμοποιούνται για ποικιλία επιθέσεων όπως ανεπιθύμητη αλληλογραφία, κατανεμημένες επιθέσεις άρνησης υπηρεσιών, κλοπή δεδομένων (κωδικοί, αριθμοί πιστωτικών καρτών, εταιρικά δεδομένα) ή απάτες κλικ για τη δημιουργία ψευδούς κίνησης [20]. Οι κατανεμημένες επιθέσεις άρνησης υπηρεσιών πραγματοποιούνται όταν πολλά bots κατακλύζουν το θύμα με ψευδείς αιτήσεις ώστε να αδυνατεί να εξυπηρετήσει φυσιολογικές αιτήσεις. Μια γνωστή τέτοια επίθεση από botnet ήταν οι επιθέσεις Mirai [21] το 2016.

Παραδοσιακά τα botnets λειτουργούσαν με μοντέλο πελάτη-εξυπηρετητή, για παράδειγμα μέσω Internet Relay Chat (IRC) [22]. Το IRC είναι ένα σύστημα συνομιλιών όπου τα bots συνδέονται σε έναν διακομιστή και εγγράφονται σε ένα κανάλι όπου στέλνονται οδηγίες. Λογισμικά ασφαλείας αντιμετωπίζουν τις περιπτώσεις αυτές κλείνοντας τα κανάλια οδηγιών, με αποτέλεσμα τα bots να ληθαργούν, οπότε τα botnets αναγκάζονται να αλλάζουν συνεχώς κανάλια λειτουργίας. Πιο πρόσφατα botnets, όπως το Gameover ZeuS [23], χρησιμοποιούν μοντέλα αλληλεπίδρασης ομότιμων και κρυπτογραφία δημόσιου κλειδιού.

Εξαιτίας της υψηλής συχνότητας κατανεμημένων επιθέσεων άρνησης υπηρεσιών [24] έχουν αναπτυχθεί πολλές μέθοδοι για την ανίχνευση botnets. Κάποιες μέθοδοι επιτηρούν ένα συγκεκριμένο μηχάνημα για ύποπτα αρχεία ή υψηλό επεξεργαστικό κόστος, ενώ άλλοι μέθοδοι επιτηρούν ένα ολόκληρο δίκτυο για ύποπτη κίνηση, ασυνήθιστη καθυστέρηση, αυξημένη κίνηση σε σπανίως χρησιμοποιούμενες θύρες, κλπ [25].

0.3 Μεθοδολογία

Η μέθοδος που χρησιμοποιούμε για την ανίχνευση botnets περιγράφεται στο [26], με τη διαφορά ότι χρησιμοποιούμε διαφορετικές μεθόδους ανίχνευσης κοινοτήτων στο τελικό στάδιο. Η μέθοδος χωρίζεται σε δύο μέρη που περιγράφονται στις επόμενες δύο ενότητες.

0.3.1 Ανίχνευση ανώμαλης κίνησης

Το πρώτο στάδιο της μεθόδου χρησιμοποιεί τεχνικές ανίχνευσης ανωμαλιών για τον εντοπισμό χρονικών περιόδων με ασυνήθιστη δικτυακή κίνηση. Αρχικά πραγματοποιείται εκμάθηση της φυσιολογικής κυκλοφορίας του δικτύου σε μια καθορισμένη χρονική περίοδο χωρίς επίθεση με σκοπό τη δημιουργία μιας κατανομής αναφοράς. Στη συνέχεια η περίοδος αξιολόγησης χωρίζεται σε μικρά χρονικά παράθυρα, των οποίων η κίνηση συγκρίνεται με

την κίνηση αναφοράς. Μεγάλες αποκλίσεις από την φυσιολογική κίνηση θεωρούνται ανώμαλες. Ανάλογα αν η κυκλοφορία δίνεται σε αρχείο pcap ή netflow η αντιμετώπιση είναι διαφορετική.

Αρχεία netflow

Στα αρχεία netflow κάθε ροή εκπροσωπεί την επικοινωνία μεταξύ δύο μηχανών και περιέχει πληροφορία όπως τη διάρκεια της επικοινωνίας, τον συνολικό αριθμό πακέτων που στάλθηκαν, τι πρωτόκολλο χρησιμοποιήθηκε, κλπ. Η ανίχνευση ανωμαλιών σε τέτοιου είδους αρχεία πραγματοποιείται επιλέγοντας κάποια χαρακτηριστικά των ροών, τα οποία κβαντοποιούνται, και το διάνυσμα που προκύπτει αντιμετωπίζεται ως τιμή μιας διακριτής τυχαίας μεταβλητής. Στην περίοδο εκμάθησης χτίζεται μια εμπειρική κατανομή πιθανότητας αναφοράς από όλες τις ροές της περιόδου. Έπειτα, για κάθε χρονικό παράθυρο αξιολόγησης χτίζεται κι εκεί μια κατανομή πιθανότητας, η οποία συγκρίνεται με την κατανομή αναφοράς μέσω της απόκλισης Kullback-Leibler (KL):

$$D(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{1}$$

όπου p η κατανομή αξιολόγησης, q η κατανομή αναφοράς και $\mathcal{X}$ το κοινό πεδίο τιμών τους. Η απόκλιση KL μεταξύ των δύο κατανομών συγκρίνεται με ένα κατώφλι, και αν είναι μεγαλύτερη από αυτό τότε το χρονικό παράθυρο χαρακτηρίζεται ως ανώμαλο.

Αρχεία pcap

Τα αρχεία pcap περιέχουν πληροφορίες για πακέτα που μεταδίδονται σε ένα δίκτυο. Στα αρχεία τέτοιου είδους μετριέται η κατανομή βαθμού κόμβων σε γράφους αλληλεπίδρασης και επιλέγεται ένα μοντέλο γράφου στην περίοδο εκμάθησης. Οι κατανομές κόμβων της περιόδου αλληλεπίδρασης συγκρίνονται με την κατανομή αναφοράς και αν αποκλίνουν περισσότερο από μια τιμή κατωφλίου τότε το παράθυρο θεωρείται ανώμαλο. Η συνάρτηση σύγκρισης είναι διαφορετική ανάλογα με το μοντέλο γράφου που επιλέχθηκε.

Ο γράφος αλληλεπίδρασης σε κάποιο χρονικό παράθυρο δημιουργείται ενώνοντας με ακμή δύο διευθύνσεις εάν υπήρξε επικοινωνία μεταξύ τους στο παράθυρο αυτό. Χωρίζουμε την περίοδο εκμάθησης σε πολλά παράθυρα και επιλέγουμε τυχαία μερικά από αυτά για τη δημιουργία γράφων αλληλεπίδρασης. Στους γράφους αυτούς μετράμε το βαθμό τυχαία επιλεγμένων κόμβων για να χτίσουμε την κατανομή αναφοράς. Στη συνέχεια ελέγχεται αν η κατανομή αυτή είναι πιθανότερο να προέρχεται από τυχαίο γράφο [27], [28] ή γράφο χωρίς κλίμακα [29], [30], χρησιμοποιώντας τις εξισώσεις 3.2 και 3.4. Στα πειράματά μας πάντα επιλέγονταν το μοντέλο τυχαίου γράφου με παράμετρο $\hat{\lambda}$ που υπολογίζεται από την κατανομή ως εξής:

$$\hat{\lambda} = \frac{1}{M} \sum_{i=1}^{M} d_i \tag{2}$$

όπου M ο αριθμός δειγμάτων βαθμών κόμβου της κατανομής αναφοράς και d_i το i-οστό δείγμα.

Η περίοδος αξιολόγησης επίσης χωρίζεται σε μικρότερα χρονικά παράθυρα, σε καθένα από τα οποία δημιουργείται ο γράφος αλληλεπίδρασης και υπολογίζεται η κατανομή βαθμού κόμβων. Στο μοντέλο τυχαίου γράφου, η κατανομή αξιολόγησης συγκρίνεται με την κατανομή αναφοράς μέσω της ακόλουθης συνάρτησης:

$$I_{Rand}(q;\hat{\eta}) = D(q \,||\, p_{\hat{\eta}}) + \frac{1}{2}(\bar{q} - \hat{\eta}) + \frac{\bar{q}}{2} log(\hat{\eta}) - \frac{\bar{q}}{2} log(\bar{q}) \tag{3}$$

όπου $p_{\hat{\eta}}$ η κατανομή αναφοράς, q η κατανομή αξιολόγησης και $\bar{q}$ η μέση τιμή της. Αν η τιμή της συνάρτησης αυτής είναι μεγαλύτερη από κάποιο προκαθορισμένο κατώφλι τότε το χρονικό παράθυρο θεωρείται ανώμαλο.

0.3.2 Ανίχνευση botnets

Το σύνολο ανώμαλων χρονικών παραθύρων χρησιμοποιείται στη συνέχεια για τον εντοπισμό των κόμβων-bots. Αρχικά εντοπίζονται κάποιοι κεντρικοί κόμβοι που είτε είναι bots-αρχηγοί είτε είναι κύριοι στόχοι της επίθεσης. Οι κόμβοι αυτοί έχουν αυξημένες αλληλεπιδράσεις με το υπόλοιπο δίκτυο, καθώς οι bots-αρχηγοί πρέπει να στέλνουν οδηγείες στα υπόλοιπα bots, και οι κύριοι στόχοι δέχονται διαρκώς επίθεση από τα bots. Ένας κόμβος i χαρακτηρίζεται ως κεντρικός αν έχει συνολικό μέτρο αλληλεπίδρασης e_i μεγαλύτερο από κάποιο κατώφλι τ:

$$e_i = \frac{1}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} \sum_{j=1}^{n} G_k^{ij} \tag{4}$$

όπου $\mathcal{A}$ το σύνολο των ανώμαλων παραθύρων και G_k^{ij} ο αριθμός των αλληλεπιδράσεων μεταξύ των κόμβων i,j στο παράθυρο k.

Αφού εντοπίσουμε τους κεντρικούς κόμβους, εξετάζουμε τις αλληλεπιδράσεις των υπόλοιπων κόμβων με αυτούς. Για κάθε κόμβο i έστω $x_i^k = \sum_{j \in \mathcal{N}} G_k^{ij}$ ο αριθμός των αλληλεπιδράσεών του με κεντρικούς κόμβους στο παράθυρο k, όπου $\mathcal{N}$ το σύνολο των κεντρικών κόμβων. Έστω επίσης X_i ο συνολικός αριθμός αλληλεπιδράσεων του κόμβου i με κεντρικούς κόμβους σε όλο το $\mathcal{A}$. Υπολογίζουμε το συντελεστή συσχέτισης μεταξύ ζευγών κόμβων i,j:

$$\rho(X_i, X_j) = \frac{\sum_{k=1}^{|\mathcal{A}|} \left((x_i^k - \bar{X}_i)(x_j^k - \bar{X}_j) \right)}{(|\mathcal{A}| - 1)\sigma(X_i)\sigma(X_j)} \tag{5}$$

όπου $\bar{X}_i$ και $\sigma(X_i)$ ο μέσος όρος και η διασπορά των x_i και $\rho(X_i, X_j) = 0$ αν κάποια διασπορά ισούται με 0.

Φτιάχνουμε τον γράφο κοινωνικής συσχέτισης (ΓΚΣ) με κόμβους όλους τους κόμβους στο $\mathcal{A}$ και κάθε ακμή i,j υπάρχει εάν $|\rho(X_i, X_j)| > \tau_\rho$ όπου τ_ρ κάποιο κατώφλι. Στο γράφο αυτό τα bots είναι αρκετά πιθανό να είναι συνδεδεμένα μεταξύ τους επειδή οι αλληλεπιδράσεις τους με τους κεντρικούς κόμβους συσχετίζονται. Οπότε για να τους εντοπίσουμε αρκεί να χρησιμοποιήσουμε κάποιο αλγόριθμο ανίχνευσης κοινοτήτων στο ΓΚΣ. Τέλος, για να αποφασίσουμε ποια κοινότητα είναι τα bots, επιλέγουμε αυτή με τη μεγαλύτερη μέση τιμή

μέτρου αλληλεπίδρασης με κεντρικούς κόμβους που ορίζεται ως εξής για τον κόμβο i:

$$r_i = \frac{1}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} \sum_{j \in \mathcal{N}} e_j G_k^{ij} \tag{6}$$

0.4 Πειράματα

Στην ενότητα αυτή θα παρουσιάσουμε τα πειράματα που διεξήχθηκαν στα πλαίσια της διπλωματικής εργασίας. Για κάθε πείραμα αρχικά αναφέρουμε πληροφορίες για τα δεδομένα και ό,τι προεπεξεργασία πραγματοποιήθηκε σε αυτά. Μετά εξηγούμε τη διαδικασία εκμάθησης και δείχνουμε τα αποτελέσματα ανίχνευσης ανωμαλιών. Τέλος αναφέρουμε τις παραμέτρους που χρησιμοποιήθηκαν για τη δημιουργία του ΓΚΣ και παρουσιάζουμε τα αποτελέσματα της ανίχνευσης κοινοτήτων.

Το πρώτο και τρίτο πείραμα ξεκίνησαν ως προσπάθεια αναπαραγωγής των αποτελεσμάτων του [26], αλλά στην πορεία απέκλιναν και χρησιμοποιήσαμε διαφορετικό ΓΚΣ. Το δεύτερο πείραμα ήταν πρωτότυπο.

Οι αλγόριθμοι ανίχνευσης κοινοτήτων που δοκιμάστηκαν ήταν οι Girvan-Newman, Louvain, Spectral Clustering, Walktrap, Hyperbolic Girvan-Newman. Αξιολογήθηκαν ως προς την κανονικοποιημένη αμοιβαία πληροφορία (normalised mutual information, NMI) και το F1-score σχετικά με τις πραγματικές κοινότητες, που ήταν διαθέσιμες σε κάθε πείραμα. Η αμοιβαία πληροφορία μεταξύ δύο διακριτών τυχαίων μεταβλητών X, Y ορίζεται ως εξής:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_{XY}(x,y) log\left(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\right) \tag{7}$$

ενώ η κανονικοποιημένη αμοιβαία πληροφορία:

$$NMI(X,Y) = \frac{2I(X;Y)}{H(X) + H(Y)} \tag{8}$$

όπου $H(X), H(Y)$ η εντροπία των X, Y. Το F1-score ορίζεται ως εξής:

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{9}$$

όπου TP = *true positives* ο αριθμός των bots που ανιχνεύθηκαν σωστά, FP = *false positives* ο αριθμός των άκακων κόμβων που ανιχνεύθηκαν ως bots και FN = *false negatives* ο αριθμός των bots που δεν ανιχνεύθηκαν.

Η υλοποίηση κώδικα έγινε σε Python3, εκτός από το κομμάτι της ενσωμάτωσης Rigel που υλοποιήθηκε από τους [31] σε C++. Ο κώδικας του HGN υλοποιήθηκε από τους [5]. Χρησιμοποιήσαμε επίσης το Wireshark, εργαλείο ανάλυσης πρωτοκόλλων δικτύου, για προεπεξεργασία αρχείων pcap.

0.4.1 Πειράματα στο σύνολο δεδομένων CTU-13

Περιγραφή και προεπεξεργασία συνόλου δεδομένων

Το CTU-13 είναι ένα σύνολο δεδομένων που καταγράφηκε στο Czech Technical University [32]. Περιέχει 13 καταγραφές δικτύων με επιθέσεις από botnets σε μορφή netflow όπου κάθε ροή είναι χαρακτηρισμένη ως botnet ή όχι από τους συγγραφείς του συνόλου δεδομένων. Χρησιμοποιήσαμε τη 2η καταγραφή που περιέχει 4.5 ώρες κυκλοφορίας με επίθεση από IRC botnet που έστελνε διαρκώς ανεπιθύμητη κίνηση.

Το αρχείο είχε μέγεθος 241MB και περιείχε 1808122 ροές, από τις οποίες το 1.04% ήταν από bots. Φιλτράραμε τις ροές ώστε να απομείνουν μονάχα όσες χρησιμοποιούσαν πρωτόκολλα IPv4. Από κάθε ροή κρατήσαμε τα εξής χαρακτηριστικά:

- **StartTime:** ο χρόνος εκκίνησης της ροής
- **Duration:** η διάρκεια της ροής
- **EndTime:** ο χρόνος τερματισμού της ροής, υπολογίστηκε προσθέτοντας το χρόνο εκκίνησης με τη διάρκεια
- **SrcAddr:** η διεύθυνση πηγής
- **DstAddr:** η διεύθυνση προορισμού
- **Sport:** η θύρα πηγής
- **Dport:** η θύρα προορισμού
- **SrcBytes:** ο αριθμός των bytes που στάλθηκαν από την πηγή στον προορισμό
- **DstBytes:** ο αριθμός των bytes που στάλθηκαν από την τον προορισμό στην πηγή, υπολογίστηκε αφαιρώντας το SrcBytes από τον συνολικό αριθμό bytes της ροής
- **Label:** ο χαρακτηρισμός της ροής

Στην εικόνα 2 φαίνονται οι πέντε πρώτες γραμμές του συνόλου δεδομένων μετά την προεπεξεργασία.

	StartTime	EndTime	Dur	SrcAddr	DstAddr	Sport	Dport	SrcBytes	DstBytes	Label
0	2011/08/11 09:49:35.721274	2011/08/11 10:24:05.694419	2069.973145	203.253.8.233	147.32.84.229	30533	13363	123	74	flow=Background-UDP-Established
1	2011/08/11 09:49:35.721530	2011/08/11 10:04:31.710788	895.989258	81.47.154.13	147.32.84.229	49200	13363	4501	2531	flow=Background
2	2011/08/11 09:49:35.721918	2011/08/11 09:49:35.722038	0.000120	147.32.84.229	78.42.25.171	13363	42988	2858	0	flow=Background-UDP-Attempt
3	2011/08/11 09:49:35.722518	2011/08/11 10:48:57.649764	3561.927246	147.32.84.229	113.128.219.130	13363	59790	13419	2351	flow=Background-UDP-Established
4	2011/08/11 09:49:35.723816	2011/08/11 09:49:35.723816	0.000000	147.32.84.229	60.50.167.24	13363	40253	60	0	flow=Background-UDP-Attempt

Figure 2. *Αποτέλεσμα προεπεξεργασίας του συνόλου δεδομένων CTU-13*

Περίοδος εκμάθησης

Η περίοδος εκμάθησης ήταν τα πρώτα 25 λεπτά ροών, που δεν είχε ξεκινήσει ακόμα η επίθεση. Ο συνολικός αριθμός ροών εκμάθησης ήταν 257728 (14% του συνόλου δεδομένων). Σε συμφωνία με το αντίστοιχο πείραμα στο [26] κβαντοποιήθηκαν τα εξής χαρακτηριστικά κάθε ροής: διεύθυνση πηγής, διεύθυνση προορισμού, θύρα πηγής, θύρα προορισμού, διάρκεια, bytes πηγής, bytes προορισμού.

Οι διευθύνσης πηγής προορισμού κβαντοποιήθηκαν μέσω ομαδοποίησης K-Means 256 ομάδων, αφού οι διευθύνσεις IP $a = (a_1, a_2, a_3, a_4) \in \{0, 1, ..., 255\}^4$ μετατραπούν σε δεκαδικό σύστημα μέσω του τύπου $x_{10} = \sum_{i=1}^{4} 256^{4-i} a_i$. Η διάρκεια και ο αριθμός bytes πηγής και προορισμού κβαντοποιήθηκαν με εκθετική κλίμακα, ενώ οι θύρες πηγής και προορισμού κβαντοποιήθηκαν γραμμικά ανά 10000 με εξαίρεση τις πρώτες 1024 καταχωρημένες θύρες. Η εικόνα 3 δείχνει τις πρώτες πέντε γραμμές του κβαντοποιημένου συνόλου εκμάθησης. Αφού κβαντοποιηθούν τα δεδομένα, κάθε ροή αντιπροσωπεύεται πλέον από ένα διάνυσμα 7 ακεραίων, οι τιμές του οποίου δημιουργούν την εμπειρική κατανομή πιθανότητας αναφοράς.

	Dur	SrcAddr	DstAddr	Sport	Dport	SrcBytes	DstBytes
0	4	245	140	4	2	2	1
1	3	151	140	5	2	3	3
2	0	140	135	2	5	3	0
3	4	140	128	2	6	4	3
4	0	140	213	2	5	1	0

Figure 3. *Κβαντοποιημένο σύνολο εκμάθησης*

Ανίχνευση ανωμαλιών

Επιλέγεται μια 5λεπτη περίοδος αξιολόγησης 26 λεπτά μετά του τέλους της εκμάθησης (51-56 λεπτά στο σύνολο δεδομένων). Το σύνολο αξιολόγησης κβαντοποιείται όπως την περίοδο εκμάθησης, εκτός από τις διευθύνσεις όπου χρησιμοποιείται το εκπαιδευμένο μοντέλο K-means. Έπειτα το 5λεπτο σύνολο αξιολόγησης διασπάται σε παράθυρα 2 δευτερολέπτων και η κατανομή κάθε παραθύρου συγκρίνεται με την κατανομή αναφοράς. Τα αποτελέσματα της ανίχνευσης ανωμαλιών φαίνονται στην εικόνα 4. Η μπλε γραμμή αντιστοιχεί στον αριθμό των ροών bot επί 0.005, ενώ η πορτοκαλί γραμμή αντιστοιχεί στην απόκλιση των κατανομών. Παρατηρούμε ότι η απόκλιση είναι μεγαλύτερη στα παράθυρα με πολλές ροές bot οπότε επιλέγουμε κατώφλι 0.1 για να τις ανιχνεύσουμε.

Δημιουργία ΓΚΣ

Για την ανίχνευση bots χρησιμοποιήσαμε ένα μικρό σύνολο ανώμαλων παραθύρων, συγκεκριμένα τα 47-55 της εικόνας 4. Περιείχε 936 διευθύνσεις, μεταξύ των οποίων 119 ήταν bots. Το συνολικό μέτρο αλληλεπίδρασης των κόμβων φαίνεται στην εικόνα 5, όπου κάθε κόκκινος σταυρός αντιστοιχεί σε έναν κόμβο. Χρησιμοποιήσαμε κατώφλι $\tau = 20$ για την επιλογή 4 κεντρικών κόμβων, 2 bots και 2 θύματα.

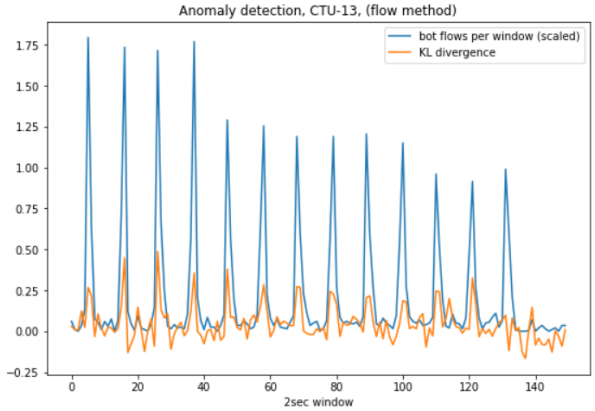

Figure 4. *Ανίχνευση ανωμαλιών στο σύνολο δεδομένων CTU-13*

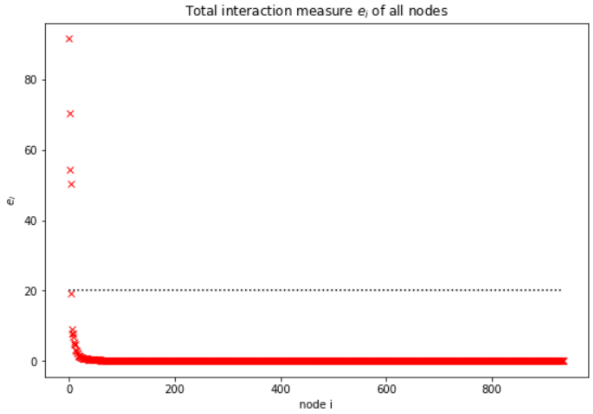

Figure 5. *Συνολικό μέτρο αλληλεπίδρασης*

Για τη δημιουργία του ΓΚΣ χρησιμοποιήσαμε $\tau_\rho = 0.88$ και προέκυψε γράφος 103 κόμβων με 8 bots όπως φαίνεται πάνω αριστερά στην εικόνα 6.

Σύγκριση αλγορίθμων ανίχνευσης κοινοτήτων

Εφαρμόζουμε τους 5 αλγορίθμους ανίχνευσης κοινοτήτων στο ΓΚΣ. Για το HGN χρησιμοποιήθηκαν 3 κόμβοι ορόσημα και μέγεθος δέσμης 1. Τα αποτελέσματα φαίνονται στον πίνακα 1 και στην εικόνα 6. Παρατηρούμε ότι όλοι οι αλγόριθμοι εκτός του Spectral Clustering εντοπίζουν την κοινότητα 4 bots στην κορυφή, ενώ ο Spectral Clustering εντοπίζει μονάχα τους 2 κεντρικούς κόμβους.

Αλγόριθμος	**NMI**	**F1-score**
Girvan-Newman	0.5	0.67
Louvain	0.5	0.67
Spectral Clustering	0.28	0.44
Walktrap	0.5	0.67
HGN	0.5	0.67

Table 1. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στο CTU-13*

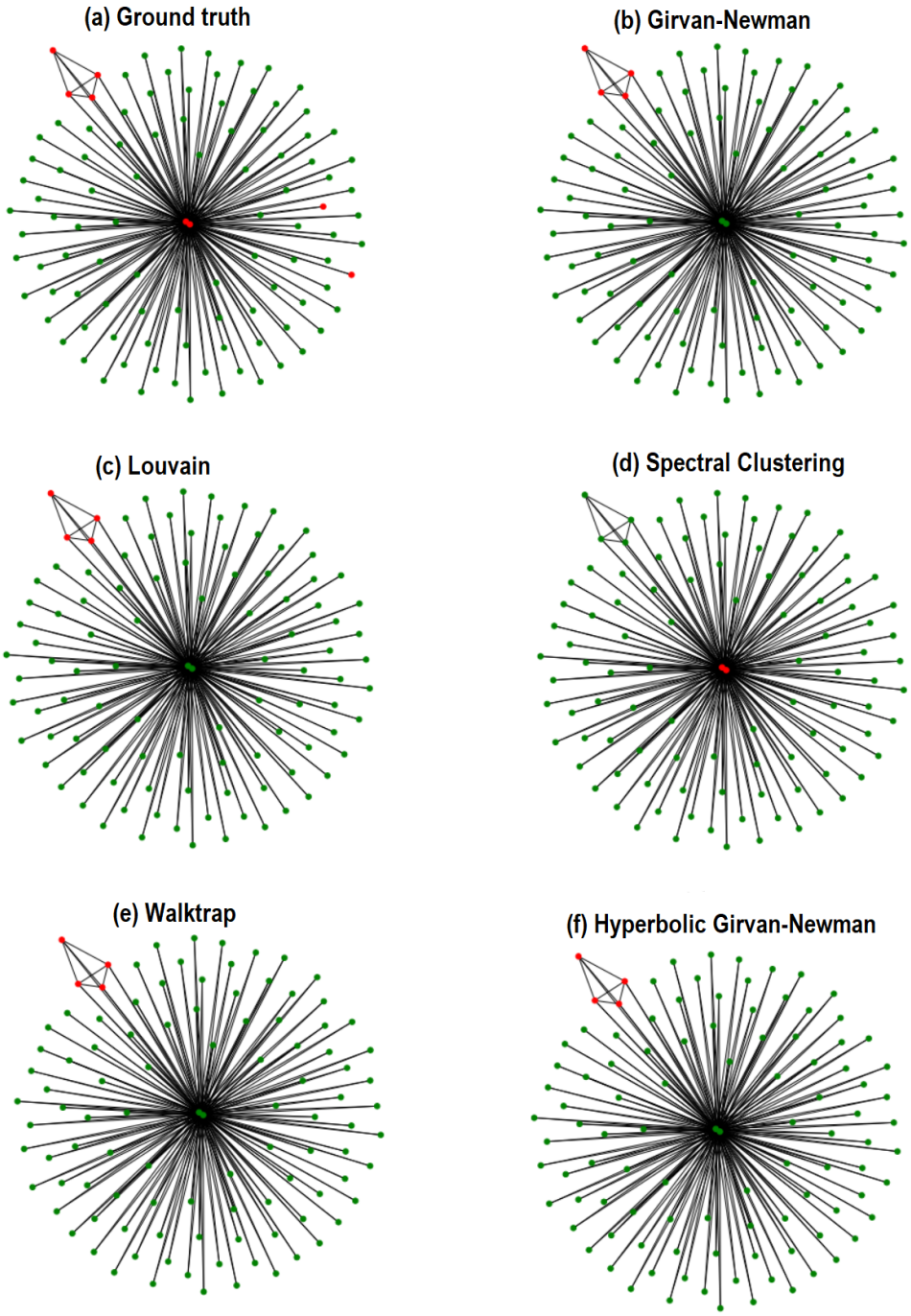

Figure 6. *Αποτελέσματα ανίχνευσης κοινοτήτων στο CTU-13*

0.4.2 Πειράματα στο σύνολο δεδομένων Kitsune

Περιγραφή και προεπεξεργασία δεδομένων

Το σύνολο δεδομένων Kitsune Network Attack Dataset [33] αποτελείται από 9 καταγραφές κίνησης δικτύων με ποικίλες επιθέσεις. Χρησιμοποιήσαμε μια καταγραφή όπου ένα δίκτυο συσκευών Διαδικτύου-των-Πραγμάτων μολύνθηκε με λογισμικό Mirai μετατρέποντας τις μολυσμένες συσκευές σε bots.

