NNT : 201XSACLSXXX

Thèse de doctorat de l'Université Paris-Saclay préparée á l' École Polytechnique

Ecole doctorale n°XXX
Sciences et technologies de l'information et de la communication
Spécialité de doctorat : Informatique

par

Mme. Maria Evgenia Rossi

"Graph Mining for Influence Maximization in Social Networks"

Thèse présentée et soutenue à "lieu", le XX Novembre 2017.

Composition du Jury :

M.	Michalis Vazirgiannis	Professeur École Polytechnique	(Directeur de thèse)
M.	Benjamin Nguyen	Professeur INSA Centre Val de Loire	(Co-Directeur de thèse)
M.	David Gross-Amblard	Professeur Université de Rennes 1	(Rapporteur)
M.	Vasileios Megalooikonomou	Professeur University of Patras	(Rapporteur)
M.	Cédric Eichler	Maître de conférences INSA Centre Val de Loire	(Examinateur)
M.	Yuxiao Dong	Applied Scientist Microsoft Research Redmond	(Examinateur)
Mme.	Ioana Manolescu	Professeure École Polytechnique	(Examinatrice)

Maria Evgenia Rossi: *Graph Mining for Influence Maximization in Social Networks* © 2017

ABSTRACT

Modern science of graphs has emerged the last few years as a field of interest and has been bringing significant advances to our knowledge about networks. Until recently the existing data mining algorithms were destined for structured/relational data while many datasets exist that require graph representation such as social networks, networks generated by textual data, 3D protein structures and chemical compounds. It has become therefore of crucial importance to be able to extract in an efficient and effective way meaningful information from that kind of data and towards this end graph mining and analysis methods have been proven essential.

The goal of this thesis is to study problems in the area of graph mining focusing especially on designing new algorithms and tools related to information spreading and specifically on how to locate influential entities in real-world social networks. This task is crucial in many applications such as information diffusion, epidemic control and viral marketing.

In the first part of the thesis, we have studied spreading processes in social networks focusing on finding topological characteristics that rank entities in the network based on their influential capabilities. We have specifically focused on the K-truss decomposition which is an extension of the k-core decomposition of the graph. Both methods partition a graph into subgraphs whose nodes and/or edges have some common characteristics. For the case of the K-truss, the edges belonging to such subgraph are contained to at least K-2 triangles. After extensive experimental analysis in real-world networks, we showed that the nodes that belong to the maximal K-truss subgraph show a better spreading behavior when compared to baseline criteria such as degree and k-core centralities. Such spreaders have the ability to influence a greater part of the network during the first steps of a spreading process but also the total fraction of the influenced nodes at the end of the epidemic is greater. We have also observed that node members of such dense subgraphs are those achieving the optimal spreading in the network.

In the second part of the thesis, we focused on identifying a group of nodes that by acting all together maximize the expected number of influenced nodes at the end of the spreading process, formally called Influence Maximization. The Influence Maximization problem is actually NP-hard though there exist approximation guarantees for efficient algorithms that can solve the problem while obtaining a solution within the 63% of optimal classes of models. As those guarantees propose a greedy approximation which is computationally expensive especially for large graphs, we proposed the MATI algorithm which succeeds in locating the group of users that maximize the influence while also being scalable. The algorithm takes advantage of the possible paths created in each node's neighborhood and precalculates each node's potential influence and achieves to produce competitive results in quality compared to those of baseline algorithms such as the Greedy, LDAG and SimPath.

In the last part of the thesis, we study the privacy point of view of sharing such metrics that are good influential indicators in a social network. We have focused on designing an algorithm that addresses the problem of computing through an efficient, correct, secure, and privacy-preserving algorithm the k-core metric which measures the influence of each node of the network. We have specifically adopted a decentralization approach where the social network is considered as a Peer-to-peer (P2P) system. The algorithm is built based on the constraint that it should not be possible for a node to reconstruct partially or entirely the graph using the information they obtain during its execution. While a distributed algorithm that computes once and for all the nodes' coreness is already proposed, networks that evolve over time are not taken into account. Our main contribution is an incremental algorithm that efficiently solves the core maintenance problem in P2P while limiting the number of messages exchanged and computations. Through extensive experiments we provide a security and privacy analysis of the solution regarding network deanonimization. We showed how it relates to previously defined attacks models and discuss countermeasures.

LIST OF PUBLICATIONS

The following publications and submissions under review are included in parts or in an extended version in this thesis:

- Maria-Evgenia G Rossi, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "Spread it good, spread it fast: Identification of influential nodes in social networks." In: *Proceedings of the 24th International Conference on World Wide Web*. ACM. 2015, pp. 101–102.

- Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. "Locating influential nodes in complex networks." In: *Scientific reports* 6 (2016).

- Maria-Evgenia G Rossi and Michalis Vazirgiannis. "Exploring Network Centralities in Spreading Processes." In: *International Symposium on Web AlGorithms (iSWAG)*. 2016.

- Konstantinos Skianis, Maria-Evgenia G Rossi, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "SpreadViz: Analytics and Visualization of Spreading Processes in Social Networks." In: *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1324–1327.

- Maria-Evgenia G Rossi, Cédric Eichler, Pascal Berthomé, and Benjamin Nguyen. "Private, Secure and Distributed Computation of k-cores." In: *Manuscript, presented in APVP*. 2017.

- Maria-Evgenia G Rossi, Bowen Shi, Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Christos Giatsidis, and Michalis Vazirgiannis. "MATI: An Efficient Algorithm for Influence Maximization in Social Networks." In: *Manuscript*. 2017.

CONTENTS

1 INTRODUCTION 1
- 1.1 Social Networks and Social Influence 1
- 1.2 Social Influence examples 2
- 1.3 Social Influence Analysis Applications 3
- 1.4 Thesis statement and overview of contributions 4
  - 1.4.1 Identification of individual influential spreaders 5
  - 1.4.2 Identification of a group of influential spreaders 5
  - 1.4.3 Secure and Private computation of influential metrics 6
- 1.5 Outline of the thesis 6

2 BASIC CONCEPT AND PRELIMINARIES 8
- 2.1 Introduction to Graph Theory 8
- 2.2 Adjacency Matrix and Eigenvalues 11
- 2.3 Node centralities 11
  - 2.3.1 Structural centralities 12
  - 2.3.2 Iterative refinement centralities 15
- 2.4 Description of Graph Datasets 16

3 LOCATING INFLUENTIAL SPREADERS IN SOCIAL NETWORKS 20
- 3.1 Introduction 20
- 3.2 Preliminaries and Background 22
  - 3.2.1 k-core decomposition 22
  - 3.2.2 K-truss decomposition 23
  - 3.2.3 Epidemic models 26
  - 3.2.4 The SIR model applied in networks 28
- 3.3 Related work 28
  - 3.3.1 Identification of Individual Influential Spreaders in Networks 28
- 3.4 K-truss decomposition for identifying influential nodes 29
  - 3.4.1 Evaluating the spreading performance 32
  - 3.4.2 Comparison to the optimal spreading 37
  - 3.4.3 Impact of infection and recovery rate on the spreading process 43
- 3.5 Exploration of network centralities in spreading processes 45
  - 3.5.1 Evaluation of Results 47
- 3.6 Conclusions and Future Work 51