Η καταγραφή ήταν σε μορφή pcap και περιείχε 73MB από 764137 πακέτα. Κρατήσαμε μόνο τα πακέτα IPv4 οπότε έμειναν 197701. Από κάθε πακέτο κρατήσαμε 4 χαρακτηριστικά: χρόνος, διεύθυνση πηγής, διεύθυνση προορισμού και χαρακτηρισμός, ο οποίος προερχόταν από διαφορετικό αρχείο csv.

Περίοδος εκμάθησης

Η περίοδος εκμάθησης που χρησιμοποιήσαμε ήταν η πρώτη ώρα πακέτων όπου δεν είχε ξεκινήσει ακόμα η επίθεση. Ο συνολικός αριθμός πακέτων ήταν 90476 (46% των δεδομένων) με 76 διευθύνσεις IP. Χωρίσαμε τα δεδομένα εκμάθησης σε παράθυρα 10 δευτερολέπτων και επιλέξαμε 25 τυχαία παράθυρα. Σε καθένα από αυτά δημιουργήσαμε το γράφο αλληλεπίδρασης και μετρήσαμε το βαθμό 10 τυχαίων κόμβων ανά γράφο, δημιουργώντας την κατανομή αναφοράς. Επιλέχθηκε το μοντέλο τυχαίου γράφου με παράμετρο $\hat{\lambda} = 2.664$.

Ανίχνευση ανωμαλιών

Η περίοδος αξιολόγησης είχε διάρκεια 20 λεπτών και ξεκίνησε 3 λεπτά μετά το τέλος της εκπαίδευσης. Περιείχε 33903 πακέτα, 16829 από τα οποία ήταν απόbots και ξεκίνησαν να εμφανίζονται λίγο μετά τα 10 λεπτά. Η περίοδος αξιολόγησης χωρίστηκε σε παράθυρα 10 δευτερολέπτων, τα οποία συγκρίθηκαν με την κατανομή εκμάθησης. Τα αποτελέσματα φαίνονται στην εικόνα 7. Βλέπουμε μια ανοδική αλλαγή τάσης όταν εμφανίστηκαν τα bots οπότε επιλέξαμε κατώφλι 0.78.

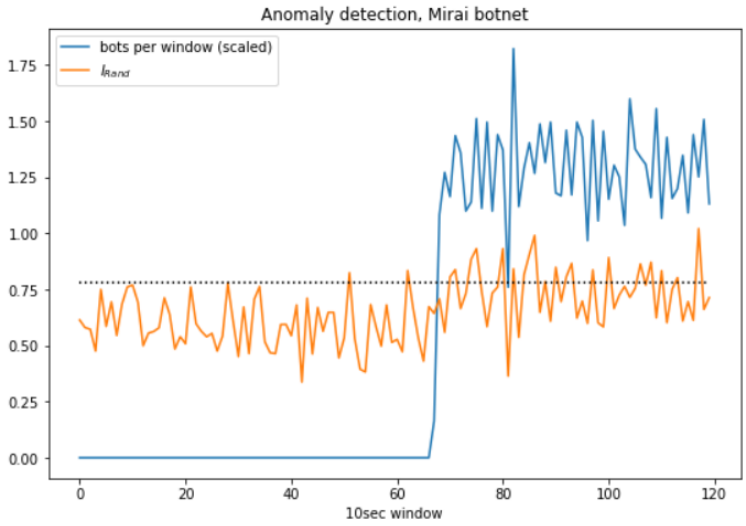

Figure 7. *Ανίχνευση ανωμαλιών στα δεδομένα Kitsune*

Δημιουργία ΓΚΣ

Εξετάσαμε 2 διαφορετικά ανώμαλα σύνολα. Το πρώτο αποτελείται από 10 ανώμαλα παράθυρα γύρω στα 15 λεπτά του συνόλου αξιολόγησης, ενώ το δεύτερο αποτελείται από τα 20 πρώτα ανώμαλα παράθυρα.

Το ανώμαλο σύνολο στα 15 λεπτά περιείχε 44 διευθύνσεις, από τις οποίες 18 ήταν bots. Ο γράφος αλληλεπίδρασης φαίνεται στο σχήμα 8. Με κατώφλι $\tau = 20$ επιλέχθηκαν 5 κεντρικοί κόμβοι, όλοι bots, και με κατώφλι $\tau_\rho = 0.65$ δημιουργήθηκε το ΓΚΣ του σχήματος 9, με 27 διευθύνσεις και 13 bots.

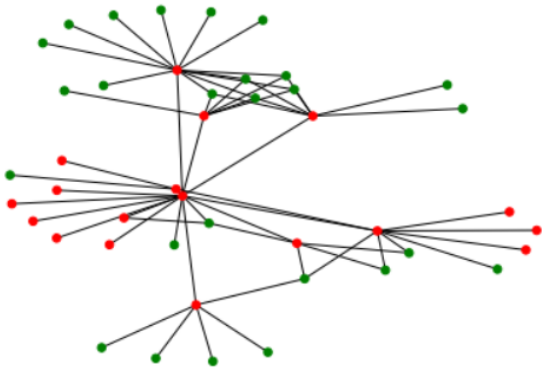

Figure 8. *Γράφος αλληλεπίδρασης των 10 ανώμαλων παραθύρων*

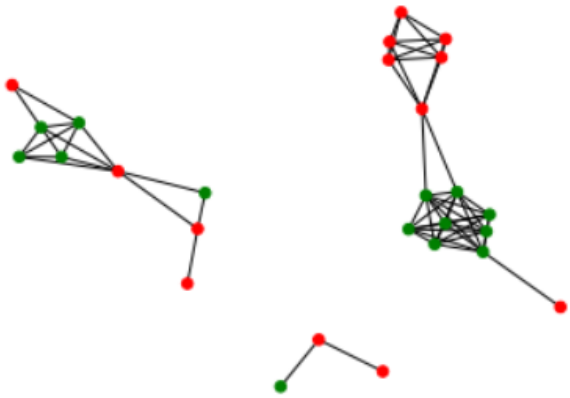

Figure 9. *ΓΚΣ των 10 ανώμαλων παραθύρων*

Το ανώμαλο σύνολο των πρώτων 20 ανώμαλων παραθύρων περιείχε 52 διευθύνσεις, από τις οποίες 18 ήταν bots. Χρησιμοποιήθηκε κατώφλι $\tau = 20$ για την επιλογή 4 κεντρικών κόμβων, όλοι bots. Με κατώφλι $\tau_\rho = 0.47$ δημιουργήθηκε ΓΚΣ με 29 διευθύνσεις και 13 bots, όπως φαίνεται στο σχήμα 10.

Σύγκριση αλγορίθμων ανίχνευσης κοινοτήτων

Στο ΓΚΣ του σχήματος 9 εφαρμόσαμε τους αλγορίθμους ανίχνευσης κοινοτήτων στις δύο μεγαλύτερες συνεκτικές συνιστώσες. Τα αποτελέσματα για τη μεγαλύτερη συνεκτική

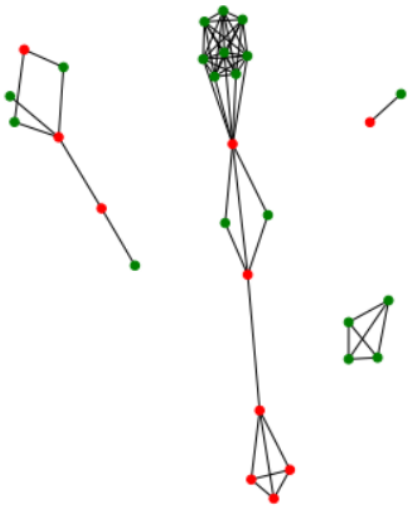

Figure 10. *ΓΚΣ των πρώτων 20 ανώμαλων παραθύρων*

συνιστώσα φαίνονται στον πίνακα 2 και στην εικόνα 11. Παρατηρούμε ότι όλοι οι αλγόριθμοι εντοπίζουν την μεγάλη κοινότητα bots, αλλά μόνο ο Spectral Clustering ανιχνεύει και το μοναδικό bot κάτω δεξιά.

Αλγόριθμοι	**NMI**	**F1-score**
Girvan-Newman	0.71	0.94
Louvain	0.71	0.94
Spectral Clustering	1	1
Walktrap	0.71	0.94
HGN	0.71	0.94

Table 2. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στη μεγαλύτερη συνεκτική συνιστώσα του ΓΚΣ*

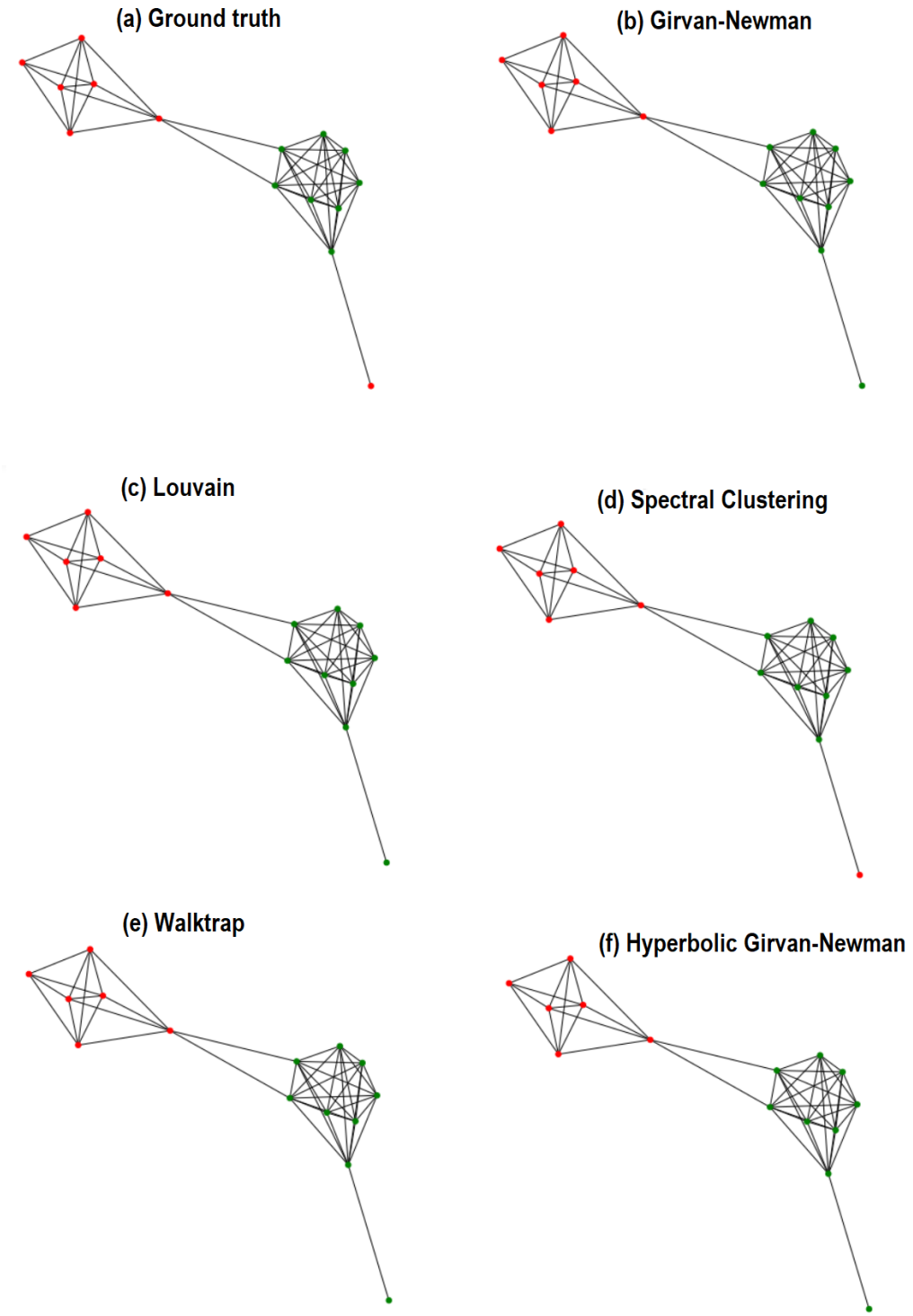

Figure 11. *Αποτέλεσμα ανίχνευσης κοινοτήτων στη μεγαλύτερη συνεκτική συνιστώσα του ΓΚΣ*

Τα αποτελέσματα για τη 2η συνεκτική συνιστώσα φαίνονται στον πίνακα 3 και στην εικόνα 12. Όλοι οι αλγόριθμοι εκτός του Louvain εντοπίζουν 2 από τα 4 bots, ενώ ο Louvain εντοπίζει 3. Όλοι έχουν 1 ψευδώς θετικό αποτέλεσμα.

Αλγόριθμος	**NMI**	**F1-score**
Girvan-Newman	0.08	0.57
Louvain	0.23	0.75
Spectral Clustering	0.08	0.57
Walktrap	0.08	0.57
HGN	0.08	0.57

Table 3. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στη 2η συνεκτική συνιστώσα του ΓΚΣ*

Στο ΓΚΣ των 20 πρώτων ανώμαλων παραθύρων εφαρμόσαμε τους αλγορίθμους μόνο στην μεγαλύτερη συνεκτική συνιστώσα. Τα αποτελέσματα φαίνονται στον πίνακα 4 και στην εικόνα 13. Εδώ παρατηρούμε μεγαλύτερη απόκλιση μεταξύ των αλγορίθμων και χειρότερα αποτελέσματα από πριν, που είναι αναμενόμενο καθώς το ανώμαλο σύνολο περιέχει περισσότερες αλληλεπιδράσεις άκακων κόμβων. Καλύτερα αποτελέσματα φαίνεται να έχει ο Spectral Clustering, που εντοπίζει όλα τα bots εκτός από 1 με μόνο 2 ψευδώς θετικά αποτελέσματα. Οι αλγόριθμοι Girvan-Newman χωρίζουν αρκετά καλά το γράφο σε κοινότητες, που φαίνεται στο ψηλό NMI, αλλά αναγνωρίζουν τη λάθος κοινότητα ως bots οπότε έχουν πολύ χαμηλό F1-score. Οι αλγόριθμοι Louvain, Walktrap εντοπίζουν περισσότερες από 2 κοινότητες και επιλέγουν τη μεσαία ως bots. Αυτό αναδεικνύει ότι οι μεσαίοι κόμβοι έχουν μεγάλο μέτρο αλληλεπίδρασης με κεντρικούς κόμβους, γιαυτό και επιλέχθηκαν ως bots από όλους τους αλγορίθμους.

Αλγόριθμος	**NMI**	**F1-score**
Girvan-Newman	0.53	0.13
Louvain	0.02	0.63
Spectral Clustering	0.3	0.57
Walktrap	0.01	0.22
HGN	0.53	0.13

Table 4. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στο ΓΚΣ των πρώτων 20 ανώμαλων παραθύρων*

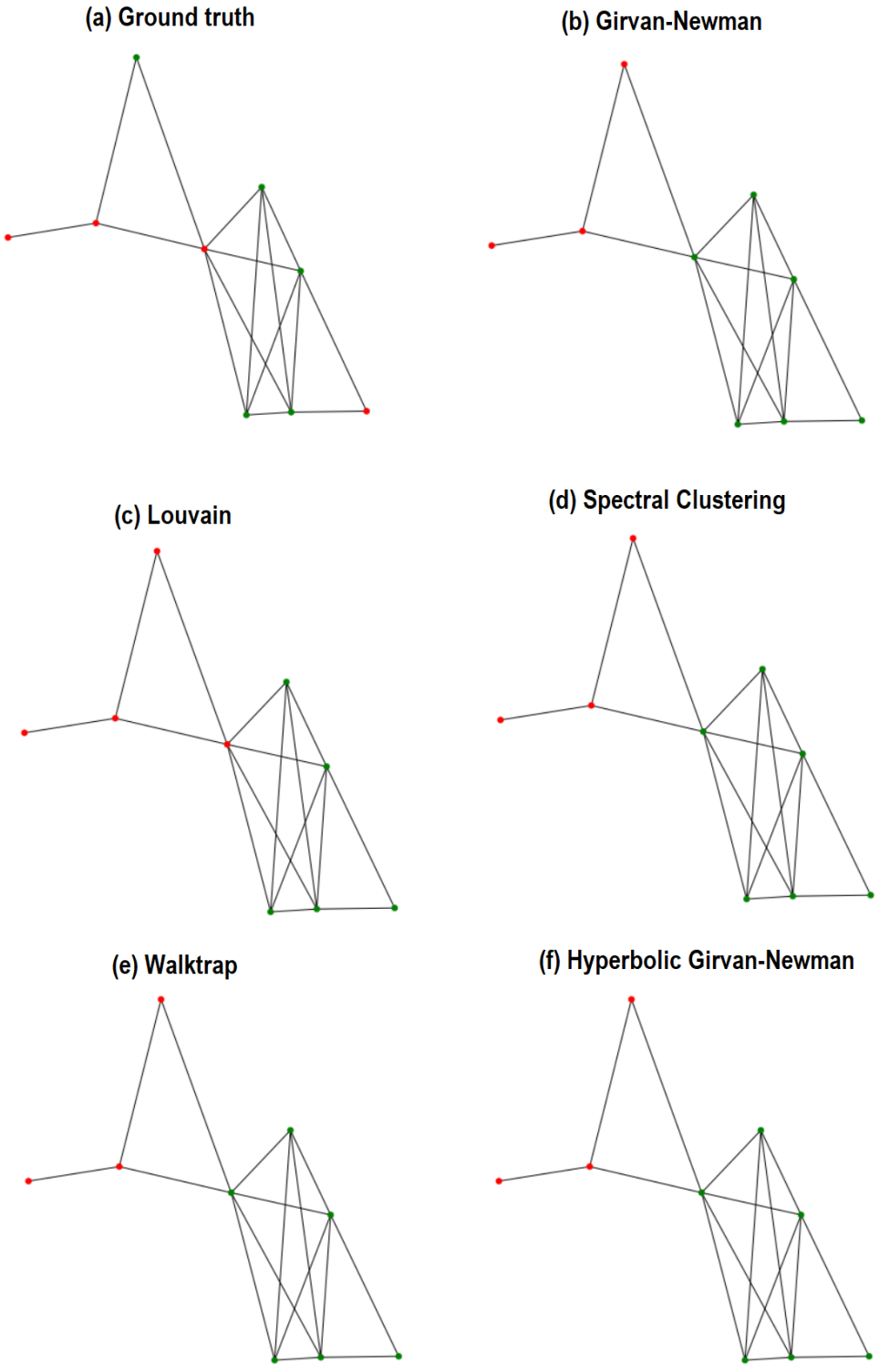

Figure 12. *Αποτελέσματα ανίχνευσης κοινοτήτων στη 2η συνεκτική συνιστώσα του ΓΚΣ*

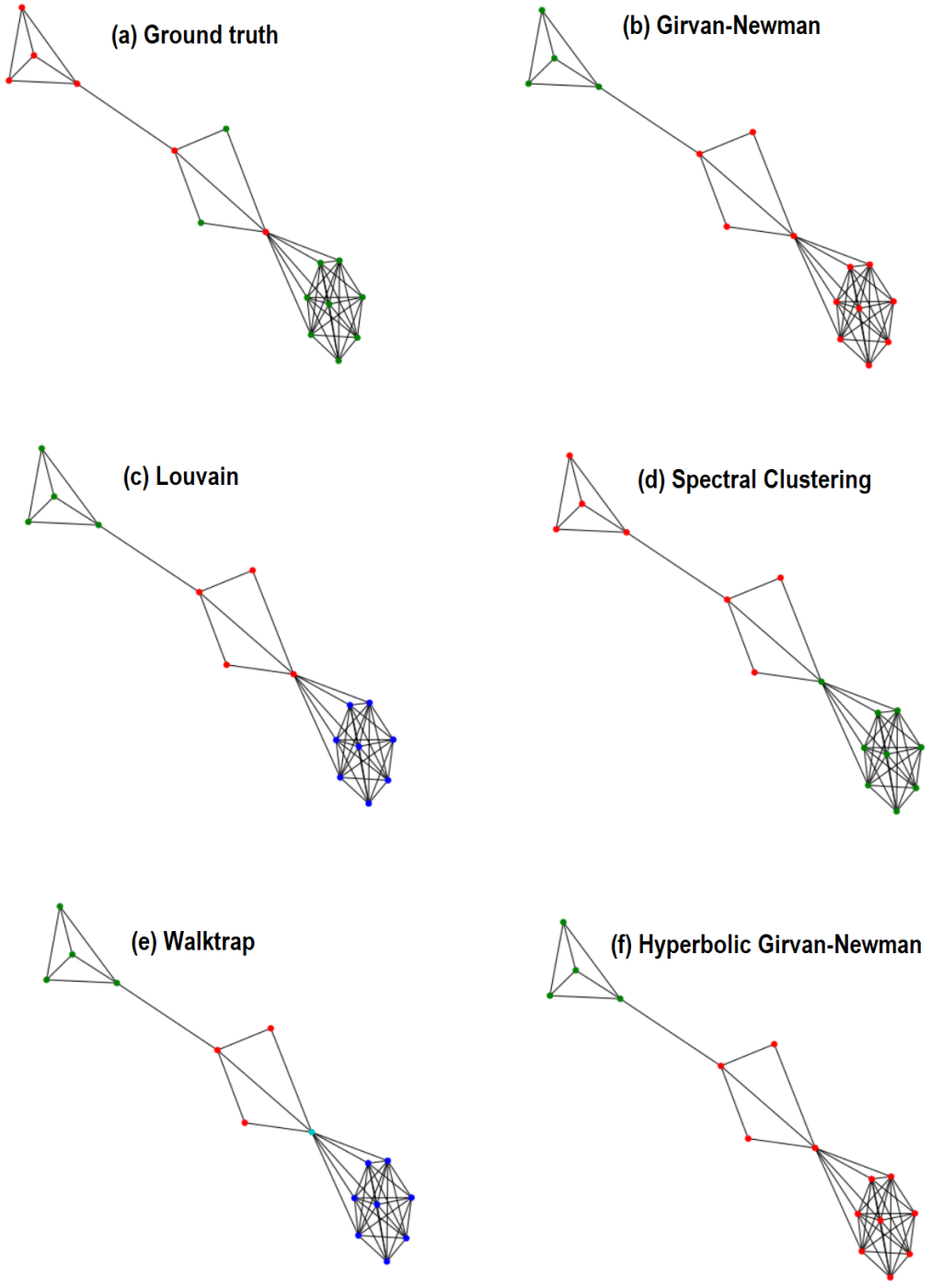

Figure 13. *Αποτελέσματα ανίχνευσης κοινοτήτων στο ΓΚΣ των πρώτων 20 ανώμαλων παραθύρων*

0.4.3 Πειράματα στα σύνολα δεδομένων CAIDA, Simpleweb

Περιγραφή και προεπεξεργασία δεδομένων

Για το τελευταίο πείραμα ανακατέψαμε κυκλοφορία από δύο διαφορετικά σύνολα δεδομένων, ένα από τα οποία περιείχε κακόβουλα πακέτα και ένα που περιείχε φυσιολογική κυκλοφορία. Για τα πακέτα του botnet χρησιμοποιήσαμε το CAIDA USCD "DDoS Attack 2007" Dataset [34]. Περιέχει 1 ώρα κακόβουλων πακέτων από μια κατανεμημένη επίθεση άρνησης πόρων. Χρησιμοποιήσαμε κυκλοφορία από τα πρώτα 5 λεπτά, μεγέθους 8.6MB. Για την φυσιολογική κίνηση χρησιμοποιήσαμε το 6ο ίχνος του University of Twente traffic traces data repository (simpleweb) [35]. Το ίχνος έχει μέγεθος 192MB και μετρήθηκε σε μια μικρή εκπαιδευτική οργάνωση με 100 σταθμούς εργασίας.

Και τα δύο αρχεία που χρησιμοποιήσαμε ήταν σε μορφή pcap, και τα μετατρέψαμε σε csv μέσω του Wireshark. Για να τα ανακατέψουμε, αλλάξαμε την χρονική στιγμή κάθε πακέτου ώστε να εκφράζει τα δευτερόλεπτα που πέρασαν από την καταγραφή του πρώτου πακέτου. Από κάθε πακέτο κρατήσαμε τα χαρακτηριστικά χρόνου εμφάνισης, διεύθυνση πηγής και διεύθυνση προορισμού, και προσθέσαμε επιπλέν χαρακτηρισμό 0 στα άκακα πακέτα και 1 στα κακόβουλα.

Από τα δεδομένα simpleweb κρατήσαμε την πρώτη ώρα κυκλοφορίας, με 329967 πακέτα και 743 διευθύνσεις IP. Τα δεδομένα CAIDA περιείχαν 166448 πακέτα από 136 διευθύνσεις IP, αλλά κρατήσαμε τυχαία μόνο 6000 πακέτα έτσι ώστε να μην είναι πάρα πολύ πιο πυκνά από τη φυσιολογική κυκλοφορία. Αντικαταστήσαμε τις 136 διευθύνσεις των πακέτων CAIDA τυχαία σε διευθύνσεις simpleweb και εισάγαμε την επίθεση μεταξύ των δευτερολέπτων 2000 και 2300. Η αντιστοίχηση διευθύνσεων έγινε με 2 τρόπους, μια φορά αντιστοιχίζοντας 3 διευθύνσεις bot σε 1 κανονική καταλήγοντας σε 45 διευθύνσεις bot, και μια φορά αντιστοιχίζοντας 5 διευθύνσεις bot σε μια κανονική καταλήγοντας σε 27 bots.

Περίοδος εκμάθησης

Η περίοδος εκμάθησης ήταν τα πρώτα 20 λεπτά κίνησης, με 198440 πακέτα (59% των πακέτων) και 539 διευθύνσεις IP. Τα δεδομένα εκμάθησης χωρίστηκαν σε παράθυρα 10 δευτερολέπτων και επιλέξαμε τυχαία 50 από αυτά. Από τον γράφο αλληλεπίδρασης κάθε πακέτου μετρήσαμε το βαθμό 10 τυχαίων κόμβων για να δημιουργήσουμε την κατανομή αναφοράς. Επιλέχθηκε το μοντέλο τυχαίου γράφου με παράμετρο $\hat{\lambda} = 1.628$.

Ανίχνευση ανωμαλιών

Η περίοδος αξιολόγησης είχε διάρκεια 20 λεπτά και ξεκίνησε 5 λεπτά μετά το τέλος της εκπαίδευσης (1500-2700 δευτερόλεπτα). Περιείχε 32763 πακέτα, από τ οποία 6000 ήταν κακόβουλα στο διάστημα 2000-2300. Το αποτέλεσμα ανίχνευσης ανωμαλιών για τα 45 bots φαίνεται στο σχήμα 14 με κατώφλι 1.5 και για τα 27 bots στο σχήμα 15 με κατώφλι 1.2. Όπως ήταν αναμενόμενο η ανίχνευση ανωμαλιών είναι πολύ πιο αποτελεσματική στην περίπτωση με περισσότερα bots.

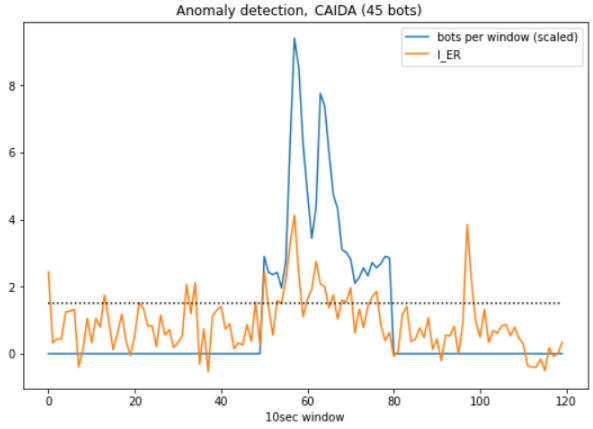

Figure 14. *Ανίχνευση ανωμαλιών στο μικτό σύνολο δεδομένων με 45 bots*

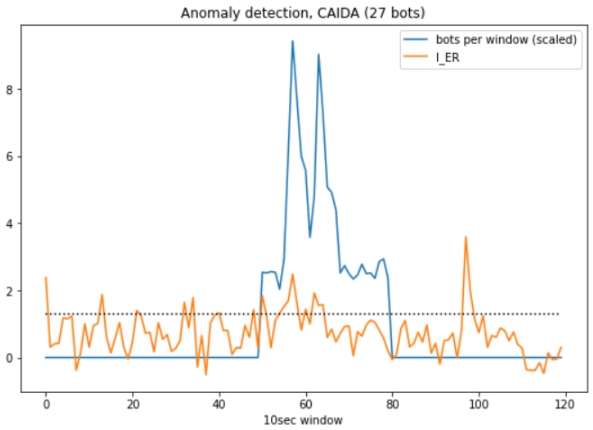

Figure 15. *Ανίχνευση ανωμαλιών στο μικτό σύνολο δεδομένων με 27 bots*

Δημιουργία ΓΚΣ

Το ανώμαλο σύνολο της περίπτωσης με 45 bots περιείχε 94 διευθύνσεις IP, από τις οποίες 44 ήταν bots. Χρησιμοποιήσαμε $\tau = 30$, επιλέγοντας 4 κεντρικούς κόμβους, από τους οποίους ένας ήταν bot και οι άλλοι θύματα. Θέσαμε $\tau_\rho = 0.38$ και καταλήξαμε σε ΓΚΣ 48 κόμβων, από τους οποίους 41 ήταν bots. Το ΓΚΣ φαίνεται πάνω αριστερά στην εικόνα 16.

Εφόσον η ανίχνευση ανωμαλιών στην περίπτωση με 27 bots απέτυχε, επιλέξαμε χειροποίητα ανώμαλο σύνολο με 101 κόμβους από τους οποίους 20 ήταν bots. Χρησιμοποιήσαμε $\tau = 30$ επιλέγοντας 5 κεντρικούς κόμβους, από τους οποίους 2 ήταν bots. Το ΓΚΣ δημιουργήθηκε με $\tau_\rho = 0.3$ και περιείχε 31 κόμβους με 15 bots. Το ΓΚΣ φαίνεται πάνω αριστερά στην εικόνα 17.

Σύγκριση αλγορίθμων ανίχνευσης κοινοτήτων

Τα αποτελέσματα της ανίχνευσης κοινοτήτων για το πρώτο ΓΚΣ φαίνονται στον πίνακα 5 και στην εικόνα 16. Στον HGN χρησιμοποιήσαμε 3 κόμβους ορόσημα και μέγεθος δέσμης 1. Παρατηρούμε ότι οι δύο αλγόριθμοι Girvan-Newman εντοπίζουν τη μεγάλη κοινότητα απόbots ενώ οι υπόλοιποι αλγόριθμοι δυσκολεύονται. Ο Spectral Clustering εντοπίζει μόνο ένα κομμάτι της κοινότητας, ενώ οι Louvain, Walktrap χωρίζουν την κοινότητα απόbots σε

μικρότερες κοινότητες και ανιχνεύουν μόνο μία από αυτές.

Αλγόριθμος	**NMI**	**F1-score**
Girvan-Newman	0.62	0.96
Louvain	0.18	0.62
Spectral Clustering	0.23	0.53
Walktrap	0	0
HGN	0.62	0.96

Table 5. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στο ΓΚΣ της περίπτωσης με 45 bots*

Στην περίπτωση των 27 bots χρησιμοποιήσαμε 10 κόμβους ορόσημα στον HGN και μέγεθος δέσμης 1. Τα αποτελέσματα φαίνονται στο σχήμα 17 και στον πίνακα 6. Πάλι καλύτερα αποτελέσματα έχουν οι Girvan-Newman αλγόριθμοι. Ο Spectral Clustering εντοπίζει όλα τα bots αλλά έχει πολλά ψευδώς θετικά αποτελέσματα, ενώ οι Louvain, Walktrap πάλι εντοπίζουν πολλές κοινότητες και ανιχνεύουν μόνο λίγους κόμβους.

Αλγόριθμος	**NMI**	**F1-score**
Girvan-Newman	0.21	0.84
Louvain	0.01	0.35
Spectral Clustering	0.03	0.77
Walktrap	0	0
HGN	0.13	0.79

Table 6. *NMI και F1-score των αλγορίθμων ανίχνευσης κοινοτήτων στο ΓΚΣ της περίπτωσης με 27 bots*

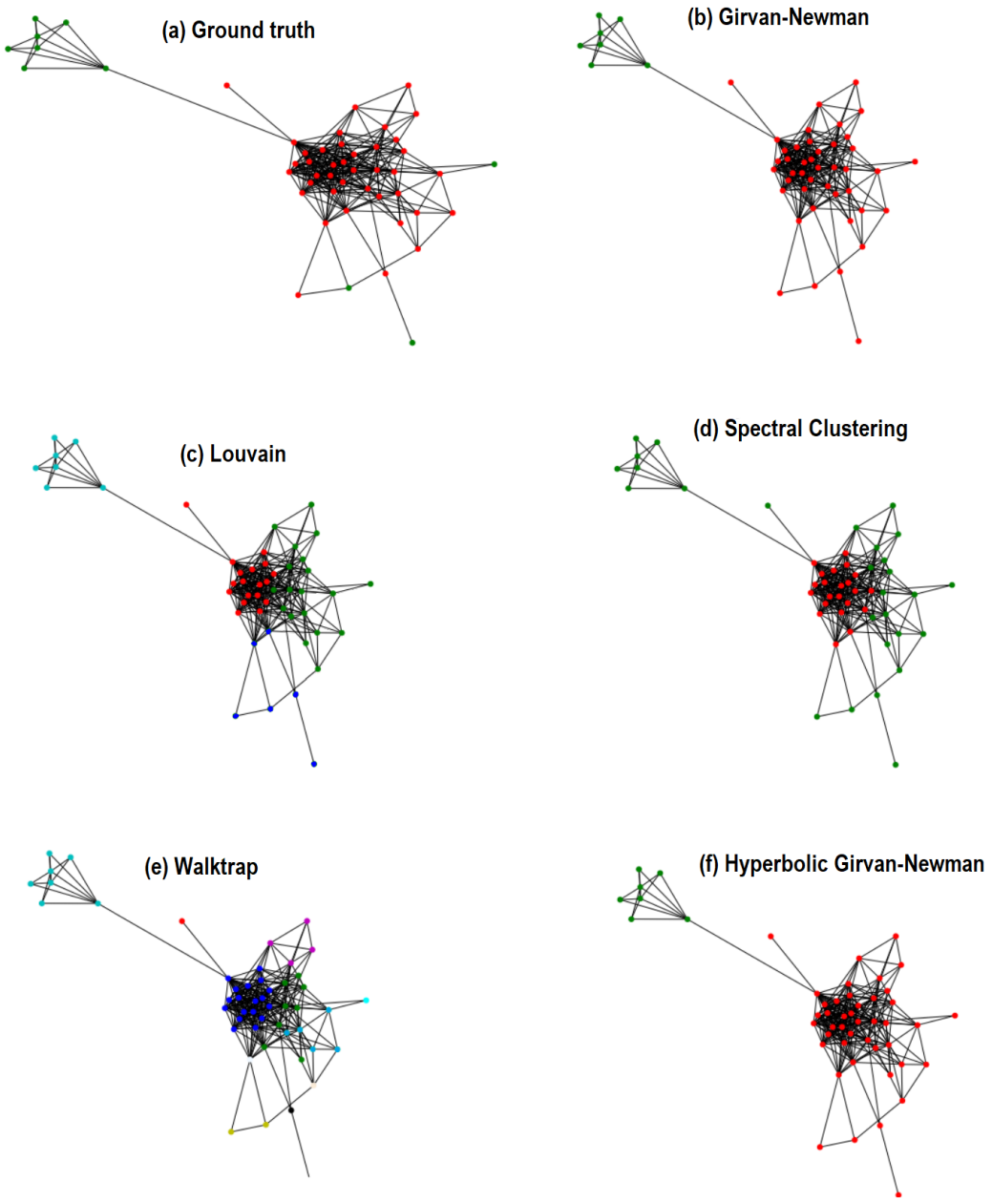

Figure 16. *Αποτέλεσμα ανίχνευσης κοινοτήτων στο ΓΚΣ της περίπτωσης με 45 bots*

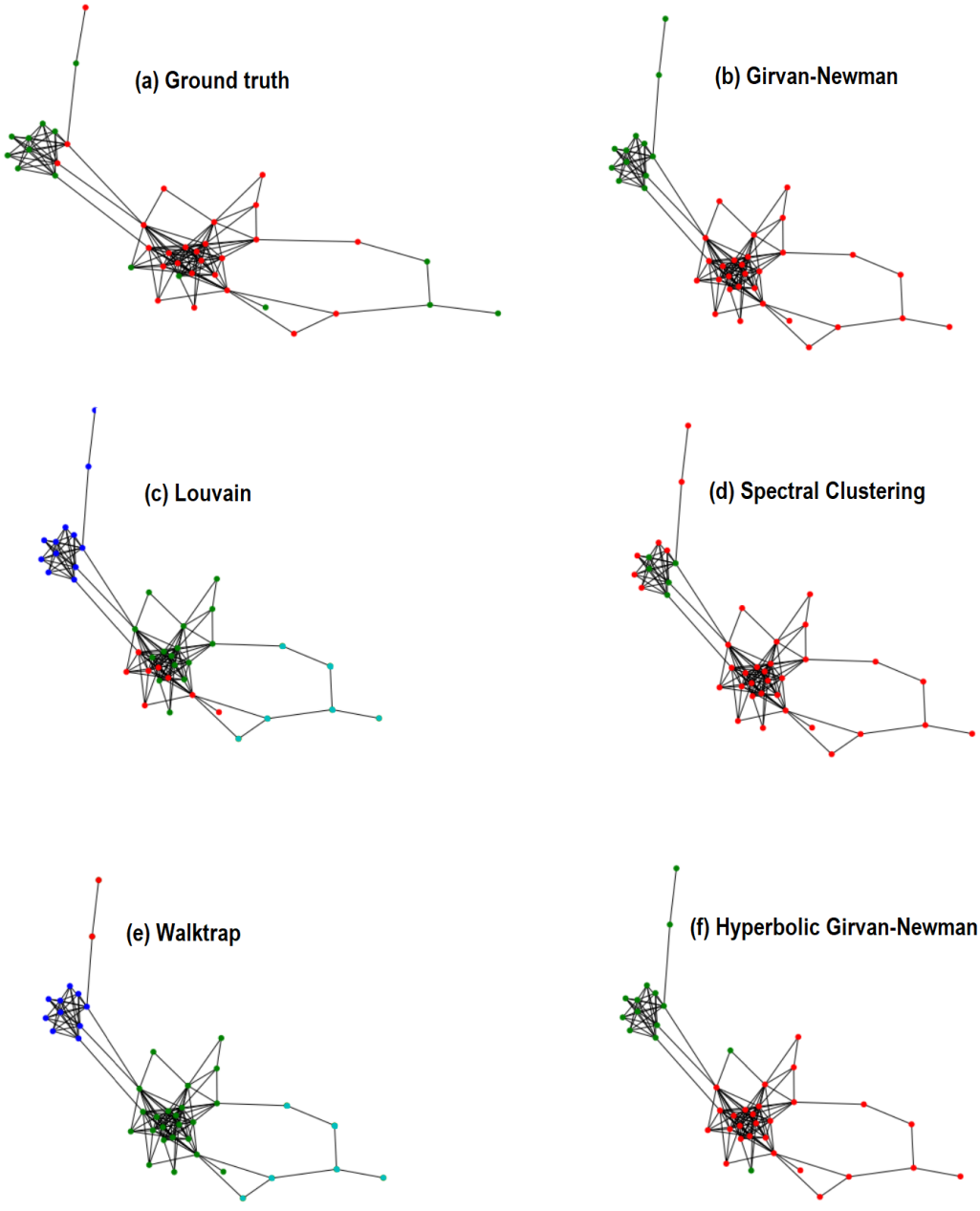

Figure 17. *Αποτέλεσμα ανίχνευσης κοινοτήτων στο ΓΚΣ της περίπτωσης με 45 bots*

0.5 Συμπεράσματα και ιδέες για παραιτέρω μελέτη

Από τα πειράματά μας επιβεβαιώσαμε ότι η ανίχνευση ανωμαλιών και botnets είναι πιο δύσκολη όταν τα bots είναι λίγα, ιδιαίτερα στην αρχή της επίθεσης. Όσον αφορά τη δημιουργία ΓΚΣ, παρατηρήσαμε ότι αν η παράμετρος τ_ρ είναι αυστηρή τότε διατηρούνται λιγότερα bots σε αυτό αλλά οι κοινότητες είναι πιο διακριτές, ενώ αν η παράμετρος δεν είναι αυστηρή τότε υπάρχουν περισσότερα bots αλλά είναι πιο δύσκολο να διακριθούν από τους άκακους κόμβους.

Για την ανίχνευση των botnets παρατηρήσαμε ότι οι αλγόριθμοι Louvain, Walktrap τείνουν να διαχωρίζουν το ΓΚΣ σε πολλές κοινότητες με αποτέλεσμα να εντοπίζουν λίγα bots. Το πρόβλημα αυτό μπορεί να επιλυθεί αν χρησιμοποιήσουμε διαφορετική στρατηγική κατηγοριοποίησης των κοινοτήτων. Αντί να επιλέξουμε μόνο μια κοινότητα ως bots, θα μπορούσαμε να εξετάζουμε κάθε κοινότητα ξεχωριστά, ίσως χρησιμοποιώντας κάποια χαρακτηριστικά του δικτύου. Με τον τρόπο αυτό οι αλγόριθμοι που εντοπίζουν περισσότερες από δύο κοινότητες ίσως έχουν καλύτερα αποτελέσματα, και θα μπορούσαμε να τρέξουμε και τους Girvan-Newman, Spectral Clustering με μεγαλύτερο αριθμό κοινοτήτων για σύγκριση. Όμως η στρατηγική αυτή προσθέτει επιπλέον παραμέτρους για να λύσει ένα πρόβλημα που δεν εμφανίζεται αν απλώς επιλεγεί διαφορετικός αλγόριθμος.

Μεταξύ των αλγορίθμων Girvan-Newman, Spectral Clustering ο δεύτερος είχε χειρότερα αποτελέσματα στα μικτά δεδομένα, ενώ ο πρώτος είχε μέτρια αποτελέσματα στα πρώτα πειράματα. Σε γενικές γραμμές οι αλγόριθμοι Girvan-Newman ήταν αρκετά αξιόπιστοι αν το ΓΚΣ ήταν καλά φτιαγμένο. Αξίζει παραιτέρω μελέτη σε μεγαλύτερα ανώμαλα σύνολα, καθώς και στην επιλογή των παραμέτρων τ, τ_ρ ώστε το ΓΚΣ να περιλαμβάνει περισσότερα bots με μικρότερη ποινή στην ακρίβεια εντοπισμού κοινοτήτων. Μεταξύ των Girvan-Newman, HGN αξίζει η επιλογή του HGN αν έχει προτεραιότητα η ταχύτητα (για παράδειγμα στην αρχή της επίθεσης) καθώς τα αποτελέσματά τους είναι σχεδόν ίδια.

Chapter 1

Introduction

Botnets are groups of internet-connected devices that have been compromised and are controlled by a malicious actor. They are used to perform Distributed Denial-of-Service (DDoS) attacks, steal data such as account or bank information, spam or click fraud. DDoS attacks overload the target with superfluous requests, resulting in it being unable to process regular requests and therefore its service becoming unavailable.

Because of the popularity and severity of botnet-led DDoS attacks, botnet detection has become an important cybersecurity problem. In order to avoid the known detection techniques, botnets constantly evolve, but their behavior continues to exhibit certain patterns that allow their detection. Overall, it is more difficult to detect an attack and identify compromised nodes during the early stages of the attack, but the earlier the botnet is caught, the easier it is to defend against it.

1.1 Thesis contribution

This thesis examines an existing botnet detection method combining anomaly and community detection. Initially, the method gathers normal traffic during a training period, which is then used to determine whether new traffic is anomalous based on its deviation from the reference. The anomalous traffic is then processed in order to group bots together based on the correlation of their interactions. The contribution of this thesis is to examine and compare multiple different community detection algorithms on the final stage of the method.

Three different botnet attacks are used to evaluate the algorithms, which are then compared in terms of their accuracy in identifying the compromised nodes. Traffic from the early stages of each attack is used in the experiments, since the later stages are straightforward to detect due to the large volume of traffic. Five community detection algorithms are compared, including Hyperbolic Girvan-Newman, an algorithm that embeds the network into hyperbolic space for efficiency of calculations, and the benefits and drawbacks of each algorithm are analysed.

1.2 Thesis outline

The rest of this thesis is organised into four chapters:

- Chapter 2: Theory, provides theoretical background on graph theory, complex networks, machine learning, community detection and botnets.
- Chapter 3: Methodology, explains in detail the algorithm used.
- Chapter 4: Experiments, presents the experimentation carried out for this diploma thesis, including information about the code and datasets used.
- Chapter 5: Epilogue, contains a summary of the thesis, some conclusions and ideas for further improvements.

Chapter 2

Theory

This section contains the theoretical background needed before our methodology and experiments are explained. It is split into five sections: graph theory, complex networks, machine learning, community detection and anomaly detection.

2.1 Graph theory

In this section we provide a quick introduction to graph theory by defining some basic concepts we will use throughout the thesis.

2.1.1 Basic definitions

A graph, or network, $G = \{V, E\}$ is a mathematical structure containing a set of vertices (or nodes) V and a set of edges E. An edge $(u, v) \in E$ represents a connection, or a relationship, between two nodes $u, v \in V$. Graphs are used in a variety of applications such as road networks, the world wide web, social media connections, etc.

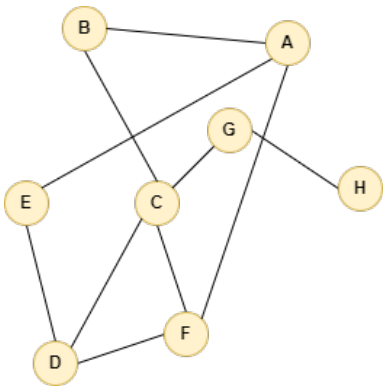

Figure 2.1. *Example of a graph with 8 nodes and 10 edges*

Directed graphs A graph may be *directed* or *undirected*. In directed graphs each edge is oriented, so it has a source node and a destination node. In undirected graphs edges are not oriented. Figure 2.1 shows an example of an undirected graph.

Weighted graphs A graph is called *weighted* if each edge has a weight assigned to it. The weight represents quantities such as distance or cost.

In figure 2.2 we can see an example of a weighted undirected graph and an unweighted directed graph.

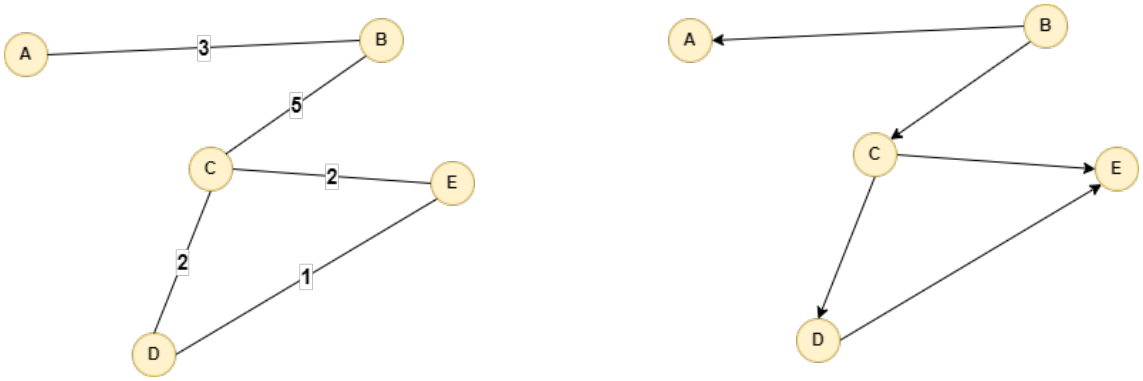

Figure 2.2. *Left: a weighted undirected graph*
Right: an unweighted directed graph

Neighbourhood Two nodes are *neighbours*, or *adjacent*, if they are connected by an edge. The *neighbourhood* N_u of a node u is the set of nodes adjacent to it.

$$N_u = \{v \in V : (u, v) \in E\} \tag{2.1}$$

Clique A *clique* is a subset of V where every pair of nodes is connected by an edge. A clique of k nodes has $\frac{k(k-1)}{2}$ edges. A clique of 4 nodes is shown in figure 2.3.

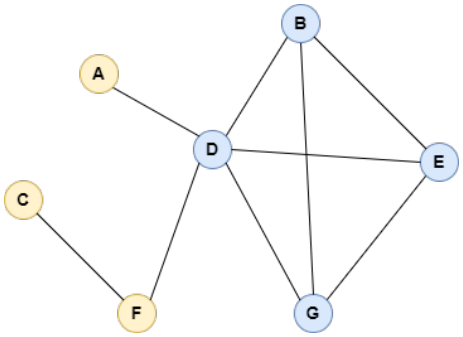

Figure 2.3. *Nodes B,D,E,G form a clique*

2.1.2 Graph representations

In this section we outline the two most common graph representations. There are benefits to using either, which depend on the situation.

Adjacency matrix

The *adjacency matrix* A of a graph with n vertices is a square $n \times n$ matrix where each row and column corresponds to a vertex. If the edge (u, v) connecting the vertices u, v exists then $A_{uv} = 1$, else $A_{uv} = 0$. The main advantage of adjacency matrices is that they provide an immediate way to add, remove, or check if an edge exists. Inserting new

vertices however is costly, and their $n \times n$ space requirement does not scale well for large graphs.

Some properties of the adjacency matrix:

- In undirected graphs the adjacency matrix is symmetric
- In weighted graphs the elements of the adjacency matrix are the weights of the edges
- In graphs without self-edges the diagonal of the adjacency matrix is all zeros

Adjacency list

The *adjacency list* of a graph with n nodes is a set of n lists. Each list's elements are the neighbours of a specific node. Adjacency lists are more space-efficient than adjacency matrices, especially in sparse graphs where the number of edges is small. They provide a quicker way to iterate over all edges, but checking for the existence of a specific edge is costlier than in adjacency matrices.

2.2 Complex networks

Complex networks are graphs with a complicated structure whose properties depend largely on the way the nodes connect and interact with each other [7]. Most real-life networks are complex, such as social networks or computer networks [8], [9]. This section presents some common metrics used to study the structure and behaviour of complex networks.

2.2.1 Degree distribution

The *degree* of a node is defined as the number of nodes adjacent to it, or the size of its neighbourhood. The *degree distribution* of a network is the probability distribution that describes the probability of a node in the network having a specific degree.

In weighted graphs, the *strength* of a node is the sum of weights of the edges connected to it. In directed graphs, we define an *in-degree* and an *out-degree* (similarly *in-strength* and *out-strength*) using only edges where the given node is the target or the destination node.

The degree distribution can give some insight on the type and structure of the graph. In section 2.2.6 we compare the degree distribution of two different network models (figures 2.7 and 2.10).

2.2.2 Average path length

A *path* is defined as a sequence of nodes where every two consecutive nodes are linked by an edge. If the nodes in a path are distinct, it is called a *simple path*. The length of the shortest path between two nodes is called their *distance*.

The *average path length* of a network is the mean of the lengths of the shortest paths between every pair of nodes in the network. If $d(u, v)$ is the distance between nodes u, v

of a graph G with $|V| = n$, then the average path length l_G is calculated by:

$$l_G = \frac{1}{n(n-1)} \sum_{u,v \in V} d(u,v) \tag{2.2}$$

The average path length is a way to estimate how quickly information travels inside a network. It can also be used to predict the spread of epidemics [36].

2.2.3 Clustering coefficient

In many complex networks, nodes tend to cluster together and create local communities with lots of connections between them. This behaviour is measured by the *clustering coefficient*. Quantifying the existence and strength of communities is important in applications such as recommendation systems [11] or epidemiology [12], [13]. The clustering coefficient can be measured either across the whole network or on a single node.

Global clustering coefficient

The *global clustering coefficient* C is a way to quantify how clustered a network is as a whole.

$$C = \frac{3 \times number\ of\ triangles}{number\ of\ triplets} \tag{2.3}$$

Local clustering coefficient

The *local clustering coefficient* refers to a single node and measures how closely connected its neighbours are. It is calculated by dividing the number of connections between the node's neighbours to the total possible connections between them (which is the case if they formed a clique). If $k_u = |N_u|$ then the local clustering coefficient C_u of node u is:

$$C_u = \frac{2 \times |\{e_{ij} : i,j \in N_u,\ e_{ij} \in E\}|}{k_u(k_u - 1)} \tag{2.4}$$

Figure 2.4 shows the local clustering coefficient of a node, marked in green, in three different scenarios. On the left, all three edges exist between the neighbours of the green node, so its local clustering coefficient is 1. In the middle, only one of the three edges exist, so the local clustering coefficient is $\frac{1}{3}$. On the right, no edges exist between the three neighbours, so the local clustering coefficient of the green node is 0.

Average clustering coefficient

The *average clustering coefficient* is an alternative to the global clustering coefficient and is measured by calculating the mean of the local clustering coefficients of all n nodes in the network.

$$\overline{C} = \frac{1}{n} \sum_{i=1}^{n} C_i \tag{2.5}$$

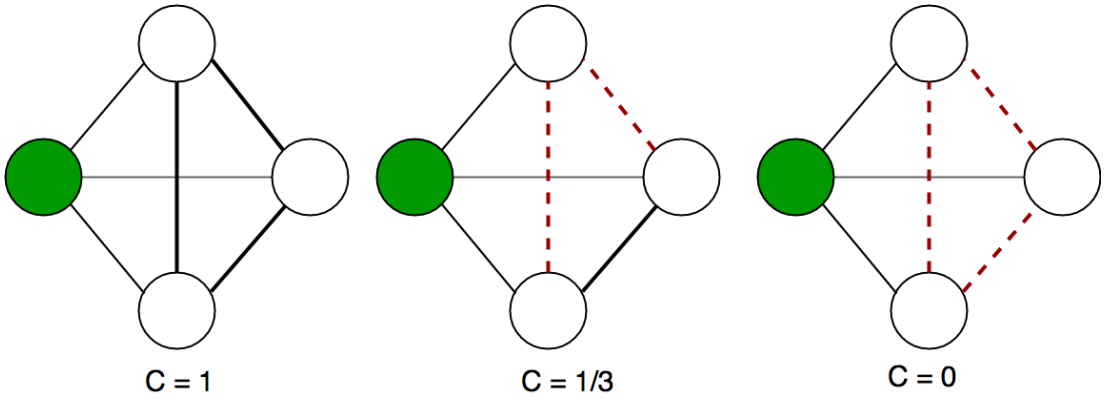

Figure 2.4. *Example of the calculation of the local clustering coefficient of the green node in three different scenarios [1]*

2.2.4 Centralities

Node centralities are metrics that indicate the importance of nodes in a network. There are multiple ways to measure the centrality of nodes, as different problems and networks view different properties as more important [37], [38], [39]. In this section we present some of the most popular centrality metrics.

Degree centrality

The simplest centrality measure is the *degree centrality*, which is equal to a node's degree. Nodes with many neighbours are deemed more important, because they have many connections and control the information flow. If A the adjacency matrix of a network $G = (V, E)$ with n nodes, then the degree centrality of node $i \in V$ is:

$$C_D(i) = \sum_{j=1}^{n} a_{ij} \tag{2.6}$$

Sometimes the normalised degree centrality is used:

$$C'_D(i) = \frac{1}{n-1} \sum_{j=1}^{n} a_{ij} \tag{2.7}$$

Closeness centrality

Closeness centrality values how close a node is to all other nodes. It is a way to estimate how quickly information travels from a node to the rest of the network. It is measured by dividing the total number of nodes minus the source with the distance of the source to all other nodes. For this reason closeness centrality is only defined if the network is connected, which means a path should exist between any pair of nodes.

$$C_C(i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)} \tag{2.8}$$

Betweenness centrality

Betweenness centrality is a metric that estimates the importance of a node based on the number of shortest paths between all nodes in the network that pass through it. A node with high betweenness centrality controls a large amount of the information that flows through the network, since many paths go through it, whereas a node with low betweenness centrality can be bypassed via different nodes. For a node $u \in V$, the betweenness centrality $C_B(u)$ is defined by iterating through every pair of nodes $i,j \in V$ and dividing the number of shortest paths $\sigma_{ij}(u)$ between i and j that pass through u by the total number of shortest paths σ_{ij} between i and j.

$$C_B(u) = \sum_{i,j \neq u \in V} \frac{\sigma_{ij}(u)}{\sigma_{ij}} \tag{2.9}$$

Figure 2.5 shows a graph whose nodes are coloured based on their betweenness centrality, from blue denoting high values to red denoting low values. As expected, the nodes on the outside of the graph have lower betweenness centrality than the more central ones.