4 INFLUENCE MAXIMIZATION IN SOCIAL NETWORKS 54
- 4.1 Introduction 54

4.2 Preliminaries and Background 56
4.2.1 The Influence Maximization (IM) problem 56
4.2.2 Diffusion Models 58
4.3 Related work 59
4.4 MATrix Influence (MATI) Algorithm 61
4.4.1 Influence in Social Networks 61
4.4.2 Influence Computation under the LT Model 63
4.4.3 Influence Computation under the IC Model 67
4.5 Experimental Evaluation 69
4.5.1 Datasets 69
4.5.2 Baseline Algorithms 69
4.5.3 Experimental Results 70
4.6 Conclusions and Future Work 73
5 PRIVATE, SECURE AND DISTRIBUTED COMPUTATION OF K-CORES 76
5.1 Introduction 76
5.2 Problem Statement and Preliminiaries 78
5.2.1 Problem Statement 78
5.2.2 Preliminaries and Background 79
5.3 Related work 81
5.3.1 k-core Computation 81
5.3.2 Core Maintenance 82
5.3.3 Decentralized Personal Data Management Platforms 82
5.4 P2P Algorithm for Core Maintenance 83
5.4.1 Local variables 84
5.4.2 Handling Messages and Events 84
5.4.3 Computing coreness estimations 87
5.4.4 Example 90
5.5 Analytical and Experimental Study 93
5.5.1 Analytical study 93
5.5.2 Complexity: Experimental Study 104
5.6 Security and Privacy Analysis 105
5.6.1 Attack Model 109
5.6.2 Privacy and Information Quality 110
5.6.3 Experimental results 111
5.7 Conclusions and Remarks 117
6 CONCLUDING REMARKS 119
6.1 Summary of Contributions and Future Work 119
6.2 Epilogue 122

BIBLIOGRAPHY 123

LIST OF FIGURES

Figure 2.1 Examples of an undirected and a directed graph 9
Figure 2.2 Examples of a connected and a disconnected graph 10
Figure 2.3 Examples of a cyclic and an acyclic graph 10
Figure 2.1 Example of the k-core decomposition 13
Figure 3.1 Schematic representation of the maximal k-core and K-truss subgraphs of a graph. 25
Figure 3.2 State diagram of the SIR model 26
Figure 3.3 State diagram of the SIS model 27
Figure 3.1 Complementary cumulative truss number distribution function 31
Figure 3.2 Cumulative difference of the infected nodes per step achieved by the truss method vs. the core and top degree methods (Continued in Fig. 3.3) 38
Figure 3.3 Cumulative difference of the infected nodes per step achieved by the truss method vs. the core and top degree methods 39
Figure 3.4 Spreading distribution of the nodes in the network 40
Figure 3.5 Distribution of the top-truss $P_W^{\mathcal{T}}$ and top-core $P_W^{\mathcal{C}}$ nodes among the nodes with optimal spreading properties under a window of size W 41
Figure 3.6 Distribution of node's truss number with respect to the ranking of the nodes under their spreading properties 42
Figure 3.7 Impact of infection and recovery probabilities of the SIR model on the spreading process 44
Figure 3.1 Complementary cumulative distribution function of nodes' (a) degree, (b) core number and (c) truss number of the EPINIONS dataset. 47
Figure 3.2 Evolution of the infected nodes' average (a) degree d_v, (b) core number and (c) truss number during a simulated spreading process for the EPINIONS dataset. 48

Figure 3.3 Comparison of the evolution of the infected nodes' average (a) degree, (b) core number and (c) truss number, between a simulated spreading process and real influence data for the HIGGS-TWITTER dataset. 50
Figure 4.1 Illustration of the Linear Threshold model. 58
Figure 4.1 Example graph. 62
Figure 4.2 Illustration of Theorem 1. 65
Figure 4.1 Influence spread in number of nodes for the different algorithms, under the LT model. 71
Figure 4.2 Influence spread in number of nodes for the different algorithms, under the IC model. 72
Figure 4.3 Comparison of running times in seconds of the different algorithms under the (a) LT and (b) IC models. 74
Figure 5.1 A simple example describing the static part of the algorithm. 91
Figure 5.2 A simple example describing the dynamic part of the algorithm. 94
Figure 5.1 Example graph 96
Figure 5.1 Toy example to demonstrate the computation of q_1, q_2 and q_N. 114
Figure 5.2 Distribution of nodes in the top k-core subgraphs in the original graph formats. 115
Figure 5.3 Normalized Quality q_N of different percentages of top spreaders selected from anonymized versions of real datasets. 116

LIST OF TABLES

Table 2.1 Networks datasets used in the thesis 19
Table 3.1 List of symbols and their definitions 23
Table 3.1 Properties of the real-world graphs used in the study 32
Table 3.2 Average number of infected nodes per step of the SIR model 35
Table 3.3 Cumulative number of infected nodes per step of the SIR model 36
Table 3.1 Network datasets used in Section 3.5. 45
Table 4.1 List of symbols used in Chapter 4. 56
Table 4.1 Properties of the real-world graphs used in Chapter 4. 70
Table 4.2 Comparison of running times in seconds and influence spread in number of nodes for different values of the parameter θ. 73
Table 5.1 List of symbols and their definitions 79
Table 5.1 Properties of the real-world graphs used in Chapter 5. 104
Table 5.2 ICS and PIICS statistics for NetHept network 105
Table 5.3 ICS and PIICS statistics for the EmailEnron network 106
Table 5.4 ICS and PIICS statistics for the WikiVote network 107
Table 5.5 ICS and PIICS statistics for the Epinions network 108

1 INTRODUCTION

NETWORKS haven been getting a lot of attention the recent years. While various kind of data can be naturally mapped to graph structures [132, 133], the scientific study of networks has greatly benefited from a broad range of ideas brought by specialists from different disciplines. *Information networks*, such as Web pages linked together by hyperlinks [85], virtual networks of computers that allow sharing of files between computer users over local- or wide-area networks (i.e., Peer-to-peer networks) [87] or even citation networks between academic papers [59], have become crucial towards information dissemination and detection of social patterns in the academic world respectively. *Biological networks*, such as the network of metabolic pathways [156], the networks of physical interactions between proteins [88] or genetic regulatory networks [79] can be used in order to better understand the phenomena that occur in nature. Of great importance are also the *technological networks*, which include man-made networks designed that distribute resources such as electricity or information: electric power grids [162], the network of airline routes [5], roads [91] and of course the Internet [62]. Last but not least come the *social networks* which represent a set of people with some pattern of contacts (i.e., friendships, business relationships etc.) between them.

1.1 SOCIAL NETWORKS AND SOCIAL INFLUENCE

Social scientists have been meticulously studying social networks for decades [11, 161]. At first, studies concerned "small-world" cases such as: friendships within small groups [126], studies of business communities [66, 67], patterns of sexual contacts [18, 90] etc. The latest years, the Internet and the online social networking sites (e.g., Facebook, Twitter, LinkedIn, Tumblr etc.) have caused a remarkable growth of research on social networks. This led to the development of many applications of social networks of which a rich body of studies has been classified as *the analysis of influence or information diffusion in social networks*.