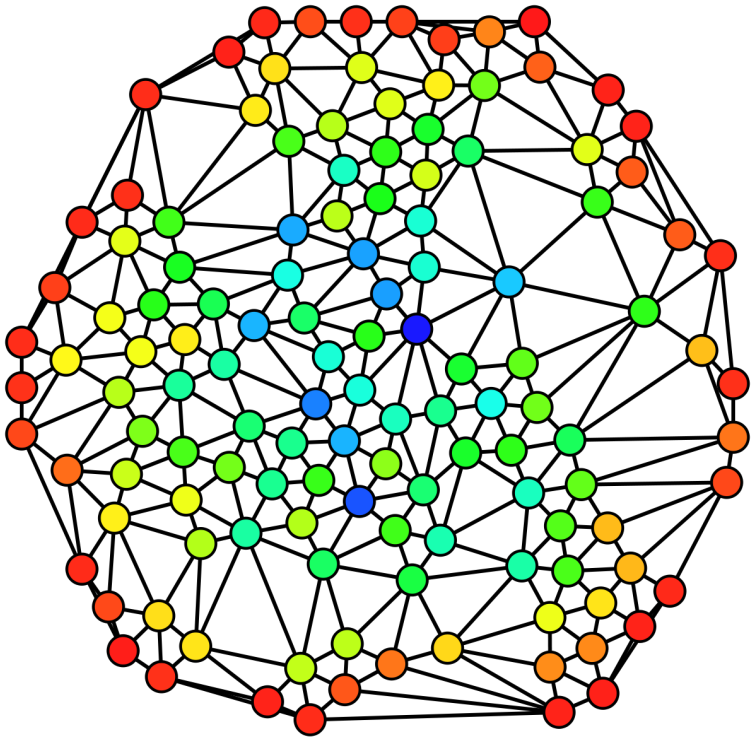

Figure 2.5. *A graph coloured based on the betweenness centrality of the nodes [2]*

Edge-betweenness centrality

Edge-betweenness centrality is a metric similar to node betweenness centrality, but referring to edges instead of nodes. It is defined as the fraction of shortest paths between all pairs of nodes that pass through a given edge. Like node betweenness centrality, many paths go through edges with a high value of edge-betweenness centrality, so those edges

control the flow of information throughout the network. This metric is used in Girvan-Newman's community detection algorithm [14], further discussed in section 2.4.2.

2.2.5 Complex network characteristics

Most networks that appear in real-life situations are complex networks and have various characteristics. Some of the most important features of real networks are listed in this section.

- **Dynamic topology:** Real networks usually don't have a static number of nodes and edges. New nodes connect to the network and existing links may change with time. This property makes analysis of real networks especially challenging.
- **Small-world effect:** Networks where the average shortest path length is small and the clustering coefficient is relatively high exhibit small-world behavior. In small-world networks most pairs of nodes can be connected by a small number of hops. This means that information travels relatively quickly even in large networks. In a study by Leskovec and Horvez in 2008 [40], a network of 180 million nodes was created from Microsoft Messenger instant-messaging conversations between users. Despite the large size of the network it was found that the average path length was only 6.6 hops.
- **Preferential growth:** In evolving real networks, new nodes tend to connect to existing nodes that have a large number of existing links. Nodes with a large number of connections are called *hubs* and they have a higher probability of connecting to new nodes than nodes with only a few connections.
- **Scale-free distribution:** The result of preferential attachment in real networks is that their degree distribution tends be exponential. This means that there are many nodes with only a few neighbours and a few nodes, the hubs, with a large number of links. More specifically, the fraction $P(k)$ of nodes with k neighbours follows $P(k)\ k^{-\gamma}$, where γ is a parameter usually valued between 2 and 3.

2.2.6 Synthetic network models

Many models exist that attempt to simulate the behaviour of real networks. In this section some popular models for synthetically creating networks are presented.

Regular graphs

Regular graphs are graphs where all nodes have the same degree. In k-regular graphs all nodes have degree k.

Lattice graphs are a similar model where the graph is organised like a grid. Not all nodes have the same degree but there are only a few possible degrees and there is an obvious pattern for the graph's creation.

Examples of regular-like networks are crystallic molecular structures or mobile cellular networks. Figure 2.6 shows a 2-regular graph and a lattice graph.

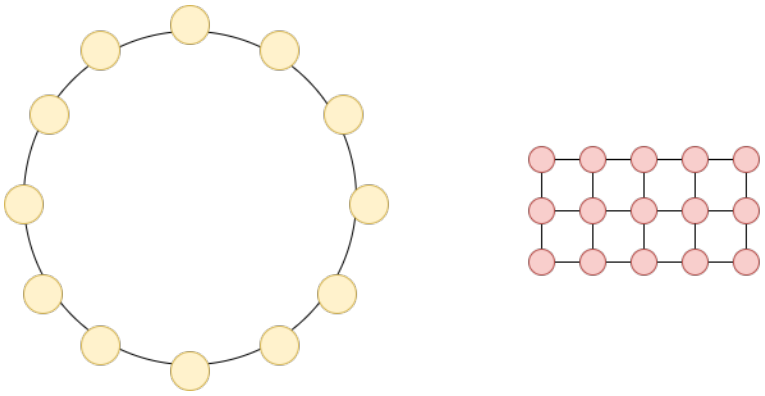

Figure 2.6. *Left: a 2-regular graph*
Right: a lattice graph

Erdős-Rényi *G(n,M)* model

A model was proposed by Erdős and Rényi [28] in 1959 to create random graphs with n nodes and M edges. The model chooses with equal probability a graph out of all possible graphs with n nodes and M edges.

Random graphs are mostly used for comparison purposes with other models, but they can also be used to model neural networks. An example of the normalised degree centralitiy distribution of an an Erdős-Rényi network is shown in figure 2.7.

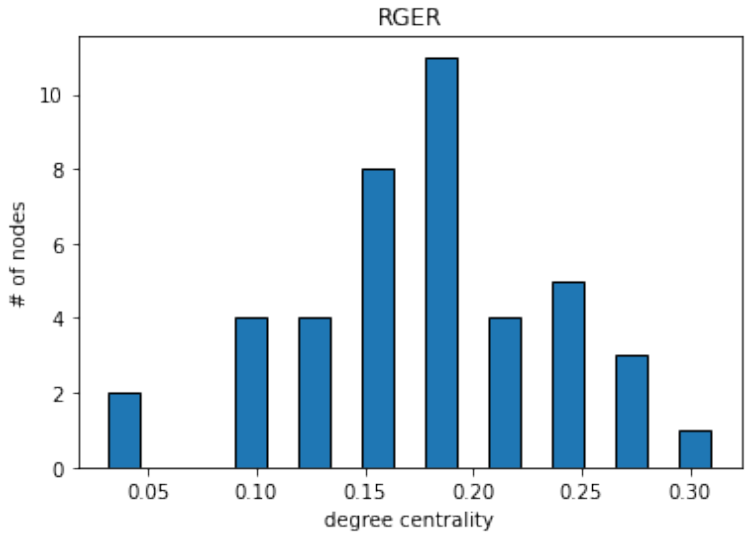

Figure 2.7. *The normalised degree centrality distribution of an Erdős-Rényi network with n=42 nodes and M=150 edges. The distribution follows a Gaussian curve.*

Gilbert's *G(n,p)* model

The $G(n, p)$ model for random graphs with n nodes was also proposed in 1959 by Edgar Gilbert [27]. A Bernoulli trial with probability p is performed independently for each possible edge, and the edge is added to the graph in case of success. In other words, for every pair of nodes the edge connecting them is added with probability p independently

of other edges.

The number of edges in a $G(n, p)$ graph follows a binomial probability distribution $B\left(\binom{n}{2}, p\right)$. The expected number of edges is $\binom{n}{2}p$. If $p = 1$ then the resulting graph is a *complete* graph with all possible $\binom{n}{2} = \frac{n(n-1)}{2}$ edges.

For large enough n the $G(n, p)$ model approximates a $G(n, M)$ model with $M = \binom{n}{2}p$, because by the law of large numbers the number of edges will tend to the expected value. This happens when $pn^2 \to \infty$ as $n \to \infty$. Therefore in this case the two models can be used interchangeably.

Random Geometric Graphs

Random Geometric Graphs $G(n, r)$ are graphs with n nodes that have randomly assigned coordinates in a metric space. Two nodes are connected by an edge if their distance in the metric space is smaller than a radius r. An example is shown in figure 2.8.

This model creates graphs with a relatively high average clustering coefficient, because nodes that are close in the metric space are connected, forming well-connected communities [41].

RGGs are often used to model road networks and other geographical data, as well as in statistical data analysis, due to the nature of their construction.

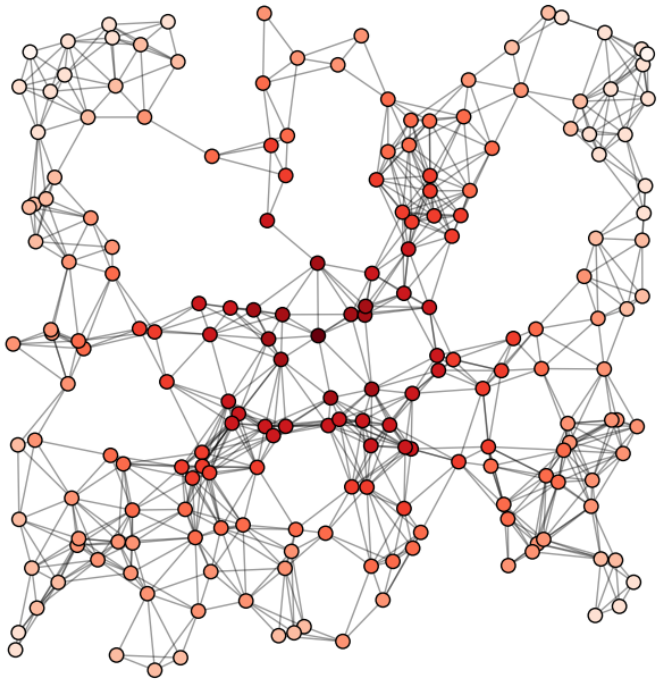

Figure 2.8. *Example of a random geometric graph [3]*

Watts-Strogatz model

The Wattz-Strogatz model proposed in 1998 [42] creates graphs with the small-world property that was analysed in 2.2.5. The model initially creates a k-regular graph with n nodes. Afterwards, each edge (u, v) from the regular graph is randomly rewired to (u, w) with probability p, where $w \in V, w \neq u \neq v$ a node chosen uniformly at random among V. This model results in graphs with a small average path length and high clustering coefficient. If $p = 0$ then the result is a k-regular graph, while if $p = 1$ the result is a random graph. As p changes from 0 to 1 the network becomes more random, as seen in figure 2.9.

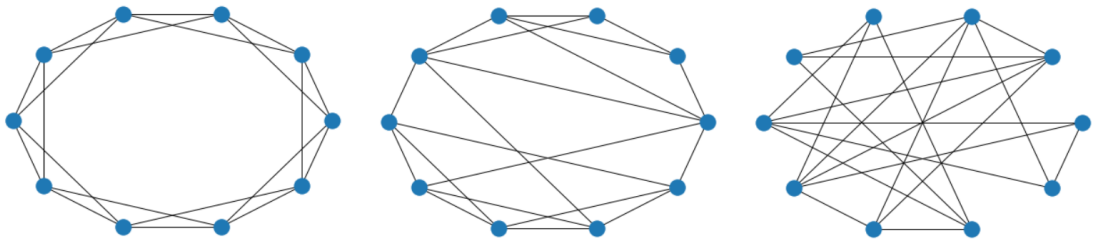

Figure 2.9. *Example of a Wattz-Strogatz network with $n = 10$ and $k = 4$. On the left with $p = 0$ the result is a regular graph. In the middle with $p = 0.3$ a small-world network. On the right $p = 1$ so the graph is random.*

Barabasi-Albert model

Barabasi and Albert proposed a model in 1999 [43] simulating preferential network growth in order to create scale-free networks. The model initially has m_0 nodes randomly connected with each other and continues by adding new nodes one at a time. Each new node is connected to $m \leq m_0$ already existing nodes, with a higher probability of connecting to nodes with high degrees. More specifically, the probability of a new node connecting to node u is $p(u) = \frac{|N_u|}{\sum_{v \in V} |N_v|}$ where $|N_v|$ the degree of node v. An example of the normalised degree centrality of a Barabasi-Albert network is shown in figure 2.10.

2.3 Machine Learning

Machine learning is a computer science field that studies methods and algorithms that allow machines to learn from data, called training data, and apply their knowledge to solve problems such as classification or data analysis.

2.3.1 Supervised learning

Supervised learning is a sub-field of machine learning where the training data are labelled, which means each data point consists of a feature vector and a label. The goal is to learn a function that will enable the correct classification of new data points given their feature vector.

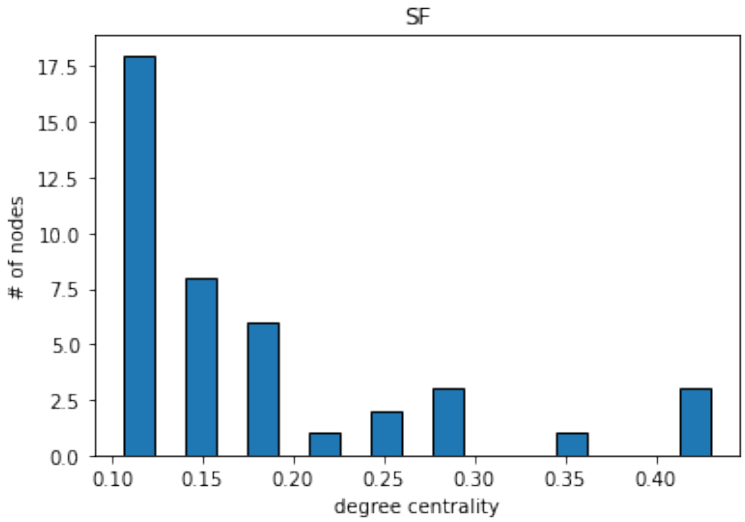

Figure 2.10. *The normalised degree centrality distribution of a Barabasi-Albert network with n=42 nodes and m=4. Unlike the random graph in figure 2.7, the distribution is exponential.*

Supervised learning is mostly used for classification problems, where the model trains on data that are labelled as different classes and then is tasked to assign labels to new data. Examples of applications include handwriting recognition [44], spam email filtering [45] and satellite image classification [46].

2.3.2 Unsupervised learning

Unsupervised learning is a sub-field of machine learning where the training data are not labelled. It includes methods for data analysis and exploration as well as generating new data.

The most common problems in unsupervised learning are clustering and anomaly detection. Clustering is used to group data into smaller groups, where data points inside the same group have similar properties. It can be applied to a variety of fields such as medicine [47], political and social science [48] and data mining [49]. Anomaly detection refers to the problem of identifying abnormal data and is expanded upon further in 2.3.5.

2.3.3 Deep learning

Deep learning refers to machine learning models that attempt to simulate neural networks like the human brain. Multi-layered architectures are used to create artificial neural networks which are very powerful but require a lot of training data and computational resources [50].

Deep learning is very successful in computer vision [51], natural language processing [52] and speech recognition [53].

2.3.4 One-class classification

One-class classification is a machine learning problem that attempts to describe a single class of objects and distinguish them from other objects. The training data consists only of objects from the class, as opposed to supervised classification problems where there exist labelled data from all classes during training.

The goal is to identify whether new data belongs to the learned class or not, thus there are applications in novelty detection, anomaly and outlier detection. Novelty detection refers to identifying new classes of objects that may start appearing, whereas anomaly and outlier detection attempt to identify exceptional data points. One-class classification is also used in scenarios with highly imbalanced training data [54], as in these cases traditional classification methods tend to have heavy bias towards the majority class.

One-class classification methods can be split into three large categories:

- **Density estimation methods:** These methods assume the data follows a probability distribution, for example Gaussian, and try to find the distribution parameters that best fit the data. New points are evaluated based on a threshold.
- **Boundary methods:** These methods attempt to set a boundary around the training data based on a few target points, for example the K-centers algorithm [55].
- **Reconstruction methods:** Reconstruction methods create a generating model that fits the data, for example self-organising maps [56].

2.3.5 Anomaly detection

Anomaly detection refers to the problem of identifying abnormal events and data points that deviate significantly from the majority of the data. These data points may make calculations unnecessarily difficult due to being outliers. For example, in supervised learning, detecting and removing exceptional training data often improves the results of the model [57]. Similarly, prediction models for time-series also benefit from the removal of outlier data [58].

Anomalous data points may also have different origins from the rest of the data, in which case the goal is to identify these events. Anomaly detection is commonly used in cybersecurity [59], [60] in order to detect network attacks. Other applications include medicine [61], financial fraud detection [62] and industrial quality assurance [63].

Some categories of anomaly detection methods are the following:

- **Statistical methods:** These methods fit the data into a probability distribution and check which points deviate most from it.
- **Density methods:** The main idea behind these methods is that the majority of the data is similar and creates a dense population, so the outlier points are in sparser regions. An example technique is isolation forests [64].
- **Clustering methods:** These techniques group the data into similar clusters and outliers are detected based on their distance to the clusters. An example clustering algorithm is K-means, explained in 2.3.6.

- **Deep learning:** Neural networks can be used to learn patterns from data and detect new points that do not follow these patterns. Long short-term memory neural networks (LSTMs) are often used for this purpose [65].
- **Bayesian networks:** These methods calculate the probability that an event occurs based on previous events. Events with low probability are deemed anomalies.

2.3.6 K-means clustering

K-means clustering is an unsupervised learning algorithm that splits data points into k clusters [66]. It is an optimisation problem with the goal of finding an assignment of points to clusters such that the sum of the distances of the data points to the mean of their cluster is minimised. The cluster means can then be used for classification, as new points can be assigned to their closest cluster.

The problem is NP-hard [67], but there are many algorithms that find potentially sub-optimal solutions. We will examine the most popular one, which after an initialisation phase, iterates over 2 steps until the cluster means converge. During initialisation, coordinates are assigned to the means of the k clusters. There are many methods for initialisation [68], but the most common is to randomly pick k data points and assign their coordinates as cluster means.

The algorithm then repeats the following two steps until the cluster means no longer change, or change less than a given tolerance. It converges to a local optimum as long as euclidean distance is used.

- Calculate the distance of each data point to every cluster mean and assign it to the nearest one.
- Compute a new mean for each cluster by averaging the coordinates of all data points belonging to it.

2.4 Community Detection

Complex networks have a tendency to form *communities*, which are sets of nodes more densely connected among each other than the rest of the network [10]. Knowledge of these communities may allow a better understanding of the structure of the network and spreading of information inside it, so their detection is a problem with an abundance of applications. For example, community-based recommendation systems take advantage of the fact that users in the same community are more likely to have similar interests [11]. Communities can also be used to examine the spread of epidemics such as COVID-19 [12], [13].

Detecting communities in networks is a difficult problem, because the number, size and density of communities is unknown and may vary. For this reason there exist multiple algorithms that tackle it, each with different advantages and disadvantages. They are usually evaluated based on accuracy, execution time and/or computational complexity. Furthermore, depending on the problem, communities may overlap or not, meaning that

in certain problems a node may belong to multiple communities, whereas in others each node belongs to a single community.

An extensive survey was published in 2021 by Moscato and Sperli [69] comparing many different community detection algorithms based on their complexity and the type of problem they solve, i.e., whether the communities are strong and/or overlapping. A community is called strong if every node that belongs to it has more within-community neighbours than outside-community neighbours, whereas it is called weak if there are more total edges inside it than edges connecting a node from the community to a node outside it.

In the survey, community detection techniques are classified into five broad categories depending on if they use topological features of the networks, game theory optimizations, artificial intelligence (AI), fuzzy or greedy methods. The AI category is further divided into two subcategories: network representation learning and deep learning, which will be discussed in section 2.4.7.

2.4.1 Hierarchical communities

Many community detection algorithms follow a hierarchical approach, which means that communities may have smaller communities inside them. Depending on when the algorithms stop, the resulting communities are larger or smaller. There are two categories of hierarchical methods, depicted in figure 2.11:

- **Agglomerative** methods begin by having each node be its own community, and continue by merging communities with each other until the whole network is a single community.

- **Divisive** methods are the opposite. At the beginning the whole network is a single community and it is constantly divided into smaller communities until each node is its own community.

We will now present the community detection methods used in this diploma thesis.

2.4.2 Girvan-Newman algorithm

The Girvan-Newman algorithm [14] is a divisive hierarchical community detection algorithm which performs an iterative removal of edges that disconnect the network. The number of communities to detect has to be specified beforehand, then the algorithm removes edges from the network until the correct number of connected components remain, each of which is a community.

The algorithm decides which edge to remove based on the edge-betweenness centrality metric. Edges with a low edge-betweenness centrality are more likely to be within-community edges, because they can be bypassed by other edges in the community. On the other hand, edges with a high edge-betweenness centrality are more likely to connect two different communities, since all shortest paths between pairs of nodes, one from each

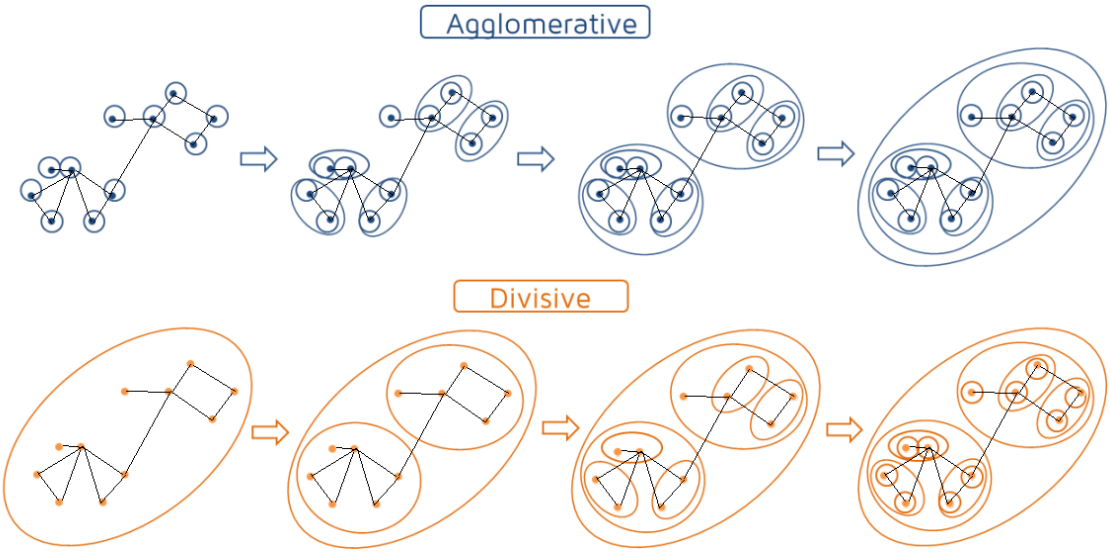

Figure 2.11. *Hierarchical communities, edited from [4]*

community, have to pass through that edge. For this reason each iteration of the algorithm removes the edge with the highest edge-betweenness centrality. Figure 2.12 shows a network partitioned into five communities via Girvan-Newman's algorithm.

Girvan-Newman algorithm can be summarised as follows, given a specified number of communities k.

1. Calculate the edge-betweenness centrality of all edges in the network.
2. Remove the edge with the highest edge-betweenness centrality.
3. If the number of connected components in the network is k, stop. Otherwise repeat steps 1 and 2.

The main problem with this algorithm is its $O(nm^2)$ time complexity, due to the constant calculations of edge-betweenness centrality. This makes it unviable for modern large-scale network applications. An alteration that speeds it up is to remove multiple edges during step 2 instead of just one. This may result in less accurate communities but saves a significant amount of time, because the edge-betweenness centralities are computed fewer times overall.

2.4.3 Louvain method

The Louvain method for community detection [15] is a greedy method that attempts to optimise the modularity score of the resulting communities. Modularity is a measure used to evaluate community detection methods, explained further in section 2.4.9. Two advantages of this algorithm relative to Girvan-Newman is that the number of communities is not specified at the start, and the time complexity is only $O(n\,log(n))$.

The Louvain algorithm is an agglomerative hierarchical method. It consists of three steps that are repeated until modularity no longer increases:

1. Assign each node of the network to its own community.

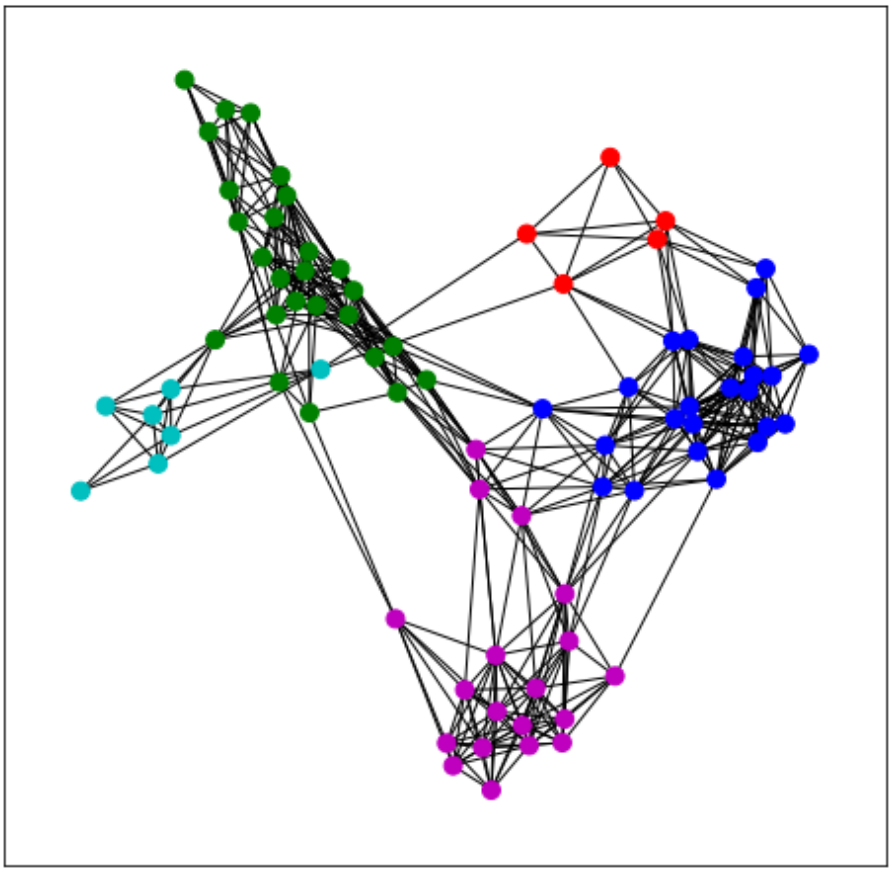

Figure 2.12. *An example of Girvan-Newman's algorithm applied to a network to separate it into five communities.*

2. For each node, calculate the modularity score after assigning it to the communities of each of its neighbours. Assign the node to the community which results in the largest modularity increase than if it remained in its own community. If no reassignments increase the modularity, leave the node in its own community.

3. Create a new network, where each previous community corresponds to a single node. Edges within a community are turned into self-loops. Multiple edges from one community to another are turned into a single edge between two nodes, with a weight equal to the number of edges that connected the communities. Repeat from step 1 with the new network.

The Louvain method is similar to the Clauset-Newman-Moore greedy method [70]. The Clauset-Newman-Moore method also begins with each node in its own community, and repeatedly joins pairs of communities into one in order to maximise modularity.

2.4.4 Walktrap

Walktrap [16] is a community detection algorithm that uses random walks. It is an agglomerative hierarchical method based on the idea that short random walks in a network tend to stay in the same community, as most edges in a community are within

itself. These random walks are easy to compute and are used to create a custom distance metric, based on which the communities are joined. The complexity of the algorithm is $O(mn^2)$ in the general case and $O(n^2 log(n))$ for sparse graphs.

2.4.5 Spectral clustering

Spectral clustering is a clustering algorithm that uses the eigenvectors of the Laplacian matrix of a network [17]. The Laplacian matrix of a network is defined as follows:

$$L = D - A \tag{2.10}$$

where A the adjacency matrix of the network and D a diagonal $n \times n$ matrix where D_{ii} is equal to the degree of node i. After the Laplacian matrix of the network has been computed, we also calculate its eigenvalues and eigenvectors. For detecting k communities, the k eigenvectors corresponding to the k largest (or smallest) eigenvalues are selected. K-means clustering is then applied (see section 2.3.6) on the vectors created by grouping each row of all eigenvectors. The results of K-means clustering are the communities.

2.4.6 Hyperbolic Girvan-Newman

Hyperbolic Girvan-Newman (HGN) [5] is a variation of the classic Girvan-Newman algorithm that utilises a mapping of the network into hyperbolic space in order to more efficiently calculate the edge-betweenness centralities. Before the algorithm is explained, we will make a short introduction to hyperbolic embedding.

Hyperbolic Embedding

Mapping networks into low dimensional geometric spaces while preserving their properties is a difficult challenge with a lot of benefits such as better visualisation or quicker calculations of distances.

This work by Serrano and Boguñá [71] presents hyperbolic geometry as an ideal metric space for mappings of complex networks, since it preserves important qualities such as the small-world property, scale-free degree distribution. Two equivalent models are discussed, the first of which gives each node an angular position in a circle and a hidden degree. The second model gives each node coordinates inside a disk, where nodes closer to the center are more popular. The probability of two nodes in the real network being connected is relative to their proximity in the hyperbolic space (similarity) and their popularity. The hyperbolic embeddings also preserve the initial network's communities, as well as mimic the phenomenon of new nodes tending to appear in popular regions.