1.2 SOCIAL INFLUENCE EXAMPLES

There exist various real-life phenomena that motivate the study of information propagation in social networks.

Let us consider a social network as Facebook, where a user Anna posts about her having dinner at a specific restaurant in town. Such information is normally available to Anna's friends. If Anna's friends react on Anna's post, then their friends have access to this information too. In this way, the information contained in Anna's post is diffused through the network.

A famous case that dates back to the 1990s is the Hotmail phenomenon [86]. Back then Hotmail wasn't a well-known e-mail service provider. The simple idea of attaching at the end of each mail message a text that invited users to join the MSN Hotmail network, had the effect of boosting the brand in just 8 months. The recipients of such messages were inspired by the appended message to try it themselves and triggered other users in their turn by sending them the same mail to behave similarly. The phenomenon diffused very soon and Hotmail acquired 8 million users.

In a famous study published in the *New England Journal of Medicine*, Christakis and Fowler [44] analyzed a network of around 12 thousand people over a period of 32 years. The focus of this study was on the smoking behavior of people and its association with his/her social contacts. Their findings suggest that decisions to quit smoking are not made by individual persons, but reflect choices made by communities that are strongly connected to each other. Similar results come from another study of the aforementioned researchers [43] about the social influence of health conditions such as obesity. They found that having an obese friend icreases by 171% the possibility for an individual to be obese when compared to a random person.

Examples of social influence exist also in people's choice on entertainment such as music. "*See you again*", Whiz Khalifa's music video which serves as a tribute to a beloved actor who found tragic death in a car accident, reached a bit over 3 billion views on YouTube almost two year after its release (i.e., April of 2015). It has overtaken "*Gangnam Style*" which was the first video to reach 1 billion views as of December 21, 2012. In August 2017, only seven months after its release, "*Despacito*" reached 3,2 billion views much quicker than the 26 months it took Wiz Khalifa's song to overtake Psy's "*Gangnam Style*."

The power of diffusion has been various times utilized by people in various kinds of disasters. During the Paris terror attack on the 13th of November in 2015, as the events were taking place, millions of "tweets" were posted on Twitter. An analysis on the respective "tweets"* showed that only the night of the attack, 1,07 million related posts were published whereas until the 16th of November, that the situation still concurred the world news, a total of 18,17 million "tweets" contained relevant information. In the summer of 2011, in Vancouver, Canada, rioters that followed the Stanley Cup final destroyed public properties in the center of the city†. This triggered broad reactions of disgust which resulted in vast amounds of footage data: 5000 h worth of 100 types of digital video available for forensic analysis. The data helped the police to analyze the riot behavior.

1.3 SOCIAL INFLUENCE ANALYSIS APPLICATIONS

Influence propagation studies has found applications in various fields. From studying human, animal or even plant epidemics [81, 96, 139] to viral marketing [104], social media analytics [166], spread of rumors [127], expert finding [9], recommendation systems [84, 111, 158] etc.

A key task in order to understand information and influence diffusion is the identification of vital nodes that play a significant role in such cases. Such nodes may allow us to control the spread of an epidemy [49, 135], to predict successful scientists and scientific publications based on co-authorship and citation networks [55, 140, 170], design influential advertisements for new products [104, 115] etc.

For example, in the case of virus propagation, such as influenza, the transmission of the disease mainly depends on the extend of contacts of the infected person to the susceptible population; thus, being able to locate and vaccinate individuals with good spreading properties can prevent from a potential outbreak of the disease, leading to efficient strategies of epidemic control. In a similar way, suppose that our goal is to promote an idea or a product in order to be adopted by a large fraction of individuals in the network. A key idea behind viral marketing is the word-of-mouth effect [159]; individuals that have already adopted the product, recommend it to their friends who in turn do the same to their own social circle, forming a cascade of recommendations [57]. The basic question here is how to target a few initial individuals (e.g., by giving them free samples of the product or explaining them the idea), that can maximize the spread of influence in the network, leading to a successful promotion campaign.

* http://tipsandviz.blogspot.fr/2015/11/parisattacks-how-twitter-tells-story.html
† https://en.wikipedia.org/wiki/2011_Vancouver_Stanley_Cup_riot

Nevertheless, locating such users in a network is not a trivial task and numerous research has been conducted to solve the problem in the area [116]. It has been of significant importance to identify such nodes that will maximize the influence and information diffusion at the end of a respective phenomenon in a network. The problem is actually split into two subtopics: i) Identification of *individual influential nodes* that have good spreading properties and ii) Identification of *a group of nodes* that by acting all together will maximize the total spread of influence in a network. Indeed the two tasks greatly differ as finding a ranking of the nodes that by acting individually can influence a great part of the network cannot be directly used in order to discover the set of nodes that will – by acting at the same time – maximize the spreading of information in a graph. Of course this is justified by the fact that putting some of the most influential spreaders together will not result in a most influential set of such spreaders, because their respective influences may be and is usually largely overlapped.

The above are the topics that triggered the discoveries of this dissertation which is involved in graph mining techniques towards studying and analyzing social influence and specifically the identification of the nodes that play a key role in influence propagation. To that end, we propose models and algorithmic tools to address the challenging problems which arise in this area.

1.4 THESIS STATEMENT AND OVERVIEW OF CONTRIBUTIONS

This thesis contributes algorithms, tools, models and new insights to problems that arise in the area of graph mining for influence maximization in social networks. We specifically:

- Develop tools for analyzing the spreading behavior of individual nodes in complex networks. Special focus is given on ways that can efficiently rank the users based on their influential capabilities.

- Design algorithms that can locate a privileged group of nodes that – by acting all together – can maximize the spread of influence in a network at the end of a diffusion phenomenon.

- Develop models that can calculate metrics which measure the influence of an individual in a network in a secure and private way.

Next, we provide an overview of the contributions of the dissertation with respect to the above points.

1.4.1 *Identification of individual influential spreaders*

What characterizes the nodes that rank high in terms of their spreading performance?

Locating the users that can efficiently spread information through the network is not a trivial task. It is an even more challenging task when no personal information is provided for the users (i.e., age, sex, occupation, city of residence etc.). Usually the network is presented as a set of connections that represent some type of relationship between two users which are labelled with numerical ids. A straightforward metric that someone might think that affects users' spreading capabilities is the number of connections that they have. Even though this can be true for some cases, it has been showed that the metrics that are more efficient are those that can locate nodes that are well connected in the network. The k-core decomposition is one of such metrics and has also been proven to work well towards our primary goal.

In this part of the thesis (Chapter 3) we capitilize on the properties of the K-truss decomposition, a triangle-based extension of the k-core decomposition towards locating influential nodes. By simulating a spreading phenomenon on real networks triggered by the nodes identified by our method we prove that the latter show better spreading behavior compared to previously used importance criteria, leading to faster and wider epidemic spreading. We additionally present an extensive analysis of the nodes that conquer the top places in the optimal spreading ranking. We can conclude that, the K-truss decomposition can reveal nodes that tend to have good spreading properties – with the specific metric being highly related to the spreading effect.

Finally, we investigate the topological characteristics of individuals that are influenced and that participate in a diffusion process and present the patterns that are detected. We provide a comparison of the individuals' characteristics between a simulated and a real world spreading process and show the need for a more comprehensive model in this area.

1.4.2 *Identification of a group of influential spreaders*

How can someone locate a group of nodes that can maximize the total influence in the network?

As mentioned earlier, the current task greatly differs than the one described in the previous subsection. The nodes that are discovered using such methods cannot be directly used in order to discover the set of nodes that – by acting at the same time – can maximize the total influence in the network. That is justified by the fact that the influence of one can overlap with the influence of another top spreader.

The problem of *Influence Maximization* – as it is usually called – constitutes an NP-hard problem. A simple greedy algorithm has been proved to provide good approximation guarantees. Nevertheless, there are obviously serious scalability concerns – the greedy algorithm cannot provide results as soon as needed for large-scale networks.

We are proposing an efficient algorithm that can be used under the most famous diffusion models in the field: the *Linear Threshold* and *Independent Cascade* models. Our algorithm takes into consideration the possible paths that are created in each node's neighborhood and pre-calculates the nodes' influences. By performing extensive experiments in real datasets, we have shown that it is competitive regarding both the quality of seed nodes and the running time when compared to state-of-the-art algorithms (Chapter 4).

1.4.3 *Secure and Private computation of influential metrics*

How can we calculate in a secure and privacy preserving way an influential indicator?

Identifying the influential entities in a social network requires sharing its structure to the entity concerned. However, distribution of such information raises serious privacy concerns. In order to securely and privately compute influential nodes in a network, we propose a distributed Peer-to-peer algorithm. We constrain our algorithm on the basis that it should not be feasible for an attacker to reconstruct partially or entirely the graph using the information that can be obtained during its execution.

Specifically, our distributed algorithm computes the k-core number of each node which has been proved to efficiently rank the nodes in a network based on their spreading capabilities. Our main contribution is an incremental algorithm that succeeds in computing such a metric for a network that evolves over time. We show via experiments in real datasets that our solution fulfills the security and privacy requirements against typical attack models.

1.5 OUTLINE OF THE THESIS

The rest of the dissertation is organized as follows. In Chapter 2 we present basic concepts and background material that will be used throughout the dissertation. The next two Chapters are devoted to our work concerning identification of influential spreaders; in particular, in Chapter 3 the methods towards identification of individual spreaders is presented while in Chapter 4 we present our algorithm that efficiently locates a group of privileged users in social networks. In Chapter 5 we present our work towards a secure and private computation of k-cores. Finally, in Chapter 6, we offer concluding remarks about the topics covered in the dissertation and future research directions.

2 BASIC CONCEPT AND PRELIMINARIES

In this Chapter we provide the basic concepts and background theory that will be used throughout the thesis. Initially a basic introduction to graph theory is given along with the definitions of the node centralities that will be mentioned throughout the Chapters. The k-core decomposition is given special attention as it is a basic concept of the thesis, specifically for Chapters 3 and 5. Finally, the graph datasets that have been used for the experiments in the Chapters to follow are described. In each Chapter, the necessary background is presented in the respective section along with the symbols that will be used.

2.1 INTRODUCTION TO GRAPH THEORY

A network is represented as a graph (the terms graph and network are used interchangeably throughout the dissertation). A graph is a pictorial representation of a set of objects some of the pairs of which are connected with links. Formally a graph *G* is a pair of sets *(V,E)* where *V* is the set of vertices and *E* is the set of edges connecting the pairs of vertices. The number of nodes in the graph is equal to $n = |V|$ and the number of edges $m = |E|$.

There are different types of graphs depending upon the number of vertices and edges, the interconnectivity and their overall structure. Some of those are defined below. Figures 2.1 to 2.3 depict some examples of different types of graphs.

Definition 2.1. *Null Graph*
A null graph $G = (V, E)$ *contains* n *isolated nodes and no edges among them. They are also called endless graphs.*

Definition 2.2. *Directed and Undirected Graph*

- *In a directed graph* $G_D = (V, E)$, *every edge* $(u, v) \in E$ *links node* u *to node* v *(ordered pair of nodes).*
- *An undirected graph* $G = (V, E)$ *is a directed one where if edge* $(u, v) \in E$, *then edge* $(v, u) \in E$ *as well.*

Definition 2.3. *Weighted Graph*
Every edge $(u, v) \in E$ *in a weighted graph* $G = (V, E)$ *is associated with a real number* w_{uv} *called its* weight.

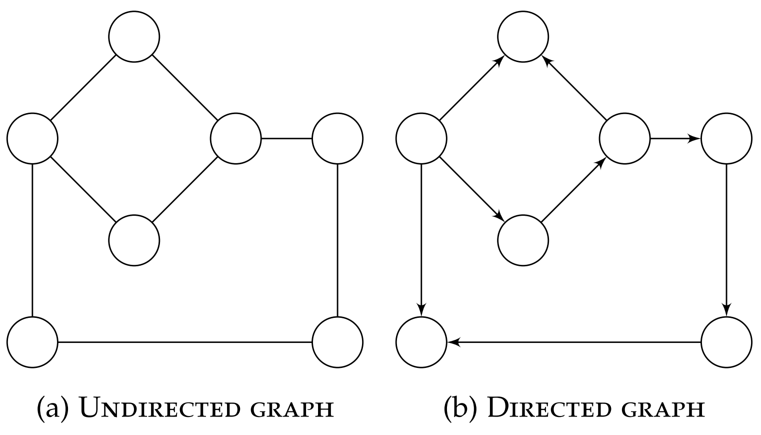


Figure 2.1: Examples of (a) an undirected and (b) a directed graph. In the case of the directed graph (b), the arrows indicate the directionality of each edge.

PATH A path is defined as a sequence of nodes $v_1, v_2, ..., v_{N-1}$, v_N. Every consecutive pair of nodes in the path v_k, v_{k+1} is connected with an edge. For a directed graph the notion of the path is extended as follows: in a *directed path* a directed edge should exist from each node of the sequence to the next node. Two nodes $u, v \in V$ are called *connected* if there is a path in the graph from node u to node v.

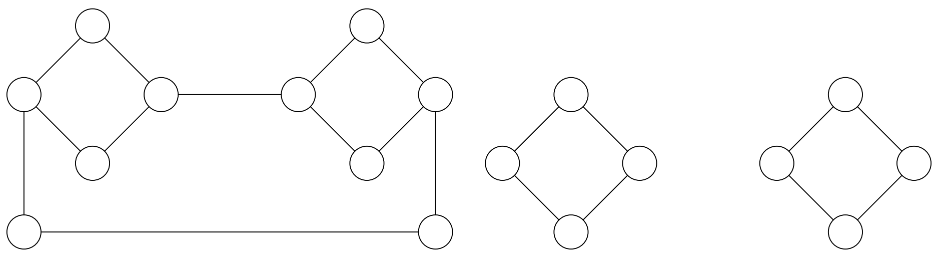
Definition 2.4. *Connected and Disconnected Graph*

- *In a connected graph* $G = (V, E)$ *there exists a path between every pair of vertices. There should be at least one edge for every vertex* u *in the graph so that it is connected to some other vertex* v *at the other side of the edge.*
- *A graph* $G = (V, E)$ *is disconnected if there exists at least a node* u *which is not connected to another node* v*.*