An algorithm to create popularity-similarity models in hyperbolic 2-dimensional space is hypermap [72]. An example of its application is shown in figure 2.13.

Another algorithm for hyperbolic embedding of networks is Rigel embedding [18]. In the hyperbolic geometry used by this algorithm, the distance $d_H(x, y)$ between two points x, y is given by:

$$cosh d_H(x, y) = \sqrt{(1 + ||x||^2)(1 + ||y||^2) - < x, y >} \tag{2.11}$$

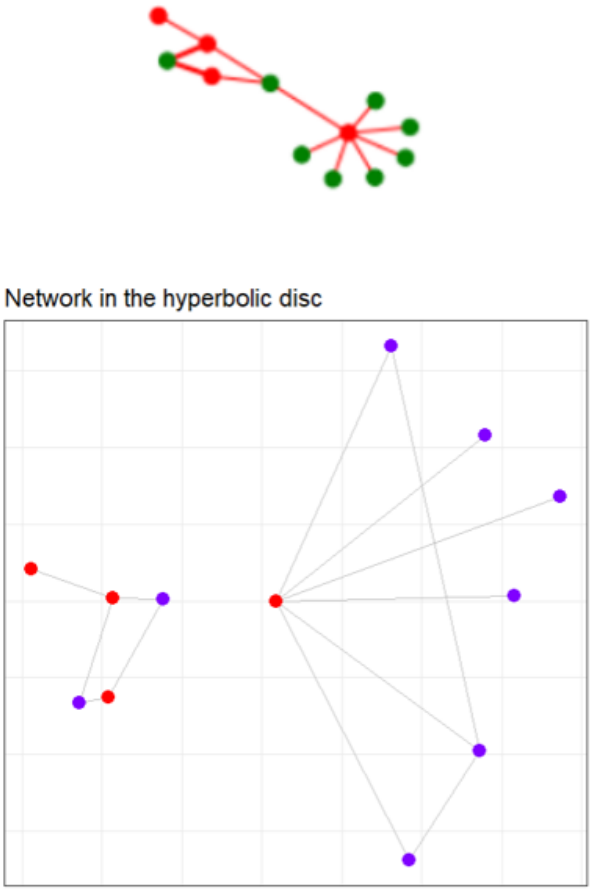

Figure 2.13. *An example of the Hypermap algorithm used to embed the above network into hyperbolic space. Hypermap follows the disk model, which means nodes closer to the center are more popular in the real network (high degree) and nodes near each other in the disk are more likely to be connected in the real network.*

where $||x||, ||y||$ the euclidean norms and $< x, y >$ the inner product of x, y. Rigel algorithm selects a subset of $l << n$ nodes from the network, called landmarks. The first step of the algorithm assigns hyperbolic coordinates to the landmarks by solving a global optimisation problem, in order for the hyperbolic distances of each pair of landmarks to be as close as possible to the hop distances of the nodes in the original network. In the next step of the algorithm the coordinates of the rest of the nodes are computed based on the coordinates of the landmarks, so that the hyperbolic distances of each node to all the landmarks are as close as possible to their hop distances before embedding.

Hyperbolic Edge-Betweenness Centrality

Girvan-Newman algorithm for community detection does not scale well for large networks, as discussed in 2.4.2, because the constant computation of edge-betweenness centralities is costly. However, by embedding the network into hyperbolic space an approximation to edge-betweenness centrality can be calculated much more efficiently [19]. The approximation measure is called Hyperbolic Edge-Betweenness Centrality (HEBC) and utilises the coordinates of the nodes in the hyperbolic space produced via Rigel em-

bedding.

Instead of shortest paths between nodes, greedy paths are used in the hyperbolic space. A greedy path is a path with a source node and destination node. It begins from the source node and chooses each next node as the one closest to the destination. Greedy paths in hyperbolic space tend to be of similar length as shortest paths in the real network, but the number of greedy paths between two nodes is not always the same as the number of shortest paths between them in the real network. For this reason the HEBC may differ from the EBC which results in a loss of accuracy. However, calculating greedy paths is much quicker than shortest paths, so HEBC is more computationally efficient than EBC. The algorithm for computing HEBC is shown in figure 2.14.

Algorithm : Hyperbolic Edge Betweenness Centrality (HEBC).

$HEBC(u,v) = 0$; $\forall\, u,\ v \in V, V$ the node set
for each node $s \in V$ **do**
  % Part I: Sort all nodes in order of decreasing hyperbolic distance towards the destination s, $v_N = s$
  Obtain S as $S = \{v_1 \preceq v_2 \preceq ... \preceq v_N\}$, % " $\preceq$ " indicates the ordering of decreasing hyperbolic distance from s, $S_1 = S$
  % Part II:
  % $\sigma_s(u)$: the number of greedy paths beginning at node u and finishing at node s
  $\sigma_s(u) = 0\ \forall u \in V,\ \sigma_s(s) = 1$
  % $N_G(i,s)$: the greedy neighbors of i in S
  for $i = N$:1 **do**
    for each $u_j :\ u_i \in N_G(i,s)$ **do**
      $\sigma_s(u_j) = \sigma_s(u_j) + \sigma_s(u_i)$;
      remove u_i from S_1
  % Part III: sum dependencies (δ), and compute HEBC values for all edges
  $\delta(u) = 0,\ \forall\, u \in V$
  for i = 1:$N-1$ **do**
    for each $u_j \in N_G(i,s)$ **do**
      $c = \frac{\sigma_s(u_j)}{\sigma_s(u_1)} * (1 + \delta(u_i))$;
      $HEBC(u_i,u_j) = HEBC(u_i,u_j) + c$;
      $HEBC(u_j,u_i) = HEBC(u_j,u_i) + c$;
      $\delta(u_j) = \delta(u_i) + c$;
      remove u_i from S_1

Figure 2.14. *HEBC Algorithm [5]*

Hyperbolic Girvan-Newman algorithm

The HGN algorithm, similarly to the Girvan-Newman algorithm, requires the number of communities k to be given as input. There is an extra parameter b, the batch size, which signifies the number of edges removed each iteration. It is described in [5] and can be summarised as follows:

1. Embed the largest connected component into hyperbolic space.

2. Calculate the HEBC of all edges and remove the b edges with the largest values.

3. If the number of connected components is equal to or higher than k, stop. Else repeat steps 1 and 2.

Even though there is some initial overhead for the network embedding, this is far less costly than the time saved by calculating HEBC instead of EBC. For this reason the HGN algorithm is much faster than the original Girvan-Newman. There is some loss of accuracy due to the inaccuracies caused by HEBC, but the authors of [5] showed that the HGN algorithm produces communities of similar modularity as Girvan-Newman. Thus it is a viable alternative for larger networks where the classic algorithm struggles.

In [73] the HGN algorithm was extended to work with graph databases. A graph database is an alternative graph representation to the common adjacency matrix, which is used for very large networks. The modified HGN algorithm stores all the necessary information in the database, such as hyperbolic coordinates and neighbours of nodes, and extracts any information it needs through SQL queries.

2.4.7 Deep learning methods

There have been many successful attempts to utilise deep learning models to solve community detection problems, some of which are outlined in a 2021 survey by Su et al. [74]. The survey divides the algorithms into six categories. The classification is based on the deep learning models used: convolutional neural networks, attention mechanisms, generative-adversarial training, autoencoders, nonnegative matrix factorization, and sparse filtering.

2.4.8 LFR Benchmark

The Lancichinetti–Fortunato–Radicchi (LFR) benchmark [75] is a synthetic network generation model that creates networks with known communities. It is used to compare different community detection algorithms.

The model creates scale-free networks where node degrees follow a power-law distribution with exponent γ. The minimum, average and maximum degree are set. The community sizes also follow a power law distribution with exponent β. The model also has a parameter μ, called the *mixing parameter*, which inversely controls the fraction of neighbours of a node that belong to the same community as the node itself. The value of μ is between 0 and 1, where at $\mu = 0$ all links from a node are with nodes of the same community, and at $\mu = 1$ they are with nodes of other communities.

A 2016 analysis by Yang et al. [76] compares eight different algorithms using the LFR benchmark. The algorithms are compared in terms of execution time, empirical mixing parameter, and accuracy, which is calculated by the NMI with the ground truth generated for the LFR networks. In the end, the authors note the best algorithms for certain regions of values of mixing parameter and network size. The proposed strategy to find the optimal algorithm for an application is to first run a good algorithm to estimate the mixing parameter, if unknown, and then pick the best method in the corresponding region.

2.4.9 Evaluation Metrics

In this section we will present two common metrics used to evaluate communities produced by community detection algorithms.

Normalised Mutual Information

Accuracy in community detection is often measured using the *normalised mutual information* (NMI) of the results of the algorithm and the ground truth communities. The mutual information between two discrete random variables X, Y is defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_{XY}(x,y) log\left(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\right) \tag{2.12}$$

The normalised mutual information is then defined as:

$$NMI(X,Y) = \frac{2I(X;Y)}{H(X) + H(Y)} \tag{2.13}$$

where $H(X), H(Y)$ the entropy of X, Y. In the case of community detection the random variables X, Y denote the probability of a node belonging to a specific community in the ground truth or the algorithm result. The distribution can be calculated empirically by dividing the number of nodes in each community to the total number of nodes in the network.

Modularity

Another measure used to evaluate the results of community detection methods is *modularity*, which compares the number of within-community edges to the expected number if the edges were randomly distributed. It is often used when there is no ground truth available for the communities, which is required in order to calculate the NMI. Modularity takes values between $-\frac{1}{2}$ and 1, where positive values mean that there exist more edges within communities than if the edges were random. It is defined as follows:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{k_i k_j}{2m}\right) \delta(c_i, c_j) \tag{2.14}$$

where m is the number of edges in the network, A the adjacency matrix, k_i, k_j the degree of nodes i, j, c_i, c_j the communities of nodes i, j and $delta(x, y)$ the Kronecker delta function:

$$\delta(x,y) = \begin{cases} 1 & \text{for } x = y \\ 0 & \text{for } x \neq y \end{cases} \tag{2.15}$$

2.5 Botnets

Botnets are groups of devices connected to the Internet, such as computers, smartphones or Internet of Things (IoT) devices whose security has been breached and are

controlled by a malicious actor. Each device is turned into a bot after it is infected by some malicious software (malware) and can be then controlled remotely. Botnets are used to perform a variety of attacks such as spamming, Distributed Denial-of-Service (DDoS) attacks (see section 2.5.1), data theft (passwords, credit card numbers, corporate information) or click fraud to generate false traffic [20].

Traditionally botnets operated via a client-server model. The bots (clients) would listen to a server waiting for incoming commands. The attacker would send instructions to the bots through the server, then bots would execute the command and send the results back through the server. Many client-server botnets used Internet Relay Chat (IRC), a text-based low-bandwidth instant messaging system [22]. The bots would connect to an infected IRC server appearing as normal users and joined a channel where the attacker sent their commands. The disadvantage of IRC-based botnets is that anti-malware software can shut down the server or channel where the botnet coordinated, leaving the infected clients dormant since they no longer receive instructions. To avoid being shut down, IRC-based botnets need to constantly switch channels and servers. A famous IRC bot that is still being maintained is Eggdrop [77].

More recent botnets operate via peer-to-peer (P2P) models, to avoid the vulnerability of centralised botnets. P2P bots can both send and receive commands. They randomly probe other IP addresses until they recognise another infected device, and then exchange information about other known bots and latest instructions. This way the botnet spreads and updates itself. To ensure the authenticity of the commands, public key encryption may be used so that a private key is necessary to control the botnet [78]. A famous P2P botnet is Gameover ZeuS [23] which was used for bank fraud.

2.5.1 Distributed Denial-of-Service attacks

A denial-of-service (DoS) attack is a type of cyber attack that aims to render a device or service unavailable by overwhelming it with requests until it is unable to process normal requests. A DDoS attack is a type of DoS attack that utilises many different sources to launch DoS attacks, making it harder to defend against. While simple DoS attacks can be mitigated by blocking traffic from the attacker, it is much more difficult to separate normal from malicious traffic during distributed attacks. The compromised devices may also hide their IP address by using a random one (spoofing) to make it near impossible to pinpoint the source of the attack [79].

Botnets are often used to conduct DDoS attacks, as the attacker can command the bots to spam the target with requests. A famous case was the Mirai botnet attacks in 2016 [21] that infected over half a million machines in the US east coast. Mirai is a malware that targets consumer IoT devices turning them into bots that can be controlled remotely, which are then used to launch DDoS attacks.

2.5.2 Botnet detection

Due to the high frequency of DDoS attacks [24] many methods have been developed attempting to detect and stop botnets. Botnet detection methods can be broadly split into

host-based and *network-based* techniques [25]. Host based techniques rely on monitoring a single machine, checking for suspicious files, high processing overhead, etc. Network based techniques monitor the traffic of a whole network either by passively waiting for possible bot-commander communications or by sending test packets into the network and observing the reaction.

We will now summarise a number of different botnet detection approaches:

- **Anomaly-based:** A very popular approach to botnet discovery is to utilise anomaly detection techniques to detect unusual traffic [80], [81]. Network anomalies may be unexpectedly high traffic, increased traffic on mostly unused ports, or higher than usual latency. They may also be loosely defined deviations, after modelling normal behavior [82].

- **Signature-based:** Signature-based methods make use of the signatures of known botnets in order to detect them [83], [84]. They are incredibly quick and accurate for detecting well-known botnets but cannot detect unknown botnets.

- **DNS-based:** DNS-based methods rely on monitoring Domain Name System (DNS) requests on a network, for example by analysing the frequency of failed DNS resolution requests since bots often used unregistered domains [85].

- **Honeypots:** A honeypot is an intentional vulnerability within a network aimed to attract botnet attacks in order to obtain informtion about their behavior [86], [87].

- **Deep learning:** Recent advances in botnet detection also utilise deep learning techniques such as graph convolutional networks [88], deep graph autoencoders [89], transformers [90], etc.

Chapter 3

Methodology

The method we use for botnet detection largely follows the method described in [26], but with different community detection algorithms in the last stage. The method consists of two broad parts. The first part analyses network traffic attempting to identify time periods with anomalous traffic, utilising anomaly detection methods. The second part detects bots in the selected anomalous time periods. In this section we will describe the method in detail.

3.1 Anomalous window detection

The first part of the botnet detection method is to use anomaly detection techniques to identify time periods with unusual network traffic. We use statistical deviation based anomaly detection. Initially, a training period without bots is used to generate a reference distribution of normal traffic. Afterwards, the evaluation period is split into smaller windows, and traffic from each window is compared to the normal traffic. Large deviations from the normal behaviour are deemed anomalous.

We use two different approaches to this problem depending on if the input traffic is in netflow or pcap format.

3.1.1 Netflow files

In the netflow format each flow represents a communication session between two hosts. It includes a lot of relevant information such as the duration of the communication, the total number of packets and/or bytes transmitted, what protocol was used, etc.

The flows may be unidirectional or bidirectional. If there is traffic between two hosts A and B, who both send packets to each other, it can be represented as two separate unidirectional flows, one from A→B and one from B→A, or it can be represented as a single bidirectional flow.

The general idea of anomaly detection using netflow files is to select a number of features from the flows, quantize them, and treat the resulting vector as a value of a discrete random varable. An empirical probability distribution is then created using all vectors inside a time window. This distribution created from the training period is the reference used to evaluate future time windows for possible anomalous behavior. The comparison of a probability distribution from an evaluation window with the reference distribution is

done using the Kullback-Leibler (KL) divergence of the discrete distributions:

$$D(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{3.1}$$

where p the evaluation distribution, q the reference distribution and $\mathcal{X}$ the common sample space of the two distributions. It should be noted that if the evaluation window contains a value x_i that was not encountered in the training period, $q(x_i) = 0$ resulting in the KL divergence being undefined.

After calculating the KL divergence of the two distributions, the resulting value is compared to a threshold. If it is larger than the threshold then the time window is anomalous.

3.1.2 Pcap files

Pcap files contain information about individual packets transmitted through a network. The main idea behind anomaly detection on pcap files is to create interaction graphs and measure their degree distribution. A graph model is chosen based on the degree distribution obtained during the training period. The evaluation period distributions are compared to the reference using a different function depending on the model used, and if the divergence is higher than a threshold then the window is considered anomalous.

The interaction graph for a given time window is a graph whose nodes are the set of all hosts that sent or received a packet during that time window. If two nodes interacted with each other inside the time window then they are connected by an edge in the interaction graph. In order to create a larger distribution of node degrees during the training period, it is split into smaller time windows. Some of these time windows are selected, possibly at random, and some nodes are chosen, also possibly randomly, to sample their degree on the interaction graph of those windows. This way we can get a set of M node degrees $\mathcal{D} = \{d_1, d_2, \ldots d_M\}$.

Selection of a graph model

After obtaining $\mathcal{D}$, we calculate whether it is more likely to be the distribution of a random graph or a preferential attachment graph. For the random graph we use a Poisson approximation of the Gilbert $G(n, p)$ model.

On a $G(n, p)$ random graph, the degree of any node follows a binomial distribution. As $n \to \infty$ and if np is constant, the binomial distribution converges to a Poisson distribution with parameter $\lambda = np$. For a $G(n, \lambda/n)$ graph the log-likelihood of degree distribution $\mathcal{D}$ is:

$$L_{Rand}(\mathcal{D}; \lambda) = \log(\lambda) \sum_{i=1}^{M} d_i - \lambda M + C \tag{3.2}$$

where $C = -\sum_{i=1}^{M} \log(d_i!)$ and the parameter λ can be calculated by maximum likelihood

estimation:

$$\hat{\eta} = \frac{1}{M}\sum_{i=1}^{M} d_i \tag{3.3}$$

For the preferential attachment case, two different models are used: offset Barabási-Albert [29] and Chung-Handjani-Jungreis (CHJ) [30]. Both models' asymptotic degree distributions are power-law like following $P_{PA}(k;\gamma) = \frac{k^{-\gamma}}{\zeta(\gamma)}$ where γ is a parameter and $\zeta(x) = \sum_{n=1}^{\infty} n^{-x}$ is Riemann's zeta function. The log-likelihood of $\mathcal{D}$ is:

$$L_{PA}(\mathcal{D};\gamma) = -\gamma\sum_{i=1}^{M} log(d_i) - M\, log(\zeta(\gamma)) \tag{3.4}$$

The parameter γ can also be calculated by maximum likelihood estimation:

$$\hat{\gamma} = \varphi^{-1}\left(-\frac{\sum_{d_i>0} log(d_i)}{M}\right) \tag{3.5}$$

where $\varphi(x) = \frac{\dot{\zeta}(x)}{\zeta(x)}$ and $\dot{\zeta}(x)$ the first derivative of $\zeta(x)$.

It should be noted that in 3.4 if a sample degree $d_i = 0$, possibly due to observation error, then the logarithm tends to $-\infty$. To deal with this possibility the authors of [26] propose ignoring those values during summation and subtracting a penalty of $\partial \times (number\ of\ 0\ values)$ at the end of 3.4.

Choosing a model between random and preferential attachment is done by comparing the log-likelihood of $\mathcal{D}$ being generated from those models.

Anomaly detection test

The purpose of the training period is to select a graph model and estimate its parameters. Afterwards, the evaluation period is also split into time windows and the interaction graph for each of them is created. If the random graph model was selected, the degree distribution q of each interaction graph is evaluated using the following function:

$$I_{Rand}(q;\hat{\eta}) = D(q \,\|\, p_{\hat{\eta}}) + \frac{1}{2}(\bar{q} - \hat{\eta}) + \frac{\bar{q}}{2} log(\hat{\eta}) - \frac{\bar{q}}{2} log(\bar{q}) \tag{3.6}$$

where D the KL divergence as defined in 3.1, $\hat{\eta}$ the MLE of the η parameter calculated during the training period by 3.3, $p_{\hat{\eta}}$ the reference distribution measured during the training period and $\bar{q}$ the expectation of the empirical distribution q of the time window. If the value of I_{Rand} is higher than a threshold then the time window is considered to be anomalous.

Otherwise, if the preferential attachment models are selected, the degree distribution q of each interaction graph is evaluated using two different functions, one for each PA model, and the minimum of the two is compared to a threshold. The function for the offset Barabási-Albert model is:

$$I_{BA}(q;\hat{a}) = \sum_{i\geq 0}(1 - [q]_i) log\left(\frac{1 - [q]_i}{(i + 1 + \hat{a})q_i/(2 + a)}\right) + \left(1 - \sum_{i\geq 0} i\, q_i\, log(2 + \hat{a})\right) \tag{3.7}$$

where $[q]_i = \sum_{j=0}^{i} q_i$, a a parameter of the offset Barabási-Albert model and $\hat{a} = \hat{\gamma} - 3$.

The function for the CHJ model is:

$$I_{CHJ}(q;\hat{p}) = (1-q_0)log\left(\frac{1-q_0}{\hat{p}+(1-\hat{p})q_0/2}\right)+\sum_{i=0}^{\infty}(1-[q]_i)log\left(\frac{1-[q]_i}{(1-\hat{p})(i+1)q_i/2}\right)+\left(1-\sum_{i\geq 0} i\,q_i\right)log\left(\frac{2}{1-\hat{p}}\right) \tag{3.8}$$

where p a parameter of the CHJ model and $\hat{p} = 1 - \frac{1}{1-\hat{\gamma}}$.

3.2 Botnet detection

Having detected a number of anomalous time windows, the next step in the process is to identify the bots in these anomalies. It is difficult to detect bots directly, so we initially detect highly influential nodes in the network, called *pivotal nodes*, which correspond to botmasters or targets. By measuring each node's interactions with the pivotal nodes and the correlations with other nodes, we then construct the *social correlation graph*, where bots are more likely to be connected to each other. The final step is to run a community detection algorithm on the social correlation graph to detect the bots.

Initially, we organise the anomalous windows detected previously by creating sets of *interaction records*. An interaction record (i, j, t) is a tuple of two nodes i, j and a timestamp t which denotes that nodes i and j interact at time t. If the data was in netflow format, each flow in the anomalous time windows can be directly converted to an interaction record. Otherwise, if the data was in pcap format, each edge in the anomalous window's interaction graph also represents an interaction record. This way the output of the anomaly detection can be written as a sequence $\mathcal{A} = \{S_1, ..., S_{|\mathcal{A}|}\}$, where $S_i = \{r_{i,1}, ..., r_{i,|S_i|}\}$ represents the i-th anomalous window containing interaction records $r_{i,1}, ..., r_{i,|S_i|}$.

3.2.1 Pivotal nodes

The first part of the botnet discovery approach is to identify *pivotal nodes*, which are the botnet's leading nodes and its primary targets. Botnet leaders (botmasters) have to control the rest of the bots, so they interact with them quite often, and in DDoS attacks bots constantly interact with their targets, so these pivotal nodes have more interactions than regular nodes.

If the total number of nodes in $\mathcal{A}$ is n, and the number of interactions between nodes i, j in anomaly S_k is G_k^{ij}, then we define the *total interaction measure* of node i as:

$$e_i = \frac{1}{|\mathcal{A}|}\sum_{k=1}^{|\mathcal{A}|}\sum_{j=1}^{n} G_k^{ij} \tag{3.9}$$

This measure is a way to quantify the interactions of each node with all other nodes in $\mathcal{A}$. Using it we can define *pivotal nodes* as the set of nodes $\mathcal{N}$ with a total interaction measure higher than a threshold τ:

$$\mathcal{N} = \{i : e_i > \tau\} \tag{3.10}$$

3.2.2 Social Correlation Graph

Having identified the pivotal nodes, we now check the interactions between them and regular nodes, because the interactions between bots and pivotal nodes are likely to be correlated.

For each node $i = 1, \ldots, n$ let X_i be the total number of interactions between i and pivotal nodes. For each anomaly S_k let $x_i^k = \sum_{j \in \mathcal{N}} G_k^{ij}$ be the number of interactions between node i and pivotal nodes in S_k. We calculate the mean $\bar{X}_i$ and standard deviation $\sigma(X_i)$ of X_i for all i:

$$\begin{aligned} \bar{X}_i &= \frac{1}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} x_i^k \\ \sigma(X_i) &= \sqrt{\frac{1}{(|\mathcal{A}| - 1)} \sum_{k=1}^{|\mathcal{A}|} (x_i^k - \bar{X}_i)^2} \end{aligned} \tag{3.11}$$

We then calculate the correlation coefficient between all pairs of nodes i, j as follows:

$$\rho(X_i, X_j) = \frac{\sum_{k=1}^{|\mathcal{A}|} \left((x_i^k - \bar{X}_i)(x_j^k - \bar{X}_j) \right)}{(|\mathcal{A}| - 1)\sigma(X_i)\sigma(X_j)} \tag{3.12}$$

and say that $\rho(X_i, X_j) = 0$ if $\sigma(X_i) = 0$ or $\sigma(X_j) = 0$. The correlation coefficient takes values between -1 and 1.

Using the correlation coefficient, we create the *social correlation graph* (SCG). It is a graph whose nodes are all nodes in $\mathcal{A}$, and each pair of nodes i, j is connected by an edge if $|\rho(X_i, X_j)| > \tau_\rho$, where τ_ρ is a threshold.

3.2.3 Community Detection

Bots are more likely to be connected to each other in the social correlation graph because their interactions with pivotal nodes are correlated. The bot detection problem is now transformed into a community detection problem. By detecting communities in the SCG we can separate the bots from the regular nodes and identify them.

After the SCG has been partitioned into communities, we have to decide which community is the botnet. The interactions between pivotal nodes and bots are expected to be stronger than the interactions between pivotal and regular nodes. Using this idea, we calculate the *pivotal interaction measure* r_i for each node i:

$$r_i = \frac{1}{|\mathcal{A}|} \sum_{k=1}^{|\mathcal{A}|} \sum_{j \in \mathcal{N}} e_j G_k^{ij} \tag{3.13}$$

The community with the highest average pivotal interaction measure is labelled as the botnet.

Chapter 4

Experiments

This chapter presents the experimentation carried out for this thesis, as well as information about the datasets and code used. The experiments test the botnet detection process outlined in the previous chapter with focus on the comparison of different community detection methods on the botnet discovery section.

For each experiment we initially present information about the dataset and outline any pre-processing we performed on the data. Next, we describe the training process and show the results of anomaly detection on the datasets. Finally, the parameters used for the creation of the SCG for each case are mentioned and the results of the different community detection algorithms are shown both visually and numerically.

During the first and third experiment we attempted to replicate the settings used in similar experiments in [26], by using the same datasets. However, in both cases our experiments diverged, having different SCGs for the botnet discovery. The second experiment was entirely original.

The community detection algorithms tested were Girvan-Newman, Louvain algorithm, Spectral Clustering, Walktrap, and Hyperbolic Girvan-Newman. They were evaluated based on the NMI and F1-scores of the botnet detection compared to the ground truth, which was available for all experiments. The F1-score is calculated as follows:

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4.1}$$

where TP the number of true positives (bots that were correctly identified), FP the number of false positives (normal nodes identified as bots) and FN the number of false negatives (bots missed).

4.1 Code implementation

Almost all the code for the experiments was implemented in Python3. The exception was Rigel embedding for the Hyperbolic Girvan-Newman algorithm, which was implemented in C++ by [31]. The HGN code was written by the authors of [5] in Python3 and uses the Rigel code above. We also used Wireshark [91], a network protocol analyzer, for some initial pre-processing of pcap files.

The following Python libraries were used in our implementation:

- **Networkx** [92]: A package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Most of the graph handling in our experiments was done via networkx, including the Girvan-Newman and Louvain algorithms.
- **scikit-learn** [93]: A machine learning library, from which we used the implementations for K-Means clustering, spectral clustering, NMI and F1-score metrics.
- **Matplotlib** [94]: A comprehensive library for data visualisation. All of our plots and figures were created with matplotlib.
- **pandas** [95]: A package for data analysis and manipulation. Used for processing our data when not in graph format.
- **CDlib** [96]: A specialised community detection library, from which we used the walktrap implementation.

4.2 Experiments on CTU-13 dataset

The first experiment we did started as an attempt to replicate the similar experiment done in [26], however the parameters used for anomaly detection were not specified so our results quickly diverged. This experiment utilises a netflow dataset and uses the flow-based anomaly detection method. The size of the dataset was quite large so it had to be scaled down significantly.

4.2.1 Dataset description

CTU-13 is a dataset captured in the Czech Technical University [32]. It includes 13 network captures with botnet traffic. The pcap captures were converted into bidirectional netflow files and each flow was labelled as 'botnet', 'normal' or 'background' by the authors of the dataset.

We use the 2nd capture, which consists of 4.5 hours of traffic with a spamming IRC-based botnet. IRC bots connect to IRC as clients appearing as normal users, but are remotely controlled in order to spread malware or conduct DDoS attacks.