Definition 2.5. *Cyclic and Acyclic Graph*

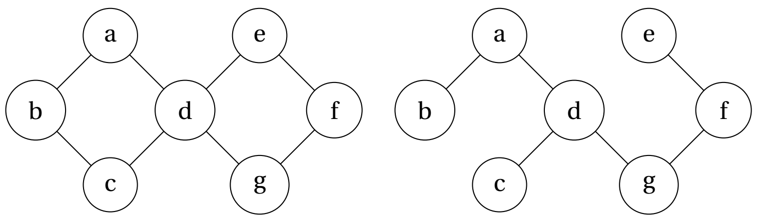
- *A cyclic graph* $G = (V, E)$ *is a graph that contains at least one cycle: the starting vertex of its first edge equals the ending vertex of its last edge.*
- *An acyclic graph* $G = (V, E)$ *is a graph that contains no cycles.*

DEGREE In an undirected graph $G = (V, E)$, a node $v \in V$ has a degree $d_G(v) = d$ if it has d incident edges. For the case of an undirected graph, every node is characterized by two types of degrees: its *in-degree* and its *out-degree*. The *in-degree* of node v equals to the number of incoming edges $d_G^{in}(v) = |u|(u, v) \in E|$



(a) Connected graph (b) Disconnected graph

Figure 2.2: Examples of (a) a connected and (b) a disconnected graph. In the connected graph we observe that each vertex has its edge connected to another edge whereas in the disconnected graph there exist two components which are not connected to each other.



(a) Cyclic graph (b) Acyclic graph

Figure 2.3: Examples of (a) a cyclic and (b) an acyclic graph. In the cyclic graph, we have two cycles a-b-c-d-a and d-e-f-g-d whereas in the acyclic we observe no cycles.

whereas its *out-degree* equals to the number of outcoming edges $d_G^{out}(v) = |u|(v,u) \in E|$. In undirected graphs $d_G^{in}(v) = d_G^{out}(v)$.

Definition 2.6. *Simple, Regular and Complete Graph*

- *A graph* $G = (V, E)$ *is a simple graph when it does not contain any loops (i.e., there exist no edges connecting a vertex to itself) or any parallel edges (i.e., edges that are incident to the same two vertices). The maximum number of edges possible in a single graph with* n *vertices are equal to* ${}^nC_2 = n(n-1)/2$*. The number of simple graphs possible with* n *nodes are* $2^{{}^nC_2} = 2^{n(n-1)/2}$.

- *The vertices of a regular graph* $G = (V, E)$ *have the same degree. If* $\forall v \in V : d_G(v) = k$ *then the graph is called a* k*-regular graph.*

- *Every pair of nodes in a complete graph* $G = (V, E)$ *is connected by a unique edge. A complete graph with* n *vertices has* $m = \binom{n}{2} = n(n-1)/2$ *edges.*

For a complete introduction to the field of complex networks, the reader may refer to Refs. [20, 25, 36, 132].

2.2 ADJACENCY MATRIX AND EIGENVALUES

Every type of graph can be represented as a matrix. This matrix is called the *adjacency matrix* **A** of the graph. Matrix **A** is a square matrix of size $n \times n$ where the rows and columns represent the nodes of the graph and the entries indicate whether there exists an edge between the respective pair of nodes.

Definition 2.7. *Adjacency Matrix [19]*

The adjacency matrix **A** *of a graph* $G = (V, E)$ *is a* $n \times n$ *matrix such that:*

$$A_{uv} = \begin{cases} a_{uv}, & \textit{if}\,(u,v) \in E, \;\; \forall\, u,v \in 1,\ldots,n \\ 0, & \textit{otherwise.} \end{cases} \tag{2.1}$$

For weighted graphs, each value a_{uv} represents the weight associated with each edge (u,v) while for unweighted graphs $a_{uv} = 1, \forall (u,v) \in E$. For a simple graph with no self-loops, the adjacency matrix must have zeros on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

Let **A** be a real $n \times n$ matrix. An *eigenvector* of **A** is a vector such that **A**x is parallel to x. In other words **A**$x = \lambda x$ where λ is a scalar called *eigenavlue* of **A** belonging to *eigenvector* v. λ is an eigenvalue iff the matrix $\mathbf{A} - \lambda I$ is singular, equivalently, iff $\det(\mathbf{A} - \lambda I) = 0$.

If matrix **A** is symmetric then all eigenvalues are real numbers. Additionally there is an orthogonal basis of the space $v_1, v_2, ..., v_n$ consisting of eigenvectors of **A** and the corresponding eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ are precisely the roots of $\det(\mathbf{A} - \lambda I) = 0$. Assuming that $|v_1| = ... = |v_n| = 1$, then the adjacency matric can be written as:

$$\mathbf{A} = \sum_{i+1}^{n} \lambda_i v_i v_i^T. \tag{2.2}$$

2.3 NODE CENTRALITIES

The concept of *centrality* is being used in order to define a node's importance according to its involvement in the network [21, 26, 27, 64]. In general centrality measures assign values to the nodes of a graph which can be used to rank the latter subject to their importance in the network. The different centralities that will be presented can be split in two subcategories:

i) *Structural centralities* which can be obtained exclusively on structural information.

ii) *Iterative refinement centralities* which use dynamical processes and iterative refinement methods to explore the nodes' structural properties.

2.3.1 *Structural centralities*

Structural centralities can be further classified into *neighborhood-based* and *path-based* centralities.

2.3.1.1 *Neighborhood-based centralities*

Degree centrality

As described earlier in the Chapter, the degree $d_G(v)$ of a node v of an undirected network $G = (V, E)$ represents the number of incindent edges on this node. It can also be calculated as follows: $d_G(v) = \sum_u a_{uv}$ where $\mathbf{A} = \{A_{uv}\}$ is the adjacency matrix of the graph and $u \in \mathrm{Neighbors}(v)$.

Degree centrality is used for a wide range of applications due to its simplicity and low-computational complexity. In some cases, degree acts exceptionally good. When the rate of the spreading of information is very low, degree can identify better the influential capabilities of nodes than some other centralities which on average are better influential indicators [99, 112]. Additionally, concerning network vulnerability, degree-targeted attack can destroy scale-free networks and exponential networks very effectively [89].

As mentioned before, nodes in directed networks are characterized by two types of degrees the *in-degree* and the *out-degree* based on whether the incoming or outcoming edges of the node are taken into account respectively. In weighted graphs degree is usually replaced by the strength which is defined as the sum of weights of the node's associated edges.

k-core centrality

The k-core centrality or node's core number or *coreness* is a number assigned to each node after the k-core decomposition of the network. The k-core decomposition is a hierarchical decomposition of the graph into nested subgraphs. Let $G = (V, E)$ be an undirected graph with $k \in \mathbb{Z}$ and $k > 0$.

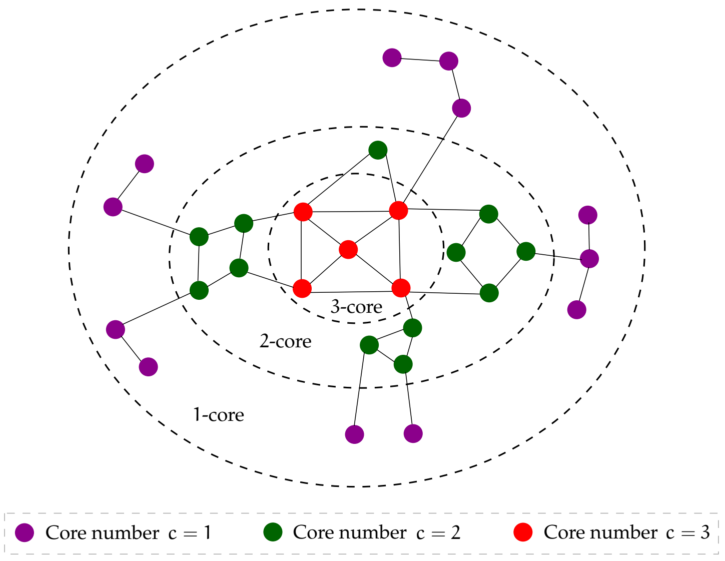


Figure 2.1: Example of the k-core decomposition.

Definition 2.8. *k-core subgraph*
Let H be a subgraph of G, (i.e., $H \subseteq G$). Subgraph H is defined to be a k-core subgraph of G, denoted by C_k, if it is a maximal connected subgraph in which all nodes have degree at least k.

Definition 2.9. *Node's Core Number*
A node i has core number $c_i = k$, if it belongs to a k-core but not to any $(k+1)$-core.

It is evident that if all the nodes of the graph have degree at least one, i.e., $d(v) \geqslant 1, \forall v \in V$, then the 1-core subgraph corresponds to the whole graph, i.e., $C_1 \equiv G$. Furthermore, assuming that $C_i,\ i = 0, 1, 2, \ldots, k_{max}$ is the i-core of G, then the k-core subgraphs are nested, i.e.:

$$C_0 \supseteq C_1 \supseteq C_2 \supseteq \ldots \supseteq C_{k_{max}}. \tag{2.3}$$

Typically, subgraph $C_{k_{max}}$ is called maximal k-core subgraph of G. Figure 2.1 depicts an example of a graph and the corresponding k-core decomposition.

Computing the k-core decomposition of a graph can be done through a simple process that is based on the following property: to extract the k-core subgraph, all nodes with degree less than k and their adjacent edges should be recursively deleted [151]. That way, beginning with $k = 0$, the algorithm removes all the nodes (and the incident edges) with degree equal or less than k, until no such nodes have been remained in the graph. Also notice that, removing edges that are incident to a node may cause reductions to the degree of neighboring nodes; the degree of some nodes may become at most k, and thus, they should also be removed at this step of the algorithm. When all remaining nodes have degree $d(v) > k$, k is increased by

Algorithm 2.1 k-core decomposition

```
1:  Input: Undirected graph $G = (V, E)$
2:  Output: Vector of core numbers $c_i$, $i = 1, 2, ...n = |V|$
3:  Compute the degrees of each node $d_G(v), \forall v \in V$
4:  Order the nodes in increasing order of their degrees $d_G(v)$
5:  for each $v \in V$ do
6:      $c_v \leftarrow d_G(v)$
7:      for each $u \in \mathtt{Neighbors}(v)$ do
8:          if $d_G(u) > d_G(v)$ then
9:              $d_G(u) \leftarrow d_G(u) - 1$
10:             Reorder V accordingly
    return Core numbers $c_i, \forall i \in V$
```

one and the process is repeated until no more remaining nodes have left in the graph. Since each node and edge is removed exactly once, the running time of the algorithm is $\mathcal{O}(n + m)$ [121]. Batagelj and Zaveršnik later proposed an $\mathcal{O}(m)$ algorithm for k-core decomposition [15] presented in 2.1.

This algorithm considers unweighted and undirected graphs. Nevertheless lot of effort has been made from researchers towards extending the k-core decomposition to other types of graphs. Refs. [60, 68, 71] present extensions on *weighted graphs* while Giatsidis et. al [70] present an extension for *directed graphs*. A core decomposition on *probablistic graphs* is proposed by Bonchi et. al [24]. In a *probabilistic or uncertain graph* every edge is associated with a probability of existence.

2.3.1.2 *Path-based centralities*

Closeness centrality

In a connected graph $G = (V, E)$ the closeness centrality of a node is calculated as the sum of the length of the shortest paths between the node and all other nodes in the graph. For a node $v \in V$ it can be calculated as follows [65, 147]:

$$CC_v = \frac{n-1}{\sum_{u \neq v} d_{uv}} \tag{2.4}$$

where $n = |V|$ and d_{uv} expresses the distance between vertices u and v. Nevertheless, when the graph is not connected there exist some node pairs for which $d_{uv} = \infty$. In this case closeness centrality is defined in terms of the inverse of the harmonic mean distances between the nodes:

$$CC_v = \frac{1}{n-1} \frac{1}{\sum_{u \neq v} d_{uv}} \tag{2.5}$$

Betweeness centrality

The betweeness centrality [17, 152, 153] is a centrality measure based on the notion of shortest paths between nodes. In an unweighted and connected graph, there exists between every pair of nodes one shortest path such that the number of edges passing through is minimized. For weighted and connected graphs it is the sum of the weights that is minimized [64]. Then, the betweeness centrality of each node is the number of these shortest paths passing through this node. It is calculated as follows:

$$BC_v = \sum_{v \neq x, v \neq y, x \neq y} \frac{g_{xy}^{v}}{g_{xy}} \tag{2.6}$$

where g_{xy} is the number of the shortest paths between nodes x and y and g_{xy}^{v} is the number of the paths which pass through v among all the shortest paths between x and y.

2.3.2 *Iterative refinement centralities*

Eigenvector centrality

The eigenvector centrality of a node v is proportional to the summation of the centralities of the nodes to which it is connected [22]. It is a measure of the importance of a node, denoted by x_v and is calculated as follows:

$$x_v = c \sum_{u=1}^{n} a_{vu} x_u \tag{2.7}$$

where $c = 1/\lambda$. c is a proportionality constant and λ is the largest eigenvalue of the adjacency matrix $\mathbf{A}$. The eigenvector centrality can be efficiently computed via the power iteration method [82] at the beginning of which each node is assigned with a score of 1. Every node shares its score in an even way to its neighbors and receives a new value during every iteration of the method. The process ends when the values of each node reach a steady state.

PageRank

PageRank (PR) is a famous variant of the eigenvector centrality which supposes that the importance of an entity in a network is determined by both the quantity and the quality of its connected neighbors. The PageRank algorithm [31] that calculates the respective centrality is used to rank webpages in the

Google search engine but also for other scenarios in the commercial sector. It distinguishes the importance of a webpage by performing a random walk on the network created by the linked webpages (i.e., one website referring to another creates a link between them). Initially, a node in the network, or equivalently a webpage, is assigned on unit PR value. Then every node distributes its PR value to its neighbors evenly using its outgoing edges. The PR value for a node v of graph G at a specific step t can be calculated as follows:

$$PR_v(t) = \sum_{u=1}^{n} a_{vu} \frac{PR_u(t-1)}{d_G^{out}(u)} \tag{2.8}$$

where n is the total number of nodes in G, $u \in \mathrm{Neighbors}(v)$ and $d_G^{out}(u)$ is the out-degree of node u. The algorithm stops when the PR values reach a steady state. Nevertheless, Eq. 2.8 cannot guarantee converge in some cases. An example of such a case is the existence of nodes with zero out-degree that cannot redistribute their PR value. For that reason, a jumping factor has been introduced assuming that the user will visit a web page with probability p, and close the current page and open a random page with probability 1-p.Then Eq. 2.8 is modified as follows:

$$PR_v(t) = p \sum_{u=1}^{n} a_{vu} \frac{PR_u(t-1)}{d_G^{out}(u)} + (1-p)\frac{1}{n} \tag{2.9}$$

2.4 DESCRIPTION OF GRAPH DATASETS

In this section, we briefly describe the graph datasets used in this thesis. We have considered data from different domains, including social, collaboration, information and technological networks. All datasets are publicly available. Table 2.1 provides a summary of the network statistics.