4.2.2 Data pre-processing of CTU-13 dataset

The netflow file we used from the CTU-13 dataset was 241MB in size containing 1808122 labelled flows, created from a 60GB pcap file of almost 72 million packets. 1.04% of those flows are bot flows, 98.33% background and the rest are normal. Each flow had the following attributes:

- **StartTime:** The timestamp of the beginning of the flow.
- **Dur:** The duration of the flow in seconds.
- **Proto:** The protocol used.
- **SrcAddr:** The address of the source host.
- **Sport:** The port number of the source address.
- **Dir:** The direction of communication. '->' denotes that the communication was one-way, while '<->' denotes that both hosts sent packets to each other.
- **DstAddr:** The address of the destination host.
- **Dport:** The port number of the destination address.
- **State:** Unused.
- **sTos:** Unused.
- **dTos:** Unused.
- **TotPkts:** Total number of packets exchanged.

- **TotBytes:** Total number of bytes exchanged.
- **SrcBytes:** Number of bytes sent from the source to the destination.
- **Label:** The label of the flow: normal, botnet or background. Also contains a summary of the communication.

For our experimentation we limited ourselves to IPv4 flows. Using the Proto attribute we filtered out the "ipv6", "ipv6-icmp", "ipx/spx" and "rarp" flows. We then checked the source and destination addresses for any remaining IPv6 addresses, by checking if the address contained the ":" symbol, and removed them. The remaining flows were 1808056.

We then removed the Proto, Dir, State, sTos, dTos and TotPkts columns, and replaced the TotBytes column with a new column called **DstBytes**. DstBytes's value was calculated by subtracting the SrcBytes from TotBytes, and the result corresponds to the number of bytes sent from the destination address to the source address. We also added another column called **EndTime**, equal to the timestamp when the flow ended, whose value was calculated by adding the duration of the flow and the starting time from the corresponding columns.

Figure 4.1 shows the first five rows of the dataset after processing.

	StartTime	EndTime	Dur	SrcAddr	DstAddr	Sport	Dport	SrcBytes	DstBytes	Label
0	2011/08/11 09:49:35.721274	2011/08/11 10:24:05.694419	2069.973145	203.253.8.233	147.32.84.229	30533	13363	123	74	flow=Background-UDP-Established
1	2011/08/11 09:49:35.721530	2011/08/11 10:04:31.710788	895.989258	81.47.154.13	147.32.84.229	49200	13363	4501	2531	flow=Background
2	2011/08/11 09:49:35.721918	2011/08/11 09:49:35.722038	0.000120	147.32.84.229	78.42.25.171	13363	42988	2858	0	flow=Background-UDP-Attempt
3	2011/08/11 09:49:35.722518	2011/08/11 10:48:57.649764	3561.927246	147.32.84.229	113.128.219.130	13363	59790	13419	2351	flow=Background-UDP-Established
4	2011/08/11 09:49:35.723816	2011/08/11 09:49:35.723816	0.000000	147.32.84.229	60.50.167.24	13363	40253	60	0	flow=Background-UDP-Attempt

Figure 4.1. *Snapshot of the CTU-13 dataset after processing.*

4.2.3 Training period on CTU-13 dataset

The training period we used for the CTU-13 dataset were the first 25 minutes of flows, where there was no bot traffic. The total number of training flows were 257718 (14% of the dataset).

Since the file was in netflow format we used the flow method for anomaly detection. The following attributes of each flow were quantized, in accordance with the similar experiment of [26]: source address, destination address, source port, destination port, duration, source bytes, destination bytes.

The source and destination IP addresses were quantized using K-Means clustering with 256 clusters, as described in 2.3.6. Before the algorithm was applied, the IP addresses were transformed into base10 integers using the following formula:

$$x_{10} = \sum_{i=1}^{4} 256^{4-i} a_i \tag{4.2}$$

where $a = (a_1, a_2, a_3, a_4) \in \{0, 1, ..., 255\}^4$ an IP address. The 117619 training addresses were used to train the K-Means model, which will be used to fit the evaluation addresses into their nearest cluster during evaluation.

The source port and destination port columns were quantized as follows:

$$q_{ports}(x) = \begin{cases} -1 & \text{for } x \leq 0 \text{ or missing or invalid} \\ 0 & \text{for } x < 1024 \text{ (registered port)} \\ 1 & \text{for } 1024 \leq x < 10000 \\ 1 + x \text{ div } 10000 & \text{for } x \geq 10000 \end{cases} \tag{4.3}$$

where x is a port number.

The number x of source and destination bytes was quantized using the following function:

$$q_{bytes}(x) = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } 1 \leq x < 100 \\ 2 & \text{for } 100 < x < 1000 \\ 3 & \text{for } 1000 < x < 10000 \\ 4 & \text{for } x \geq 10000 \end{cases} \tag{4.4}$$

Similarly, the duration x of each flow was quantized as follows:

$$q_{dur}(x) = \begin{cases} 0 & \text{for } 0 \leq x < 1 \\ 1 & \text{for } 1 < x < 10 \\ 2 & \text{for } 10 < x < 100 \\ 3 & \text{for } 100 < x < 1000 \\ 4 & \text{for } x \geq 1000 \end{cases} \tag{4.5}$$

Figure 4.2 shows the first five rows of the training dataset after quantization.

	Dur	SrcAddr	DstAddr	Sport	Dport	SrcBytes	DstBytes
0	4	245	140	4	2	2	1
1	3	151	140	5	2	3	3
2	0	140	135	2	5	3	0
3	4	140	128	2	6	4	3
4	0	140	213	2	5	1	0

Figure 4.2. *Snapshot of the quantized training data from the CTU-13 dataset.*

After the data has been quantized, each flow is now represented by a vector of 7 integers. The values that the flow vector takes become the sample space of the reference discrete random variable that represents the flows. An empirical probability distribution is created by dividing the number of appearances of each vector value by the total number

of training flows.

4.2.4 Anomaly detection on the CTU-13 dataset

A 5 minute evaluation period is selected 26 minutes after the end of the training period (51-56 minutes into the dataset). The evaluation dataset is quantized just like the training dataset, with the exception of the IP addresses now being fitted into the already trained KMeans model. Afterwards, the 5 minute evaluation period is split into 2 second time windows. The distribution of vector values for each 2 second time window is computed and compared to the reference distribution using KL divergence 3.1.

The results of the anomaly detection process are shown in figure 4.3. The blue line corresponds to the number of bot flows per 2 second window, multiplied by 0.005 for visualisation purposes, while the orange line corresponds to the KL divergence of each 2 second window to the normal training traffic. The KL divergence is higher in time windows with a lot of bot traffic, so by picking a threshold around 0.1 we can detect most of the bot windows.

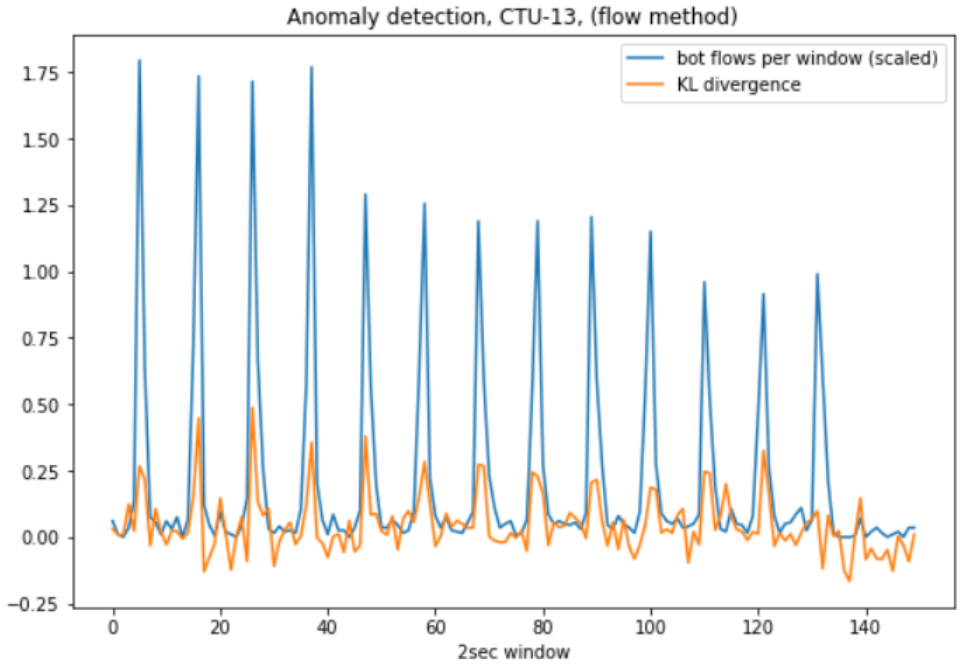

Figure 4.3. *Anomaly detection on the CTU-13 dataset.*

4.2.5 Creation of the SCG for the CTU-13 dataset

For the botnet detection process we require the unquantized value of the source and destination IP addresses in order to create the interaction records. As the size of the anomalous set increases, so does the number of unique IP addresses inside it, so for simplification purposes we only evaluated a small anomalous time period between windows 47-55 shown in figure 4.3. This time period contained 936 unique IP addresses, out of which 119 were bots.

The total interaction measure e_i of all 936 nodes is shown in figure 4.4. Each red cross "x" represents one node. We can clearly see that there exist a few nodes with a much larger interaction than the rest. We use a threshold of $\tau = 20$ for determining the pivotal nodes. Four nodes are designated as pivotal, of which the two middle ones belong to the botnet while the first and fourth are victims.

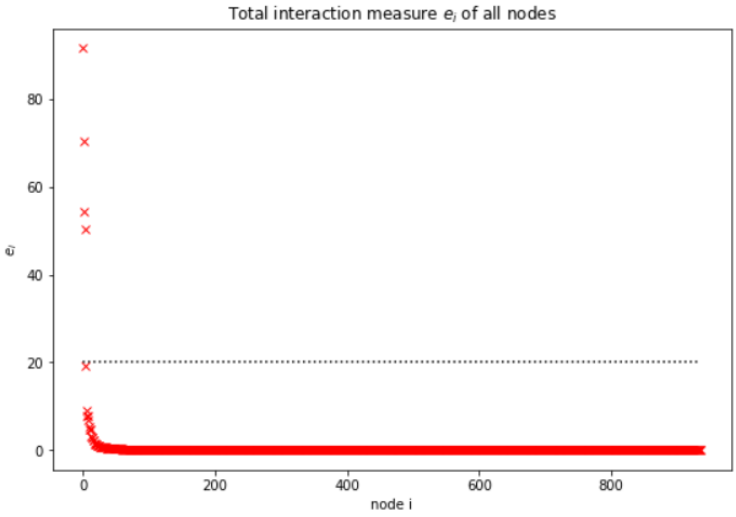

Figure 4.4. *Total interaction measure of all nodes in $\mathcal{A}$. The four nodes above the dotted line are the pivotal nodes.*

To create the SCG we use a strict threshold $\tau_\rho = 0.88$ for the correlation in order for the SCG to be relatively small. There are multiple connected components to it, so the community detection algorithm should be applied to each of them separately, but we will only show the largest connected component. The resulting graph has 103 nodes, out of which 8 are bots, as seen in figure 4.5. The red nodes correspond to the bots.

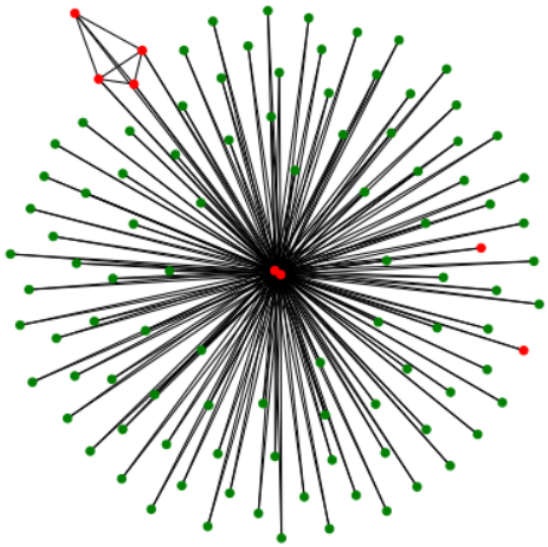

Figure 4.5. *Largest connected component of the social correlation graph with $\tau_\rho = 0.88$. The bots are marked in red.*

For comparison purposes we include an SCG created with $\tau_\rho = 0.5$ (figure 4.6). In this case τ_ρ is too small to distinguish the bots into communities, however more bots are present in the SCG.

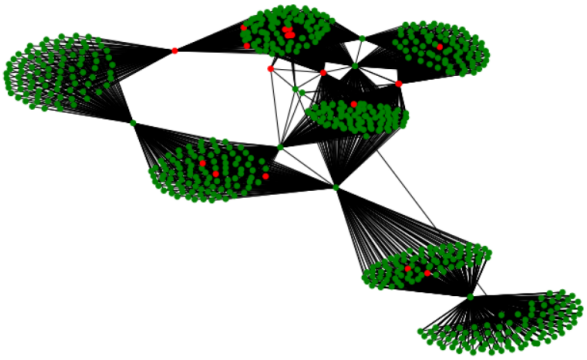

Figure 4.6. *Social correlation graph with* $\tau_\rho = 0.5$

4.2.6 Comparison of community detection methods on the CTU-13 dataset

After the SCG has been created, we apply the 5 community detection algorithms on it. For the HGN algorithm we use 3 landmark nodes and a batch size of 1. The results are shown in figure 4.7. All algorithms except spectral clustering are able to detect the community of four bots on the top left. Spectral clustering detects the two central bots, while no algorithm is able to detect the two bots at the right side of the SCG.

The NMI and F1-scores of each method are displayed in table 4.1. Since spectral clustering only detects two bots, its scores are lower than the other algorithms.

Algorithm	NMI	F1-score
Girvan-Newman	0.5	0.67
Louvain	0.5	0.67
Spectral Clustering	0.28	0.44
Walktrap	0.5	0.67
HGN	0.5	0.67

Table 4.1. *NMI and F1-scores of community detection algorithms on the SCG*

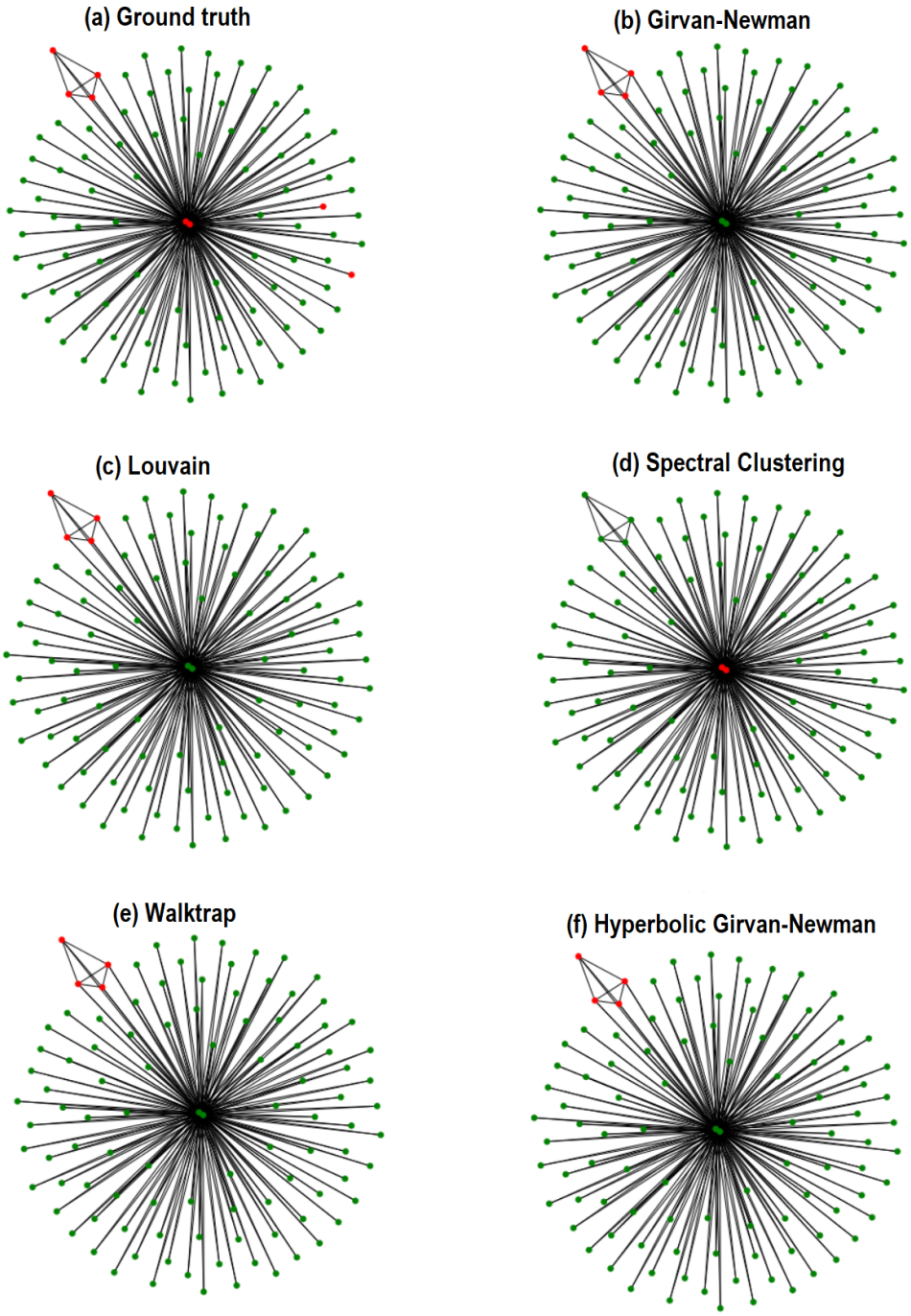

Figure 4.7. *Results of community detection algorithms on the SCG*

4.3 Experiments on Kitsune dataset

The second experiment was carried out on a dataset that was in pcap format, so the second anomaly detection method was used. This experiment was not replicated from [26].

4.3.1 Dataset description

The Kitsune Network Attack Dataset [33] consists of nine traffic captures of various network attacks. We used one capture where an IoT network was infected with Mirai malware. The network, connected over Wi-Fi, consists of three PCs, a webcam, a baby monitor, two doorbells, a thermostat and four security cameras, one of which was infected with Mirai botnet malware mentioned in 2.5.1.

The capture includes around 1 million packets of clean traffic before the botnet attack begins. The authors of the dataset provide a csv file labelling nefarious packets as 1 and normal packets as 0.

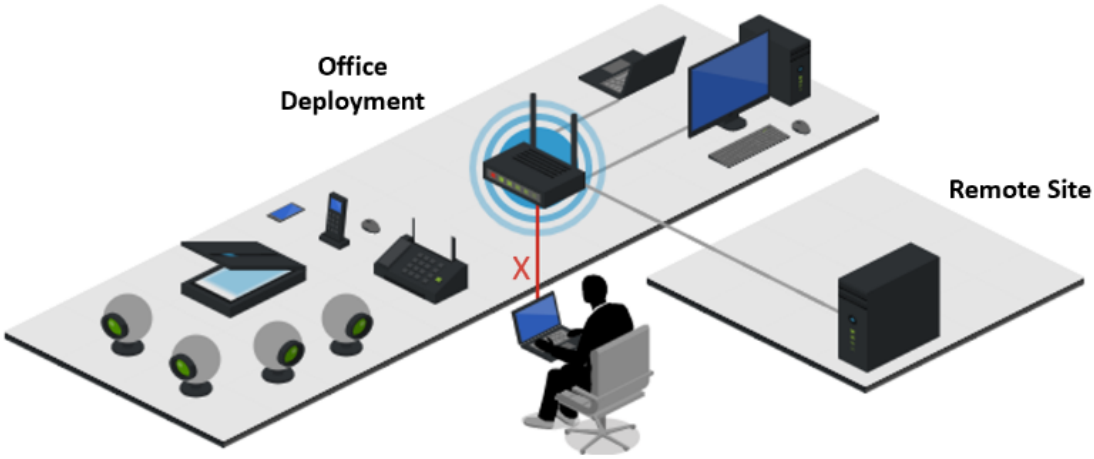

Figure 4.8. *Setup of IoT network used for the Kitsune dataset [6]*

4.3.2 Data pre-processing of Kitsune dataset

From the Kitsune dataset we used a 73MB pcap file containing the traffic data of 764137 packets and a 2.2MB csv file containing the label for each packet. We converted the pcap file into a csv file via the Wireshark program in order to parse it easier. The first 5 rows of the dataset are shown in 4.9

Each packet had the following attributes:

- **No.:** Index number of packet.
- **Time:** Timestamp of packet in YYYY-MM-DD hh:mm:ss.ffffff format
- **Source:** Address of source host.
- **Destination:** Address of destination host.

- **Protocol:** The protocol used.
- **Length:** The size of the packet in bytes.
- **Info:** A summary of the contents of the packet

	No.	Time	Source	Destination	Protocol	Length	Info
0	1	2018-10-25 05:46:22.933899	192.168.2.108	52.24.43.67	TCP	60	21074 > http(80) [SYN] Seq=0 Win=5840 Len=0 ...
1	2	2018-10-25 05:46:22.933904	192.168.2.108	52.25.66.250	TCP	60	20532 > synapse-nhttp(8280) [SYN] Seq=0 Win=...
2	3	2018-10-25 05:46:22.934426	192.168.2.1	192.168.2.108	ICMP	86	Destination unreachable (Network unreachable)
3	4	2018-10-25 05:46:22.934636	192.168.2.1	192.168.2.108	ICMP	86	Destination unreachable (Network unreachable)
4	5	2018-10-25 05:46:23.291054	48:02:2e:01:83:15	Broadcast	ARP	60	Who has 192.168.2.106? Tell 192.168.2.109

Figure 4.9. *First 5 rows of the Kitsune dataset*

Again, we limit our experiment to IPv4 packets so we filtered out all the other packets, leaving 197701 remaining ones. We then removed all columns except Time, Source and Destination, and added the label for each packet. The first five rows of the dataset after processing are shown in figure 4.10.

	Time	Source	Destination	Label
0	2018-10-25 05:46:22.933899	192.168.2.108	52.24.43.67	0.0
1	2018-10-25 05:46:22.933904	192.168.2.108	52.25.66.250	0.0
2	2018-10-25 05:46:22.934426	192.168.2.1	192.168.2.108	0.0
3	2018-10-25 05:46:22.934636	192.168.2.1	192.168.2.108	0.0
4	2018-10-25 05:46:23.367591	192.168.2.101	192.168.2.110	0.0

Figure 4.10. *First 5 rows of the Kitsune dataset after processing*

4.3.3 Training on Kitsune dataset

The training period for the Kitsune dataset was the first hour of packets, when there was no bot traffic. The total number of training packets was 90476 (46% of the dataset), including 76 unique IP addresses.

We split the training data into 10 second windows and randomly selected 25 windows. For each of the selected windows we created the interaction graph as described in section 3.1.2. We then sampled the degree of 10 random nodes from each interaction graph, resulting in a sample size of $|\mathcal{D}| = 250$ node degrees.

We computed the L_{Rand} and L_{PA} and selected the random graph model with $\hat{\eta} = 2.664$.

4.3.4 Anomaly detection on Kitsune dataset

The evaluation period for this dataset was 20 minutes long and started 3 minutes after the end of the training period (63-83 minutes into the dataset). It contained 33903

packets, out of which 16829 were bot packets and started appearing a little bit after the 10 minute mark.

We split the evaluation dataset into 10 second windows and calculated the I_{Rand} score of each compared to the reference normal traffic collected during training. The results of anomaly detection are shown in figure 4.11. The blue line corresponds to the number of bot packets per 10 second window, multiplied by 0.004 for visualisation purposes, while the orange line corresponds to the I_{Rand} value of each window compared to the normal traffic. We can clearly see a small increase in level when the bots start appearing. By selecting a threshold of 0.78 (dotted line) we detect a good number of bot windows with only few false positives.

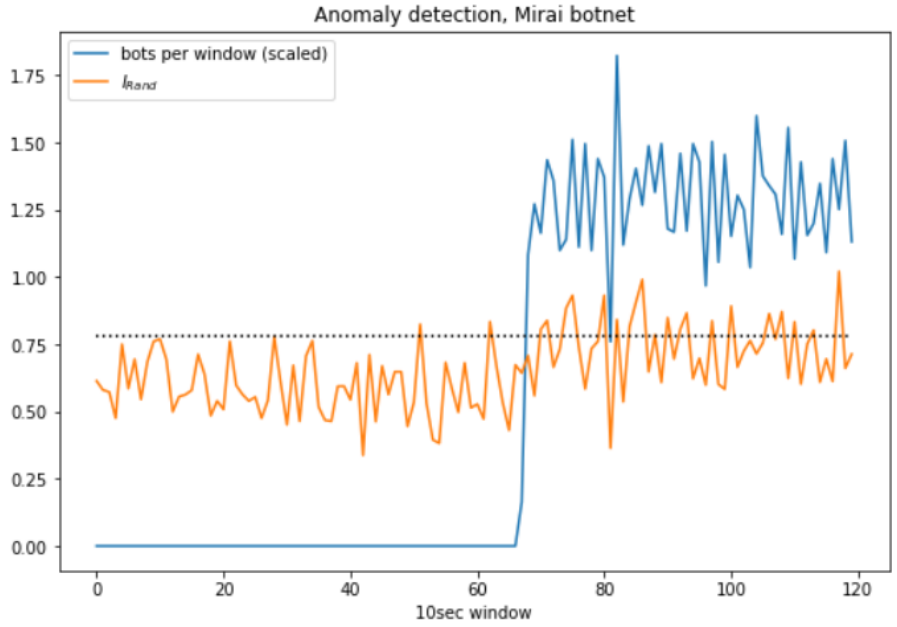

Figure 4.11. *Anomaly detection on the Kitsune dataset. The dotted line is the anomaly threshold.*

4.3.5 Creation of the SCG of the Kitsune dataset

We experimented with 2 separate anomalous sets on the Kitsune dataset. The first one was created from 10 anomalous windows around the 15 minute mark, while the second was created from the first 20 anomalous windows.

Anomalous period around 15 minutes

The anomalous set created around 15 minutes contained 44 unique IP addresses, out of which 18 were bot addresses. The interaction graph during those 10 windows is shown in figure 4.12.

We calculated the total interaction measure of each node in the anomalous period and selected 5 pivotal nodes above a threshold of $\tau = 20$, as shown in figure 4.13. Each red cross "x" represents one node, and the threshold is the dotted line. All 5 of the pivotal nodes above the threshold were bots.

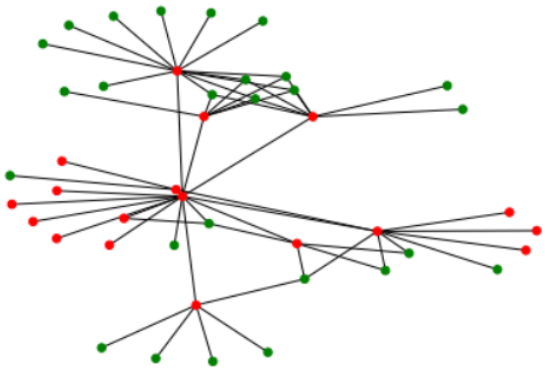

Figure 4.12. *Interaction graph of 10 anomalous windows in the Kitsune dataset. Bots are marked in red.*

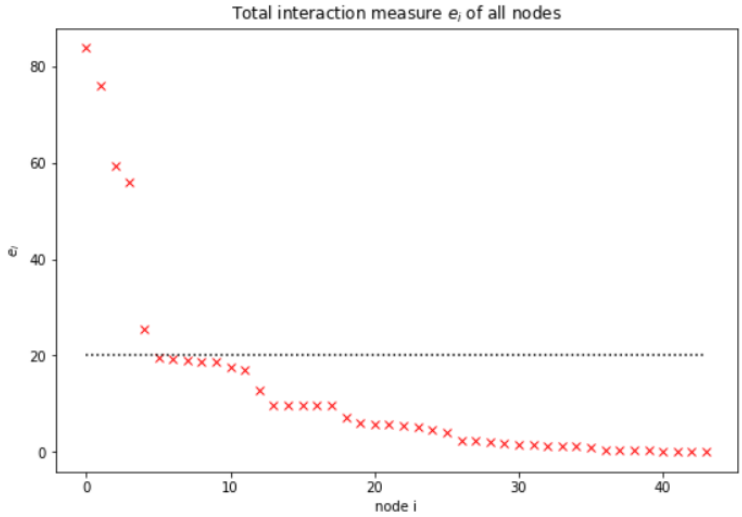

Figure 4.13. *Total interaction measure of all nodes in $\mathcal{A}$. The five nodes above the dotted line are the pivotal nodes.*

For the creation of the SCG we used a threshold $\tau_\rho = 0.65$. The SCG contained 27 nodes, out of which 13 were bots. It was split into 3 connected components, one of which only had 3 nodes. The SCG is shown in figure 4.14. We can see that in the largest connected component the bots form a community and are well separated from the normal nodes.

First 20 anomalous windows

The anomalous set created from the first 20 anomalous windows contained 52 IP addresses, out of which 18 were bots. The interaction graph during this period is shown in figure 4.15.