(i) EMAIL-ENRON. This email communication network created by email interaction between the members of Enron Corporation, and made public by the Federal Energy Regulatory Commission during its investigation [100]. It covers data from 150 users and a total of half a million messages. Each node represents an email address and an undirected edge was formed between two nodes if at least an address i sent an email to address j.

(ii) EMAIL-EUALL. This email network was collected from email communication of a large research institution [106]. To create the graph, each email address is considered as a node and an edge is created between two nodes if the latter have exchanged messages both ways. Overall there are $3,038,531$ emails between $287,755$ different email addresses recorded from October 2003 to May 2005.

(iii) EPINIONS. This is a trust-based (who-trusts-whom) online social network between the members of the Epinions.com (www.epinions.com) product review website [142]. The nodes of the network correspond to users of the website and the edges capture trust relationships between them. Although the network is signed, in our experiments we discard this information; we also convert the graph to an undirected one to use it on our experiments.

(iv) WIKI-VOTE. The graph was created from the online encyclopedia Wikipedia (www.wikipedia.org) and more precisely from the elections conducted to promote users to administrators (till January 2008) [105]. The nodes of the social network correspond to Wikipedia users and an edge between users i, j denotes that user i voted for user j.

(v) WIKI-TALK. This network is also created by data imported from Wikipedia [105]. Each user has a talk page where all interested users can edit to update various articles on Wikipedia. The specific dataset contains all discussions between users until January 2008. The nodes of this network correspond to users of Wikipedia and an edge from node i to node j indicates that user i edited a talk page of user j at least once.

(vi) SLASHDOT. Slashdot (slashdot.org) is a technology news website. The nodes of the social network that is created correspond to users and the edges capture friendship relationships among them (till February 2009) [108]. In fact, users are able to tag other users as friends or foes, forming a signed social network with positive and negative types of edges. In our experiments, we do not take into account the type of the edges.

(vii) HIGGS TWITTER DATASET. The Higgs Twitter dataset has been built after considering the information spreading process that was triggered after the announcement of the discovery of a new particle with the features of the elusive Higgs boson. The messages considered date between 1st and 7th July 2012. The activities that were taken into account in order to build the dataset are: i) re-tweets, ii) replies and iii) mentions. Of course the relations among users (friends/followers) relationships and information about activity on Twitter during the boson discovery were also taken into account.

(viii) NETHEPT.: The dataset constitutes a collaboration network taken from the "High Energy Physics (Theory)" section of *http://arxiv.org*, with nodes representing authors and edges capturing co-authorship relationships. Here, a user publishing a paper is considered as an action.

Network	|V|	|E|	**Description**
Email-Enron	33, 696	180, 811	E-mail communication network
Email-EuAll	224, 832	340, 795	E-mail communication network
Epinions	75, 877	405, 739	Who trusts whom network
Wiki-Vote	7, 066	100, 736	Elections of Wikipedia administrators
Wiki-Talk	2, 388, 953	4, 656, 682	User communication in Wikipedia
Slashdot	82, 168	582, 533	Slashdot social network (Feb. ’09)
Higgs	456, 626	14, 855, 842	Higgs Twitter Dataset (Mar. ’15)
NetHept	15, 233	62, 774	“High Energy Physics - Theory” collaboration network

Table 2.1: Networks datasets used in the thesis, along with basic statistics: nimber of nodes |V|; number of edges |E|.

3 LOCATING INFLUENTIAL SPREADERS IN SOCIAL NETWORKS

Understanding and controlling spreading processes in networks is an important topic with many diverse applications. The problem which is of crucial importance for this task is to identify which entities act as influential spreaders. Our focus of this Chapter is to identify the entities that while acting individually can propagate information to a large portion of the network. We capitalize on the properties of the K-truss decomposition, a triangle-based extension of the core decomposition of graphs, to locate individual influential nodes. Our analysis on real networks indicates that the nodes belonging to the maximal K-truss subgraph show better spreading behavior compared to previously used importance criteria, including node degree and k-core index, leading to faster and wider epidemic spreading. We further explored the centralities of the entities that are involved in a spreading process and showed that epidemic models cannot reproduce diffusion in terms of the evolution of the centralities of the infected nodes.

3.1 INTRODUCTION

Spreading processes in complex networks have gained great attention from the research community due to the plethora of applications that they occur. Typically, the interactions among individuals are responsible for the formation of information pathways in the network and to this extend, their position and topological properties have direct effect to the spreading phenomena occurring in the network. That way, a fundamental aspect on understanding and controlling the spreading dynamics is the identification of influential spreaders that can diffuse information to a large portion of the network.

As we discussed in the Introduction, the problem of identifying nodes with good spreading properties in networks, can be split in two subtopics: (i) identification of individual and (ii) identification of a group of nodes that are able to maximize the total spread of influence. In this Chapter, we focus on the problem of identifying single influential spreaders in networks. A straightforward approach towards finding effective spreading predictors, is to consider node centrality criteria and in particular the one of degree centrality. In fact, several studies have examined how the existence of heavy-tailed degree distribution in real-world networks [3, 61, 131] is related to cascading effects concerning the robustness of such complex systems [2, 3, 50, 134]. Nevertheless, there exist cases where a node can have arbitrarily high degree, while its neighbors are not well-connected, making degree a not very accurate predictor of the spreading properties. For example, this can occur when a high degree node is located to the periphery of the network. In fact, the spreading properties of a node are strongly related to the ones of its neighbors in the graph, and thus, global centrality criteria seem to be more appropriate for this task.

Of particular importance is the work by Kitsak et al. [97], which stressed out that highly connected nodes or those having high betweenness and closeness centralities, have little effect on the range of the spreading process. The main finding of their work was that, less connected but strategically placed nodes in the core of the network, are able to disseminate information to a larger part of the population. To quantify the core-periphery structure of networks, they applied the k-core decomposition algorithm [16, 33, 150] – a pruning process that removes nodes which do not satisfy a particular degree-based threshold. Their results indicated that nodes belonging to the maximal k-core subgraph are able to infect a larger portion of the network, compared to node degree or betweenness centrality, making the k-core number of a node a more accurate spreading predictor. Furthermore, extracting the k-core subgraph is a more efficient task compared to the heavy computation required by some centrality criteria (e.g., betweenness). Nevertheless, the resolution of k-core decomposition is quite coarse; depending on the structure of the network, many nodes will be assigned the same k-core number at the end of the process, even if their spreading capability differs from each other. Furthermore, building upon the good performance of the k-core decomposition, several extensions have been proposed [7, 13, 28, 83, 136, 165, 167].