A threshold of $\tau = 20$ was used to select 4 pivotal nodes, all of which were bots. For the SCG we used $\tau_\rho = 0.47$, resulting in an SCG with 29 nodes, of which 10 were bots, as shown in figure 4.16.

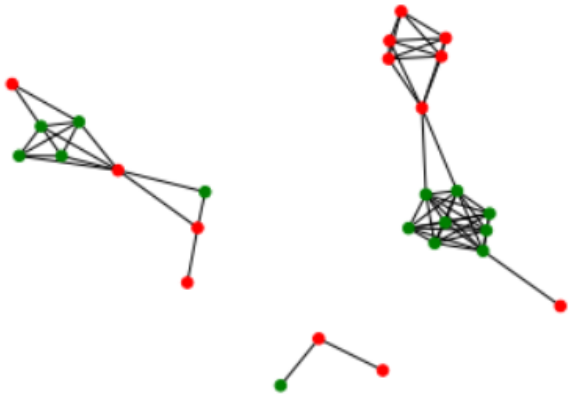

Figure 4.14. *SCG of 10 anomalous windows. Bots are marked in red.*

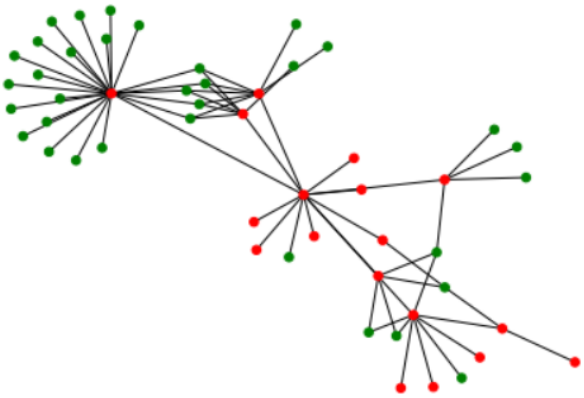

Figure 4.15. *Interaction graph of the first 20 anomalous windows*

4.3.6 Comparison of community detection methods on the Kitsune dataset

We now apply the 5 community detection algorithms on the SCGs. For the HGN algorithm we used 3 landmarks and a batch size of 1.

Anomalous period around 15 minutes

For the SCG in figure 4.14 we applied the algorithms on both the largest and second largest connected components. The results for the first component are shown in figure 4.17 and table 4.2. We can see that all algorithms are able to detect the large bot community, but only spectral clustering detects the single bot on the bottom right.

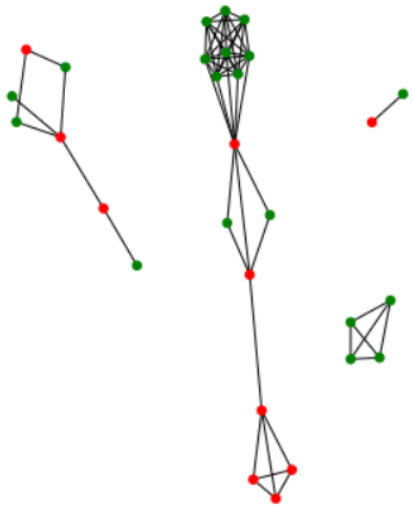

Figure 4.16. *SCG of the first 20 anomalous windows*

Algorithm	NMI	F1-score
Girvan-Newman	0.71	0.94
Louvain	0.71	0.94
Spectral Clustering	1	1
Walktrap	0.71	0.94
HGN	0.71	0.94

Table 4.2. *NMI and F1-scores of community detection algorithms on the largest connected component of the SCG*

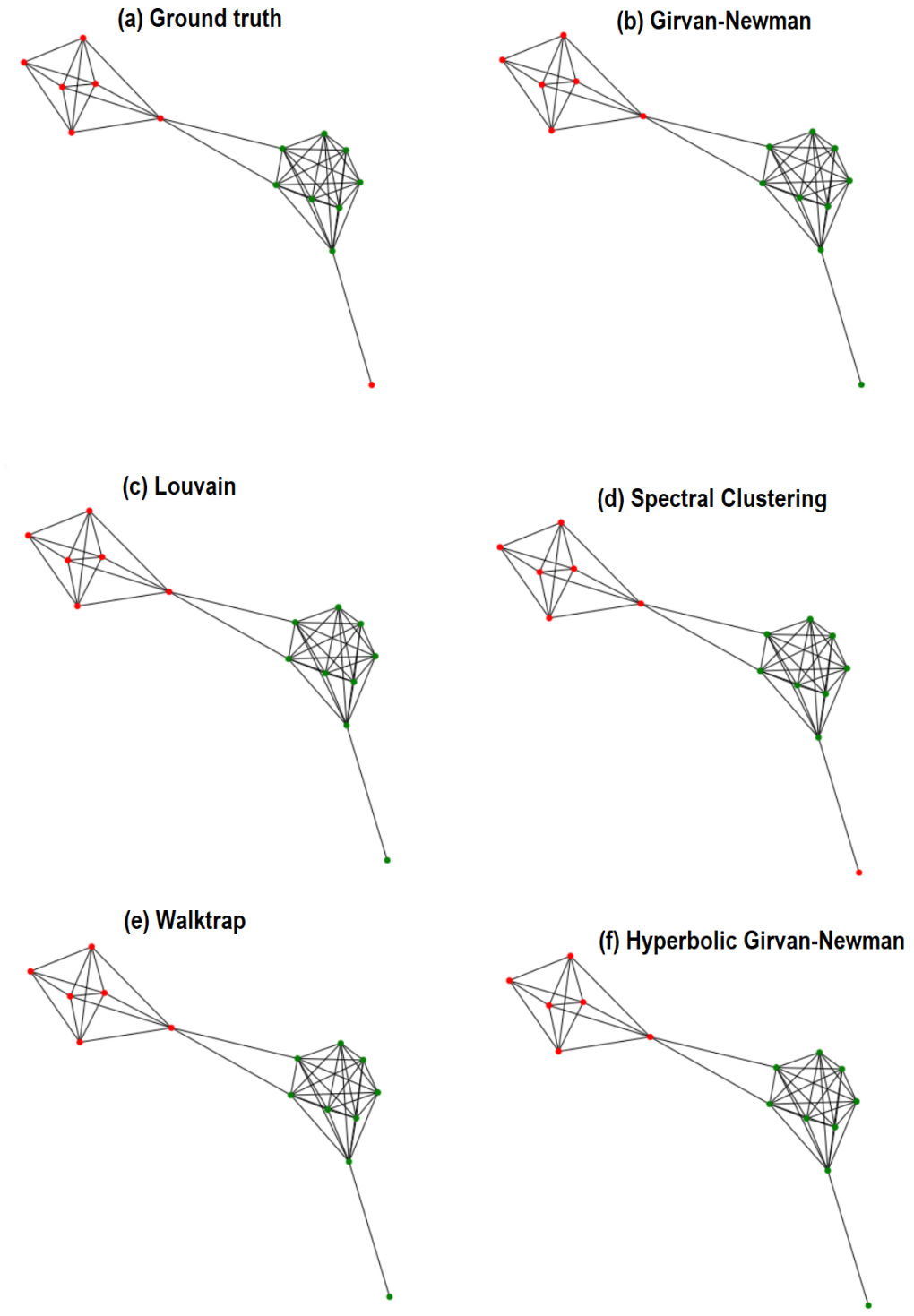

Figure 4.17. *Results of community detection algorithms on the largest connected component of the SCG*

For the second connected component the results are shown in figure 4.18 and table 4.3. All algorithms except Louvain detect 2 out of 4 bots, while Louvain detects 3. Every algorithm has a single false positive.

Algorithm	NMI	F1-score
Girvan-Newman	0.08	0.57
Louvain	0.23	0.75
Spectral Clustering	0.08	0.57
Walktrap	0.08	0.57
HGN	0.08	0.57

Table 4.3. *NMI and F1-scores of community detection algorithms on the second connected component of the SCG*

First 20 anomalous windows

For the SCG of the first 20 anomalous windows, in figure 4.16, we applied the community detection algorithms only on the first connected component. The results are shown in figure 4.19 and table 4.4.

In this case the algorithms diverge a lot more and have worse results than the previous cases, which is expected since the anomalous period contains more non-bot interactions. Spectral clustering seems to have the best results, identifying all but 1 bot and having only 2 false positives. The Girvan-Newman algorithms separate the communities quite well, based on the high NMI score, but identify the wrong community as bots. The reason becomes clearer once we look at the results of Louvain and Walktrap. These two algorithms detect more than 2 communities, and in the end select the one in the middle as the bot community while the others are classified as normal nodes. This selection shows that the middle nodes have a high pivotal interaction measure compared to the 4 bots on the top left. For this reason the bot community of all algorithms includes the middle nodes.

Algorithm	NMI	F1-score
Girvan-Newman	0.53	0.13
Louvain	0.02	0.63
Spectral Clustering	0.3	0.57
Walktrap	0.01	0.22
HGN	0.53	0.13

Table 4.4. *NMI and F1-scores of community detection algorithms on the SCG of the first 20 anomalous windows*

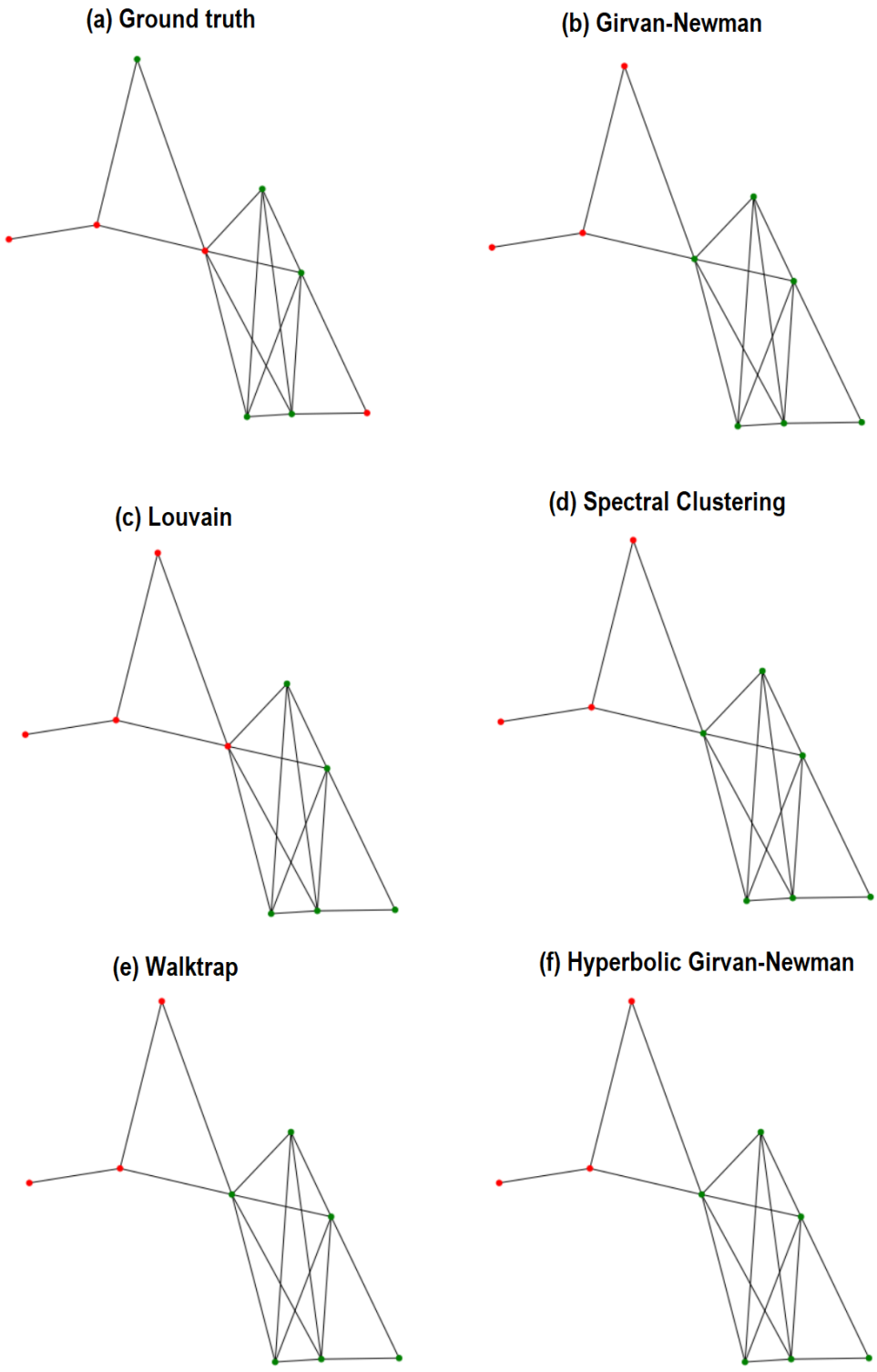

Figure 4.18. *Results of community detection algorithms on the second connected component of the SCG*

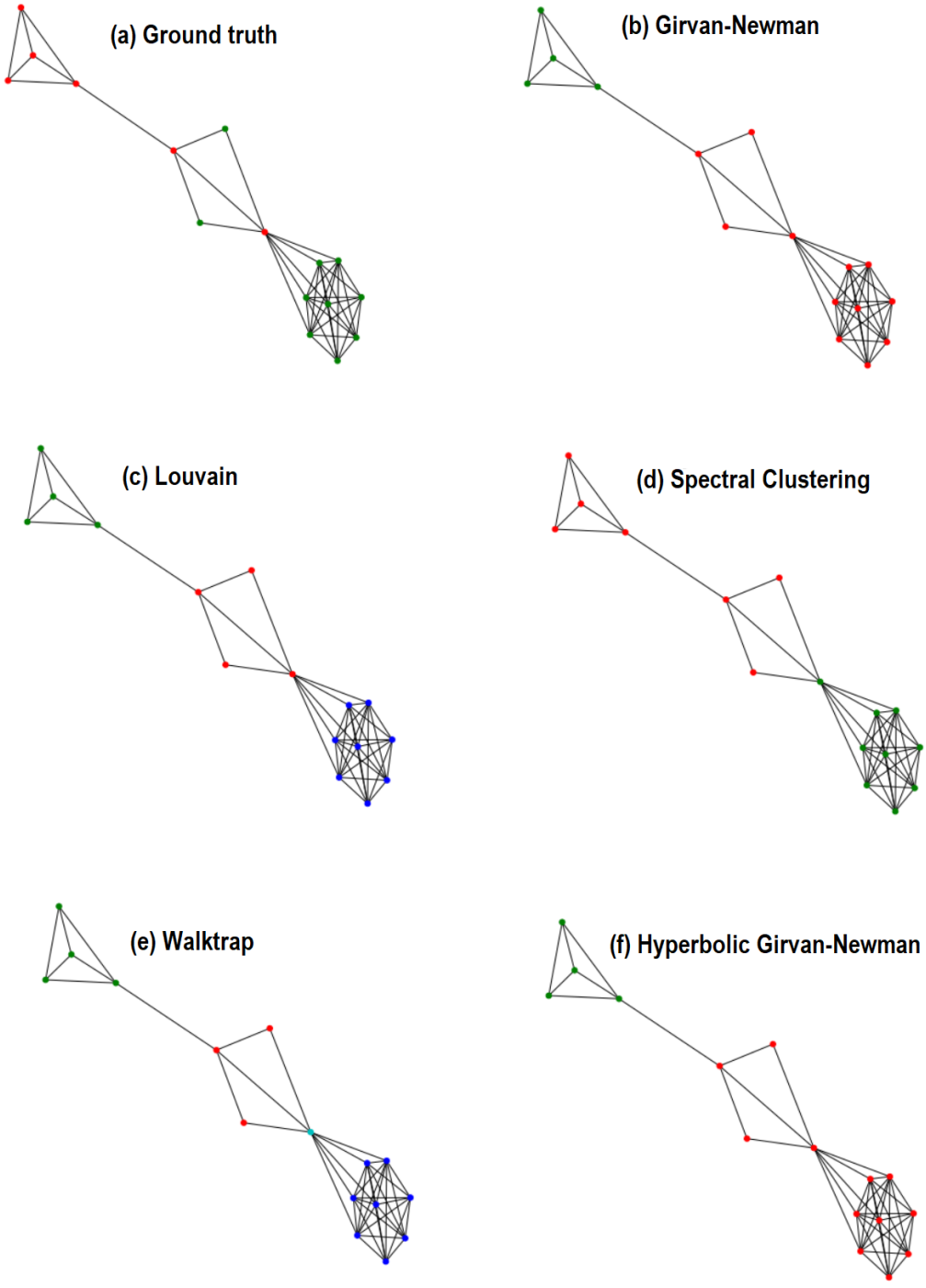

Figure 4.19. *Results of community detection algorithms on the second connected component of the SCG*

4.4 Experiments on the CAIDA and Simpleweb datasets

For the final experiment we mixed traffic from two different datasets, one containing normal background traffic and one containing only the attack. This experiment began as a replication of the 2nd experiment of [26], but the mixing was done differently so the resulting graphs were also different.

4.4.1 Dataset descriptions

CAIDA dataset

For the botnet traffic we used the CAIDA UCSD "DDoS Attack 2007" Dataset [34]. It contains an hour of traffic from a DDoS attack that happened in 2007 split into 5-minute pcap files. Only the attacking packets are included in the dataset; the background traffic has been removed by the authors.

The total size of the dataset is 21GB. During the first half hour the bots probe their targets to find vulnerabilities, so the traffic is relatively low, whereas during the second half the DDoS attack is carried out with much higher bot traffic. For our experiment we used the first 5-minute pcap file of size 8.6MB.

Simpleweb dataset

For the background traffic, like the authors of [26], we used trace 6 in the University of Twente traffic traces data repository (simpleweb) [35]. This trace, which is around 192MB in size, was measured from a 100 Mbit/s Ethernet link connecting an educational organization to the internet. This is a small organization with 100 workstations that have a 100Mbit/s Lan connection while the core network has a 1 Gbit/s connection. The recordings took place between the external optical fiber modem and the first firewall and the measured link was only mildly loaded during this period.

4.4.2 Data pre-processing of CAIDA and Simpleweb datasets

Both files for this experiment were pcap files. In order to mix them, we changed the timestamp of each packet in Wireshark to the time that passed (in seconds) since the capturing of the first packet. We then converted the pcap files into csv files and kept only the Time, Source and Destination columns like we did with the Kitsune dataset in 4.3.2. We added a label column, with values 0 for the Simpleweb dataset and 1 for the CAIDA dataset. For the Simpleweb dataset we only kept the first hour of traffic. To mix the datasets we added 2000 seconds to the timestamp of each packet in the CAIDA dataset, inserting the botnet traffic between 2000 and 2300 seconds of the background traffic.

The Simpleweb dataset contained 329967 packets and 743 unique IP addresses, while the CAIDA dataset contained 166448 packets from 136 IP addresses. Since the bot traffic is way denser than the background traffic, we kept only 6000 bot packets selected randomly, resulting in a total dataset of 335967 packets. Another problem we had to solve was that the IP addresses of the CAIDA dataset were distinct from the Simpleweb ones.

To mix them, we randomly mapped the 136 CAIDA addresses into Simpleweb addresses and changed them accordingly. We experimented with two different mappings, one of which mapped 3 bot addresses into 1 normal address resulting in 45 bots, and one which mapped 5 bot addresses into 1 normal address resulting in 27 bots.

4.4.3 Training period for the mixed dataset

The training period for this experiment was the first 20 minutes of traffic, when there were no bot packets. The total number of training packets was 198440 (59% of the dataset) with 539 unique IP addresses.

We split the training data into 10 second windows and randomly selected 50 windows to create the interaction graph. We then sampled the degree of 10 random nodes from each interaction graph, creating a sample of $|\mathcal{D}| = 500$ node degrees. We computed the L_{Rand} and L_{PA} and selected the random graph model with $\hat{\eta} = 1.628$.

4.4.4 Anomaly detection on the mixed dataset

The evaluation period for this dataset was 20 minutes long and started 5 minutes after the end of the training period (1500-2700 seconds into the dataset). It contained 32763 packets, 6000 of which were attack packets that appeared around the middle of the evaluation period (2000-2300 seconds).

Like we did with the Kitsune dataset in section 4.3.4, we split the evaluation data into 10 second windows and calculated the I_{Rand} score. The results of anomaly detection for 45 and 27 bots are shown in figures 4.20 and 4.21 respectively. The blue line corresponds to the number of bot packets per 10 second window multiplied by 0.02, while the orange line corresponds to the I_{Rand} value of the window. We can clearly see that in the 27 bot case anomaly detection is more difficult because the bot traffic is not distinct enough from the normal traffic. Nevertheless we selected a threshold of 1.5 for the 45 bot case and 1.2 for the 27 bot case in order to detect a decent number of bot windows with few false positives.

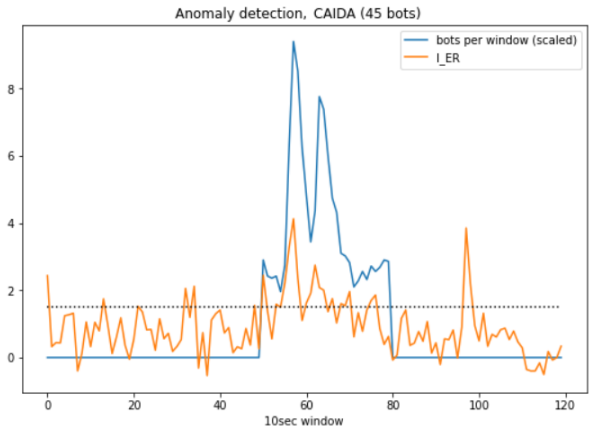

Figure 4.20. *Anomaly detection in the mixed dataset with 45 bots. The dotted line is the anomaly threshold.*

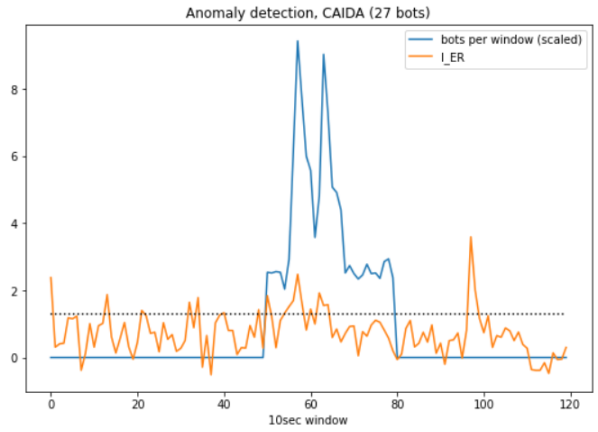

Figure 4.21. *Anomaly detection in the mixed dataset with 27 bots. The dotted line is the anomaly threshold.*

4.4.5 Creation of the SCG for the mixed dataset

We experimented with two anomalous sets on the mixed dataset, one for the 45 bot case and one for the 27 bot case.

45 bot case

The anomalous set created from the 45 bot case contained 94 IP addresses, of which 44 were bots, so only a single bot was missed. The interaction graph is shown in figure 4.22.

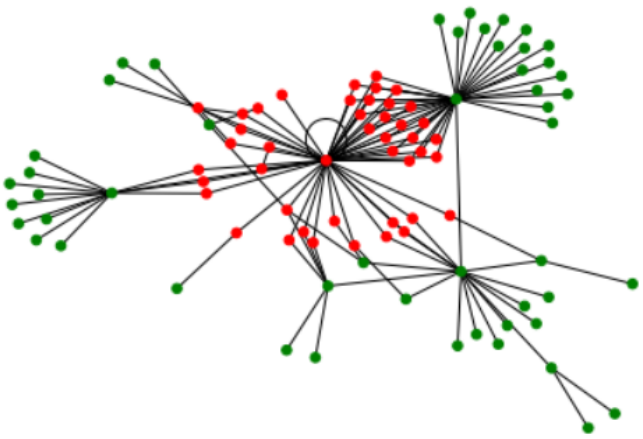

Figure 4.22. *Interaction graph of the anomalous set from the 45 bot case of the mixed dataset. Bots are marked in red.*

We calculated the total interaction measure of each node and selected a threshold $\tau = 30$ resulting in 4 pivotal nodes, of which only the third was a bot. To create the SCG we used a threshold $\tau_\rho = 0.38$, resulting in a graph of 48 nodes including 41 bots. The SCG is shown in figure 4.23, with a clear separation between the bots and the normal nodes.

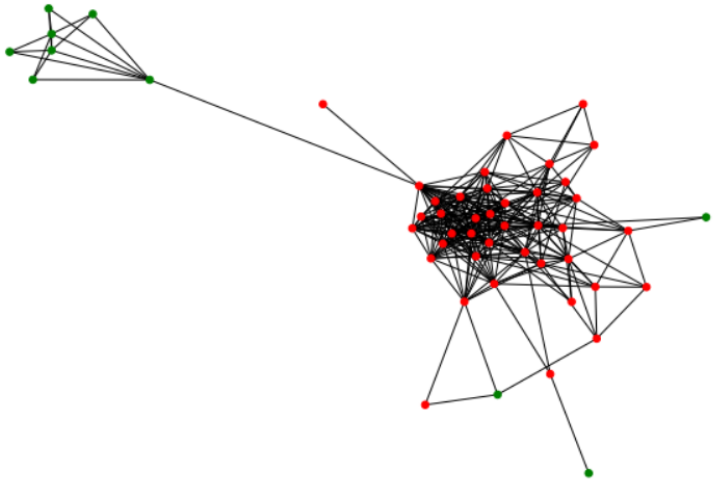

Figure 4.23. *SCG of the anomalous set from the 45 bot case of the mixed dataset. Bots are marked in red.*

27 bot case

Since anomaly detection failed in the 27 bot case, we handpicked the first half anomalous windows resulting in an interaction graph with 101 nodes, 20 of which were bots. We used a threshold $\tau = 30$ to select 5 pivotal nodes, of which 2 were bots. For the SCG we used $\tau_\rho = 0.3$ resulting in a graph of 31 nodes with 16 bots. The SCG is shown in figure 4.24. The bots arent as separated from the normal nodes as the previous case but there is a distinct bot community in the middle.

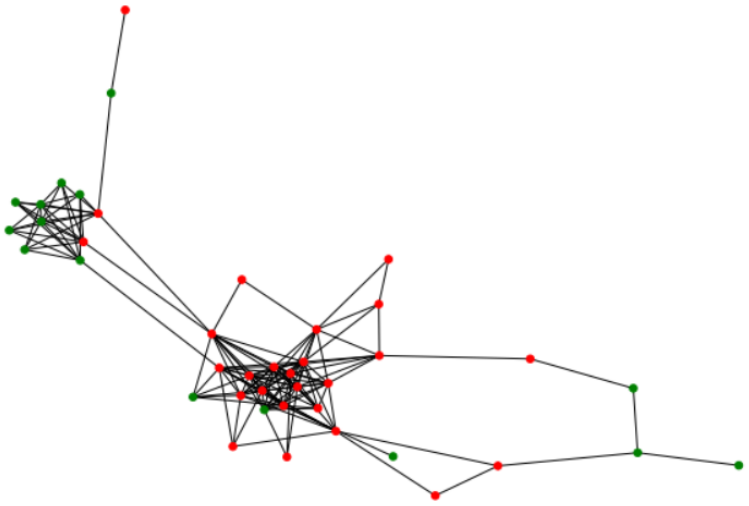

Figure 4.24. *SCG of the anomalous set from the 27 bot case of the mixed dataset. Bots are marked in red.*

4.4.6 Comparison of community detection methods on the CAIDA dataset

We now apply the community detection algorithms on the two SCGs. For the HGN algorithm we used a batch size of 1 in both cases, 3 landmarks for the 45 bot case and 10 landmarks for the 27 bot case.

The results of community detection in the 45 bot case are shown in figure 4.25 and table 4.5. We can see that both Girvan-Newman algorithms detect the large bot community but the rest of the algorithms struggle. Spectral clustering detects only part of the bot community, while Louvain and Walktrap split the bot community into several small parts and identify only one of them.

Algorithm	NMI	F1-score
Girvan-Newman	0.62	0.96
Louvain	0.18	0.62
Spectral Clustering	0.23	0.53
Walktrap	0	0
HGN	0.62	0.96

Table 4.5. *NMI and F1-scores of community detection algorithms on the mixed dataset (45 bot case)*

The results of community detection in the 27 bot case are shown in figure 4.26 and table 4.6. Again, the Girvan-Newman algorithms have the best results, detecting most of the large bot community. Spectral clustering follows, detecting all the bots but having many false positive results, while while Louvain and Walktrap again identify too many communities and only detect a few bots.

Algorithm	NMI	F1-score
Girvan-Newman	0.21	0.84
Louvain	0.01	0.35
Spectral Clustering	0.03	0.77
Walktrap	0	0
HGN	0.13	0.79

Table 4.6. *NMI and F1-scores of community detection algorithms on the mixed dataset (27 bot case)*

From this experiment it is clear that the Louvain and Walktrap algorithms, that do not specify the number of communities beforehand, perform poorly because of this extra degree of freedom. A possible solution to improve their results is to evaluate each community separately, perhaps based on a threshold on the pivotal interaction measure, whether they are a bot community or not, instead of only selecting the community with the largest value to be the bots. This however adds an extra parameter to the problem, which we can bypass by selecting one of the other algorithms that strictly detect two communities.

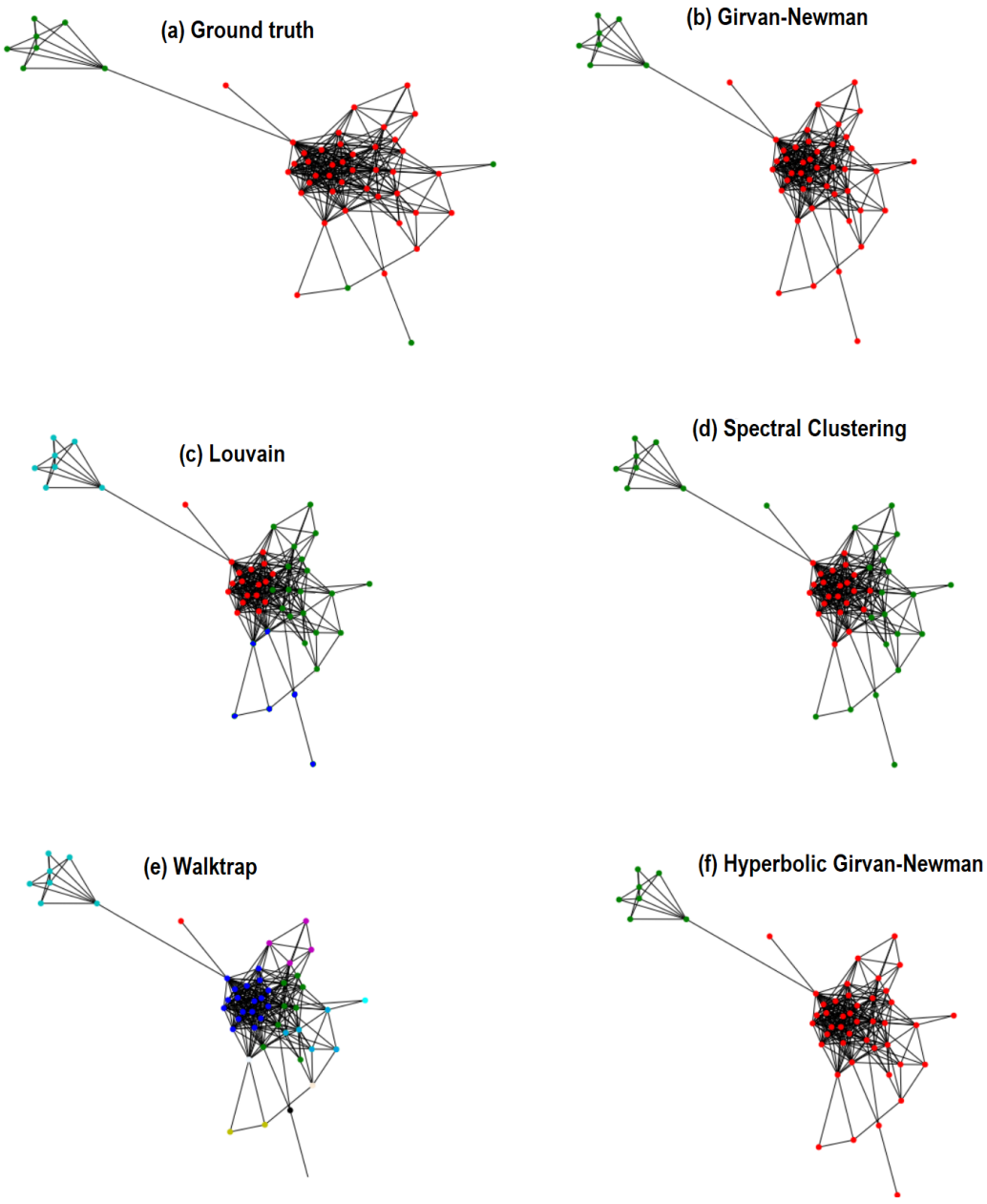

Figure 4.25. *Results of anomaly detection on the SCG of the mixed dataset (45 bot case)*

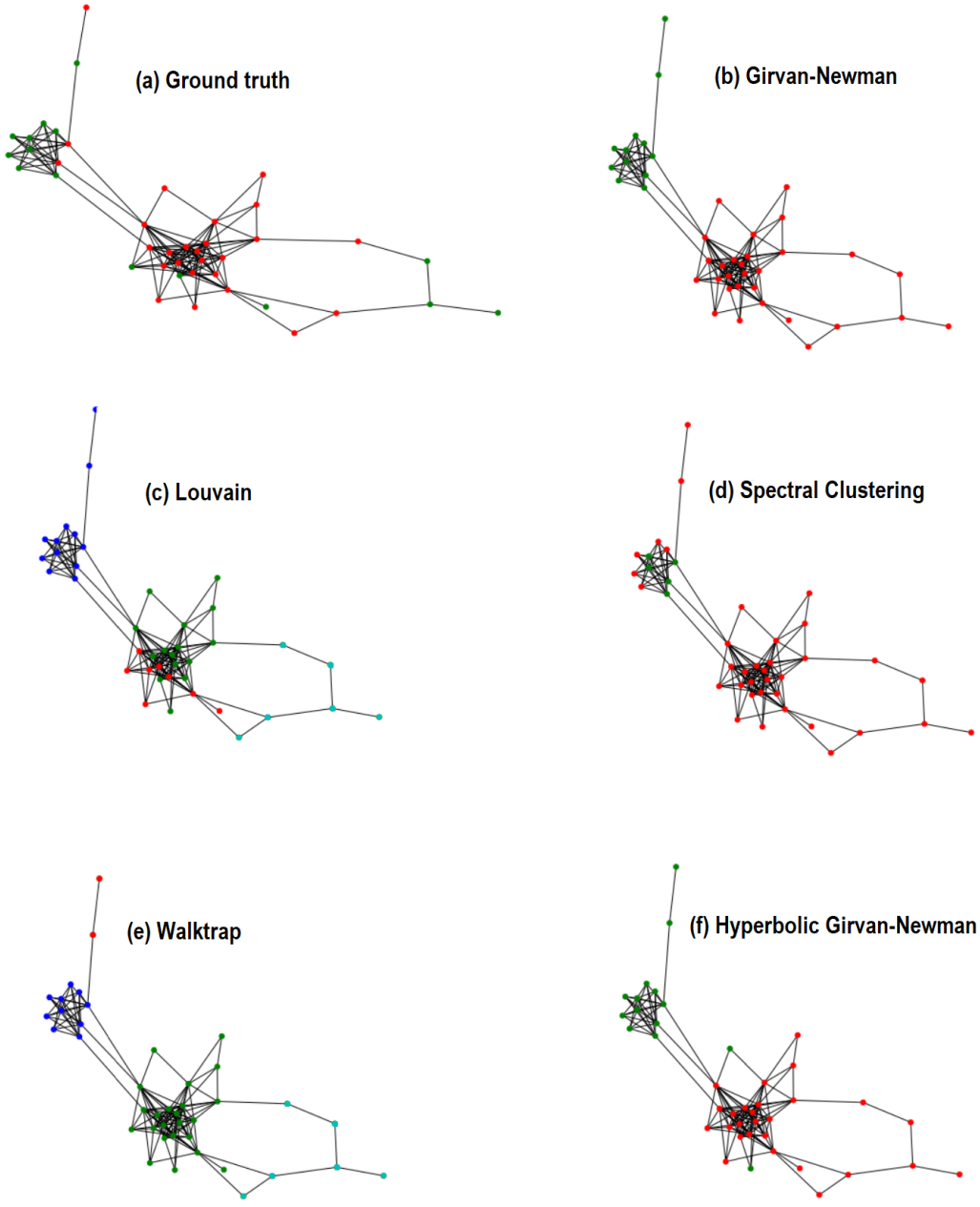

Figure 4.26. *Results of anomaly detection on the SCG of the mixed dataset (27 bot case)*

Chapter 5

Epilogue

In this final chapter we summarise the results and conclusions of this thesis and provide some ideas for future improvement.

5.1 Summary and conclusions

In this diploma thesis we implemented an existing community-based botnet detection method and experimented with different community detection techniques in it, including Hyperbolic Girvan-Newman.

Initially we provided the required theoretical background on complex networks, community detection, machine learning and botnet detection, as well as summaries of recent works on these topics. We then described in detail the botnet detection method this work was based on. Using statistical based anomaly detection, at first an empirical distribution of normal traffic is created and then used as a reference to detect small time windows of traffic that deviates from normal behavior. Highly interactive nodes, called pivotal nodes, are identified in the anomalous windows and a social correlation graph (SCG) is created based on how the interactions of regular nodes and pivotal nodes are correlated. Since pivotal nodes are either botmasters or target nodes, bot interaction with pivotal nodes is very likely to be correlated. Based on this idea, community detection is then applied on the SCG in order to detect bot communities.

We experimented with three different datasets, one of which was created by combining normal traffic with attacking traffic. For the anomaly detection part we confirmed that detecting anomalous windows is more difficult the fewer bots there are, especially at the beginning of the attack. For the SCG we saw that there is a trade-off based on how strict we set the threshold for its creation: stricter SCGs have fewer bots present in them, but the communities are more distinct, while less strict SCGs include more bots but it is more difficult to separate them from normal nodes.

We selected both easy and difficult SCGs to run five different community detection algorithms: Girvan-Newman, Louvain, Spectral Clustering, Walktrap and Hyperbolic-Girvan-Newman (HGN). HGN is a variant of Girvan-Newman that embeds the network into hyperbolic space for quicker calculations, with a slight loss of accuracy.

We evaluated the results of community detection by comparing them with the ground truth using NMI and F1-score metrics. From our experiments we saw that Louvain and

Walktrap algorithms, that do not specify the number of communities beforehand, tend to over-split the SCG into multiple communities and only identify part of the botnet. Spectral Clustering performed best on the first two datasets but struggled on the mixed one, whereas the Girvan-Newman algorithms performed decently on the first two datasets and had the best results on the mixed one. Overall the Girvan-Newman algorithms proved the most reliable as long as the SCG is correctly calibrated. In this case there are benefits of using HGN over Girvan-Newman for larger networks if speed is a priority, since its results are nearly identical.

5.2 Future work

At the end of this thesis we provide some ideas for further experimentation:

- Different way to distinguish whether a community is a bot community or not. The current algorithm picks the community with the highest average pivotal interaction measure as the bot community. This makes it so that community detection methods that detect more than two communities often only identify part of the botnet, since it has been split into multiple smaller communities. If for example a threshold-based method is used instead, or a method using different network characteristics, then perhaps more bots would correctly be detected by these algorithms. This would also allow Girvan-Newman-like algorithms to detect disconnected bots in the SCG, by having them detect more than two communities and then identifying more than one bot community.
- Tuning of τ, τ_ρ parameters to include more bots in the SCG without sacrificing accuracy during community detection.
- Testing on larger anomalous windows and larger SCGs, which would require more specific tuning of HGN parameters (landmarks, batch size).

Bibliography

[1] *Clustering Coefficient in Graph Theory*. https://www.geeksforgeeks.org/clustering-coefficient-graph-theory/, last accessed on 2023-07-05.

[2] *Memgraph's Guide for NetworkX library*. https://networkx.guide/algorithms/community-detection/girvan-newman/, last accessed on 2023-07-05.

[3] *Random Geometric Graph*. https://networkx.org/documentation/networkx-1.9/examples/drawing/random_geometric_graph.html, last accessed on 2023-07-05.

[4] *Hierarchical clustering, using it to invest*. https://quantdare.com/hierarchical-clustering/, last accessed on 2023-07-05.

[5] Vasileios Karyotis, Konstantinos Tsitseklis, Konstantinos Sotiropoulos και Symeon Papavassiliou. *Big Data Clustering via Community Detection and Hyperbolic Network Embedding in IoT Applications*. *Sensors*, 18(4), 2018.

[6] *Kitsune Network Attack Dataset*. https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune, last accessed on 2023-07-05.

[7] Vasileios Karyotis, Eleni Stai και Symeon Papavassiliou. *Evolutionary Dynamics of Complex Communications Networks*. 2013.

[8] M. E. J. Newman. *The Structure and Function of Complex Networks*. *SIAM Review*, 45(2):167–256, 2003.

[9] Lucianoda Fontoura Costa, Osvaldo N. Oliveira, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana και Luis Enrique Correa Rocha. *Analyzing and modeling real-world phenomena with complex networks: a survey of applications*. *Advances in Physics*, 60(3):329–412, 2011.

[10] Symeon Papadopoulos, Ioannis Kompatsiaris, Athena Vakali και Ploutarchos Spyridonos. *Community detection in Social Media*. *Data Mining and Knowledge Discovery*, 24:515–554, 2012.

[11] Fabio Gasparetti, Alessandro Micarelli και Giuseppe Sansonetti. *Community Detection and Recommender Systems*, σελίδες 1–14. 2017.

[12] Ashani Wickramasinghe και Saman Muthukumarana. *Social network analysis and community detection on spread of COVID-19*. *Model Assisted Statistics and Applications*, 16:37–52, 2021.

[13] Shanfeng Wang, Maoguo Gong, Wenfeng Liu και Yue Wu. *Preventing epidemic spreading in networks by community detection and memetic algorithm*. *Applied Soft Computing*, 89:106–118, 2020.

[14] Michelle Girvan και Mark E. J. Newman. *Community structure in social and biological networks*. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[15] Vincent D Blondel, Jean Loup Guillaume, Renaud Lambiotte και Etienne Lefebvre. *Fast unfolding of communities in large networks*. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[16] Pascal Pons και Matthieu Latapy. *Computing Communities in Large Networks Using Random Walks*. *Journal of Graph Algorithms and Applications*, 10:191–218, 2006.

[17] Ulrike Luxburg. *A Tutorial on Spectral Clustering*. *Statistics and Computing*, 17:395–416, 2004.

[18] Xiaohan Zhao, Alessandra Sala, Haitao Zheng και Ben Y. Zhao. *Efficient shortest paths on massive social graphs*. *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, σελίδες 77–86, 2011.

[19] Eleni Stai, Konstantinos Sotiropoulos, Vasileios Karyotis και Symeon Papavassiliou. *Hyperbolic Embedding for Efficient Computation of Path Centralities and Adaptive Routing in Large-Scale Complex Commodity Networks*. *IEEE Transactions on Network Science and Engineering*, 4(3):140–153, 2017.

[20] M. Tariq Banday, Jameel A. Qadri και Nisar A. Shah. *Study of Botnets and Their Threats to Internet Security*. 2009.

[21] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe και Kim Kwang Raymond Choo. *IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers*. *Forensic Science International: Digital Investigation*, 32:300926, 2020.

[22] *Internet Relay Chat Protocol*. RFC 1459, 1993.

[23] *GameOver Zeus P2P Malware*. https://www.cisa.gov/news-events/alerts/2014/06/02/gameover-zeus-p2p-malware, last accessed on 2023-07-05.

[24] *GLOBAL DDOS SUMMARY*. https://horizon.netscout.com/?atlas=summary, last accessed on 2023-07-05.

[25] *Botnet Detection*. https://jpdias.me/botnet-lab//countermeasures/detection.html, last accessed on 2023-07-05.

[26] Jing Wang και Ioannis Ch. Paschalidis. *Botnet Detection Based on Anomaly and Community Detection*. *IEEE Transactions on Control of Network Systems*, 4(2):392–404, 2017.

[27] E. N. Gilbert. *Random Graphs*. *Annals of Mathematical Statistics*, 30:1141–1144, 1959.

[28] P. Erdös και A. Rényi. *On Random Graphs I*. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[29] Jihyeok Choi και Sunder Sethuraman. *Large deviations for the degree structure in preferential attachment schemes*. *The Annals of Applied Probability*, 23(2), 2013.

[30] Fan Chung, Shirin Handjani και Doug Jungreis. *Generalizations of Polya's urn Problem*. *Annals of Combinatorics*, 7:141–153, 2003.

[31] SAND Lab @ UC Santa Barbara. *Rigel: Fast and Scalable Analysis of Massive Social Graphs*. https://sandlab.cs.uchicago.edu/current/rigel/, last accessed on 2023-07-05.

[32] S. García, M. Grill, J. Stiborek και A. Zunino. *An empirical comparison of botnet detection methods*. *Computers Security*, 45:100–123, 2014.

[33] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici και Asaf Shabtai. *Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection*. *The Network and Distributed System Security Symposium (NDSS) 2018*, 2018.

[34] *The CAIDA UCSD DDoS 2007 attack*. http://www.caida.org/data/passive/ddos-20070804_dataset.xml, last accessed on 2023-07-05.

[35] R.R.R. Barbosa, R. Sadre, Aiko Pras και R. van de Meent. *Simpleweb/University of Twente Traffic Traces Data Repository*. Αριθμός TR-CTIT-10-19 στο CTIT Technical Report Series. Centre for Telematics and Information Technology (CTIT), Netherlands, 2010.

[36] Andreas Reppas, Konstantinos Spiliotis και Constantinos Siettos. *On the effect of the path length of small-world networks on epidemic dynamics*. *Virulence*, 3:146–53, 2012.

[37] Andrea Landherr, Bettina Friedl και Julia Heidemann. *A Critical Review of Centrality Measures in Social Networks*. *Business Information Systems Engineering*, 2:371–385, 2010.

[38] Onur Ugurlu. *Comparative analysis of centrality measures for identifying critical nodes in complex networks*. *Journal of Computational Science*, 62:101738, 2022.

[39] Wang Mengyuan, Haiying Wang και Huiru Zheng. *A Mini Review of Node Centrality Metrics in Biological Networks*. *International Journal of Network Dynamics and Intelligence*, σελίδες 99–110, 2022.

[40] Jure Leskovec και Eric Horvitz. *Planetary-Scale Views on an Instant-Messaging Network*. *Proceeding of the 17th International Conference on World Wide Web 2008*, 2008.

[41] Christine Marshall, James Cruickshank και Colm O'Riordan. *Social Network Analysis of Clustering in Random Geometric Graphs*. *ACM WomEncourage 2014*, 2014.

[42] Duncan J. Watts και Steven H. Strogatz. *Collective dynamics of 'small-world' networks*. *Nature*, 393:440–442, 1998.

[43] Albert Laszlo Barabasi και Reka Albert. *Emergence of Scaling in Random Networks*. *Science*, 286:509–512, 1999.

[44] Ritik Dixit, Rishika Kushwah και Samay Pashine. *Handwritten Digit Recognition using Machine and Deep Learning Algorithms*. *International Journal of Computer Applications*, 176(42):27–33, 2020.

[45] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi και Opeyemi Emmanuel Ajibuwa. *Machine learning for email spam filtering: review, approaches and open research problems*. *Heliyon*, 5(6):e01802, 2019.

[46] Mark Pritt και Gary Chern. *Satellite Image Classification with Deep Learning*. *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, σελίδες 1–7, 2017.

[47] Hany Alashwal, Mohamed El Halaby, Jacob Crouse, Areeg Abdalla και Ahmed Moustafa. *The Application of Unsupervised Clustering Methods to Alzheimer's Disease*. *Frontiers in Computational Neuroscience*, 13, 2019.

[48] Philip D. Waggoner. *Unsupervised Machine Learning for Clustering in Political and Social Research*. Elements in Quantitative and Computational Methods for the Social Sciences. Cambridge University Press, 2021.

[49] Muhammad Faizan, Megat Farez Azril Zuhairi, Shahrinaz Ismail και Sara Sultan. *Applications of Clustering Techniques in Data Mining: A Comparative Study*. *International Journal of Advanced Computer Science and Applications*, 11, 2020.

[50] Neil Thompson, Kristjan Greenewald, Keeheon Lee και Gabriel F. Manso. *The Computational Limits of Deep Learning*. *Ninth Computing within Limits 2023*. LIMITS, 2023.

[51] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis και Diego Andina. *Deep Learning for Computer Vision: A Brief Review*. *Computational Intelligence and Neuroscience*, 2018, 2018.

[52] Daniel Otter, Julian Medina και Jugal Kalita. *A Survey of the Usages of Deep Learning for Natural Language Processing*. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21, 2020.

[53] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh και Khaled Shaalan. *Speech Recognition Using Deep Neural Networks: A Systematic Review*. *IEEE Access*, 7:19143–19165, 2019.

[54] *One-Class Classification Algorithms for Imbalanced Datasets*. https://machinelearningmastery.com/one-class-classification-algorithms/, last accessed on 2023-07-05.

[55] Alexander Ypma και Robert P. W. Duin. *Support objects for domain approximation*. *International Conference on Artificial Neural Networks (ICANN 98)*Lars Niklasson, Mikael Bodén και Tom Ziemke, επιμελητές, σελίδες 719–724, London, 1998. Springer London.

[56] Teuvo Kohonen. *The self-organizing map*. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[57] Weizhi Li, Weirong Mo, Xu Zhang, John Squiers, Yang Lu, Eric Sellke, Wensheng Fan, Michael DiMaio και Jeffrey Thatcher. *Outlier detection and removal improves accuracy of machine learning approach to multispectral burn diagnostic imaging*. *Journal of biomedical optics*, 20:121305, 2015.

[58] Ane Blázquez-García, Angel Conde, Usue Mori και Jose Lozano. *A Review on Outlier/Anomaly Detection in Time Series Data*. *ACM Computing Surveys*, 54:1–33, 2021.

[59] Md Siddiqui, Jack Stokes, Christian Seifert, Evan Argyle, Robert McCann, Joshua Neil και Justin Carroll. *Detecting Cyber Attacks Using Anomaly Detection with Explanations and Expert Feedback*. σελίδες 2872–2876, 2019.

[60] T. T. Teoh, Graeme Chiew, Edwin J Franco, P. C. Ng, M.P Benjamin και Y. J. Goh. *Anomaly detection in cyber security attacks on networks using MLP deep learning*. *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, σελίδες 1–5, 2018.

[61] Maximilian E. Tschuchnig και Michael Gadermayr. *Anomaly Detection in Medical Imaging - A Mini Review*. *Data Science - Analytics and Applications*, σελίδες 33–38. Springer Fachmedien Wiesbaden, 2022.

[62] Waleed Hilal, S. Andrew Gadsden και John Yawney. *Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances*. *Expert Systems with Applications*, 193:116429, 2022.

[63] Federico Pittino, Michael Puggl, Thomas Moldaschl και Christina Hirschl. *Automatic Anomaly Detection on In-Production Manufacturing Machines Using Statistical Learning Methods*. *Sensors*, 20(8), 2020.

[64] K. Ting, F. Liu και Z. Zhou. *Isolation Forest*. *ICDM 2008. Eighth IEEE International Conference on Data Mining*, σελίδες 413–422, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[65] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff και Puneet Agarwal. *Long Short Term Memory Networks for Anomaly Detection in Time Series*. 2015.

[66] Stuart P. Lloyd. *Least squares quantization in PCM*. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[67] Meena Mahajan, Prajakta Nimbhorkar και Kasturi Varadarajan. *The Planar k-Means Problem is NP-Hard*. *WALCOM: Algorithms and Computation*Sandip Das και Ryuhei Uehara, επιμελητές, σελίδες 274–285, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[68] M. Emre Celebi, Hassan A. Kingravi και Patricio A. Vela. *A comparative study of efficient initialization methods for the k-means clustering algorithm*. *Expert Systems with Applications*, 40(1):200–210, 2013.

[69] Vincenzo Moscato και Giancarlo Sperlì. *A survey about community detection over On-line Social and Heterogeneous Information Networks*. *Knowledge-Based Systems*, 224:107112, 2021.

[70] Aaron Clauset, M. E. J. Newman και Cristopher Moore. *Finding community structure in very large networks*. *Physical Review E*, 70(6), 2004.

[71] M. Ángeles Serrano και Marián Boguñá. *The Shortest Path to Network Geometry: A Practical Guide to Basic Models and Applications*. Elements in the Structure and Dynamics of Complex Networks. Cambridge University Press, 2022.

[72] Fragkiskos Papadopoulos, Constantinos Psomas και Dmitri V. Krioukov. *Replaying the Geometric Growth of Complex Networks and Application to the AS Internet*. *CoRR*, abs/1205.4384, 2012.

[73] Konstantinos Tsitseklis, Maria Krommyda, Vasileios Karyotis, Verena Kantere και Symeon Papavassiliou. *Scalable Community Detection for Complex Data Graphs via Hyperbolic Network Embedding and Graph Databases*. *IEEE Transactions on Network Science and Engineering*, 8(2):1269–1282, 2021.

[74] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng και Philip S. Yu. *A Comprehensive Survey on Community Detection With Deep Learning*. *IEEE Transactions on Neural Networks and Learning Systems*, σελίδες 1–21, 2022.

[75] Andrea Lancichinetti, Santo Fortunato και Filippo Radicchi. *Benchmark graphs for testing community detection algorithms*. *Physical Review E*, 78(4), 2008.

[76] Zhao Yang, René Algesheimer και Claudio J. Tessone. *A Comparative Analysis of Community Detection Algorithms on Artificial Networks*. *Scientific Reports*, 6(1), 2016.

[77] *Eggheads - Eggdrop Development*. https://www.eggheads.org, last accessed on 2023-07-05.

[78] Simon Heron. *Botnet command and control techniques*. *Network Security*, 2007:13–16, 2007.

[79] I. B. Mopari, S. G. Pukale και M. L. Dhore. *Detection and defense against DDoS attack with IP spoofing*. *2008 International Conference on Computing, Communication and Networking*, σελίδες 1–5, 2008.

[80] James R. Binkley και Suresh Singh. *An Algorithm for Anomaly-based Botnet Detection*. *2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI 06)*, San Jose, CA, 2006. USENIX Association.

[81] Sajjad Arshad, Maghsoud Abbaspour, Mehdi Kharrazi και Hooman Sanatkar. *An anomaly-based botnet detection approach for identifying stealthy botnets*. *2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, σελίδες 564–569, 2011.

[82] Zhou Mingqiang, Huang Hui και Wang Qian. *A graph-based clustering algorithm for anomaly intrusion detection*. *2012 7th International Conference on Computer Science Education (ICCSE)*, σελίδες 1311–1314, 2012.

[83] Ali Zand, Giovanni Vigna, Xifeng Yan και Christopher Kruegel. *Extracting probable command and control signatures for detecting botnets*. *Proceedings of the ACM Symposium on Applied Computing*, 2014.

[84] Paweł Szynkiewicz. *Signature-Based Detection of Botnet DDoS Attacks*, σελίδες 120–135. Springer International Publishing, Cham, 2022.

[85] Brent Byung Hoon Kang. *DNS-Based Botnet Detection*, σελίδες 362–363. Springer US, Boston, MA, 2011.

[86] Seungjin Lee, Azween Abdullah και Noor Jhanjhi. *A Review on Honeypot-based Botnet Detection Models for Smart Factory*. *International Journal of Advanced Computer Science and Applications*, 11, 2020.

[87] Ruchi Vishwakarma και Ankit Kumar Jain. *A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks*. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, σελίδες 1019–1024, 2019.

[88] Jun Zhao, Xudong Liu, Qiben Yan, Bo Li, Minglai Shao και Hao Peng. *Multi-Attributed Heterogeneous Graph Convolutional Network for Bot Detection*. *Information Sciences*, 537, 2020.

[89] Dali Zhu, Yuchen Ma και Yinlong Liu. *Anomaly Detection with Deep Graph Autoencoders on Attributed Networks*. *2020 IEEE Symposium on Computers and Communications (ISCC)*, σελίδες 1–6, 2020.

[90] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan και Xiuzhen Cheng. *Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT*. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2022.

[91] *Wireshark*. https://www.wireshark.org, last accessed on 2023-07-05.

[92] Aric A. Hagberg, Daniel A. Schult και Pieter J. Swart. *Exploring Network Structure, Dynamics, and Function using NetworkX*. *Proceedings of the 7th Python in Science Conference*Gaël Varoquaux, Travis Vaught και Jarrod Millman, επιμελητές, σελίδες 11 – 15, Pasadena, CA USA, 2008.

[93] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot και E. Duchesnay. *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[94] J. D. Hunter. *Matplotlib: A 2D graphics environment*. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[95] *pandas - Python Data Analysis Library*. https://pandas.pydata.org, last accessed on 2023-07-05.

[96] R. Cazabet G. Rossetti, L. Milli. *CDlib: a Python Library to Extract, Compare and Evaluate Communities from Complex Networks*. *Applied Network Science Journal*, 2019.

List of Abbreviations

ΓΚΣ	Γράφος Κοινωνικής Συσχέτισης
AI	Artificial Intelligence
CHJ	Chung-Handjani-Jungreis
DDoS	Distributed Denial-of-Service
DoS	Denial-of-Service
HEBC	Hyperbolic Edge-Betweenness Centrality
HGN	Hyperbolic Girvan-Newman
IRC	Internet Relay Chat
IoT	Internet of Things
KL	Kullback-Leibler
LFR	Lancichinetti–Fortunato–Radicchi
LSTM	Long Short-Term Memory
ML	Machine Learning
NMI	Normalised Mutual Information
P2P	Peer-to-Peer