Our proposed approach moves on a similar axis as the one by Kitsak et al. [97]; we argue that the topological properties of the nodes play a crucial role towards understanding their spreading capabilities. In particular, we consider that only a relatively small fraction of the nodes extracted by the k-core decomposition method corresponds to highly influential nodes. To that end, we propose the K-truss decomposition of a graph [48, 160, 168], a triangle-based extension of the k-core decomposition, as a more accurate method to identify privileged spreaders. The algorithm is able to extract a more refined and even more dense subgraph of the initial graph – compared to the k-core decomposition – as the K-truss is structurally more close to a clique.

The main contributions of this work can be summarized as follows:

- K-*truss decomposition for locating influential nodes:* The K-truss decomposition algorithm is proposed, as a mechanism to identify nodes with good spreading properties in the network.
- *Evaluation of our proposed approach on real graphs:* We used large scale real-world graphs while performing our experiments and showed that the maximal K-truss subgraph of the network can reveal those nodes that show better spreading behavior compared to previously used importance criteria.
- *Exploration of the centralities of the entities involved in a diffusion process:* We performed additional experiments where we trigger a diffusion process by different groups of influential spreaders and study the centralities of the entities involved in the process. We also compare the simulated diffusion process with real influence and discuss the capability of epidemic models in reproducing a real diffusion.

The rest of the Chapter is organized as follows. Section 3.2 presents the background concepts that are used throughout the Chapter and Section 3.3 reviews the related literature on the problem of identifying individual influential nodes in networks. Then, in Section 3.4 we present a detailed experimental evaluation of our proposed method for identification of privileged entities using the K-truss decomposition. In Section 3.5 we explore the centralities of the entities that are involved in a spreading process. Finally, in Section 3.6 we present concluding remarks.

3.2 PRELIMINARIES AND BACKGROUND

In this Section we present the preliminary concepts upon which we present the findings of this Chapter. We briefly recall the notion of the k-core decomposition, present the K-truss decomposition and the epidemic models that are used in order to locate individual influential spreaders in networks. A list of the symbols used in the Chapter is presented in Table 3.1.

3.2.1 k-*core decomposition*

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ edges and let H be a subgraph of G, i.e., $H \subseteq G$. Subgraph H is defined to be a k-core subgraph of G, denoted by C_k,

Symbol	Definition
$G = (V, E)$	Undirected graph G
V, E	Node and edge set of graph G
$n = \|V\|, m = \|E\|$	Number of node and edges of G
d_v	Degree of node $v \in V$
c_v	Core number of node $v \in V$
$Nb(v)$	Set of neighbors of node v
$\triangle_{uvw}$	Triangle subgraph defined by nodes u, v, w
T_K	K-truss subgraph
C_k	k-core subgraph
$t_{edge}(e)$	Truss number of edge $e \in E$
t_v	Truss number of node $v \in V$
$\mathscr{C}$	Set of nodes with maximum core number value c
$\mathscr{T}$	Set of nodes with maximum t_v value
$\mathscr{D}$	Set of nodes with maximum d_v values
M_v	Average infection size caused by node v
τ	Epidemic threshold

Table 3.1: List of symbols and their definitions.

if it is a maximal connected subgraph in which all nodes have degree at least k. Then, each node $v \in V$ has a core number $c_v = k$, if it belongs to a k-core but not to a $(k+1)$-core. We denote as $\mathscr{C}$ the set of nodes with the maximum core number k_{max} (i.e., the nodes of the k-core subgraph of G that corresponds to the maximum value of k) [150]. A detailed description is given in Chapter 2, Section 2.3.1.1.

3.2.2 K-*truss decomposition*

The K-truss decomposition extends the notion of k-core using triangles, i.e., cycle subgraphs of length 3 [48, 160].

Definition 3.1. *(Triangle subgraph). Let* $G = (V, E)$ *be an undirected graph. We define as a triangle* $\triangle_{uvw}$ *a cycle subgraph of nodes* $u, v, w \in V$. *Additionally, the set of triangles of* G *is denoted by* $\triangle_G$.

Definition 3.2. *(Edge support). The support of an edge* $e = (u, v) \in E$ *is defined as* $sup(e, G) = |\{\triangle_{uvw} : \triangle_{uvw} \in \triangle_G\}|$ *and expresses the number of triangles that contain edge* e.

Definition 3.3. *(*K*-truss subgraph). Then, the* K*-truss,* $K \geqslant 2$*, denoted by* $T_K = (V_{T_K}, E_{T_K})$*, is defined as the largest subgraph of* G*,*

Algorithm 3.1 K-truss decomposition

```
1: Input: Undirected graph $G = (V, E)$
2: Output: K-truss subgraphs, for $3 \leqslant K \leqslant K_{max}$
3: $K \leftarrow 2$
4: for each $e = (u, v) \in E$ do
5:     $sup(e) = |Nb(u), Nb(v)|$
6: while $|E| \neq \emptyset$ do
7:     while $(\exists e = (u, v) : sup(e) < K - 2)$ do
8:         $W \leftarrow Nb(u) \cap Nb(v)$
9:         for each $e' = (u, w)$ or $e' = (v, w)$, where $w \in W$ do
10:            $sup(e') \leftarrow sup(e') - 1$
11:        Remove $e$ from $G$
12:    Output $G$ as the K-truss subgraph
13:    $K \leftarrow K + 1$
   return $T_K$ for $3 \leqslant K \leqslant K_{max}$
```

where every edge is contained in at least $K - 2$ triangles within the subgraph, i.e., $\forall e \in E_{T_K}, sup(e, T_K) \geqslant K - 2$.

Definition 3.4. *(Edge truss number). The truss number of an edge* $e \in E$ *is defined as* $t_{edge}(e) = \max\{K : e \in E_{T_K}\}$. *Thus, if* $t_{edge}(e) = K$, *then the edge belongs to* T_K *but not to* T_{K+1}, *i.e.,* $e \in E_{T_K}$ *but* $e \notin E_{T_{K+1}}$. *We use* K_{max} *to denote the maximum truss number of any edge* $e \in E$.

Definition 3.5. *(Node truss number). The truss number of a node* $v \in V$, *denoted by* t_v *is the maximum truss number of its incident edges, i.e.,* $t_v = \max\{t_{edge}(e),\ e = (v, u) \forall\ u \in N(v)\}$, *where* $N(v)$ *is the set of neighborhood nodes of* v.

Definition 3.6. *(K-class). The K-class of a graph* $G = (V, E)$ *is defined as* $\Phi_K = \{e : e \in E, \tau_{edge}(e) = K\}$.

Definition 3.7. *(K-truss decomposition). Then, the K-truss decomposition is defined as the task of finding the K-truss subgraphs of* G, *for all* $2 \leqslant K \leqslant K_{max}$. *That is, the K-truss can be obtained by the union of all edges that have truss number at least K, i.e.,* $E_{T_K} = \bigcup_{j \geqslant K} \Phi_j$.

The computation of the K-truss subgraph, for a specific value of $K \geqslant 2$, follows similar methodological procedure as the one of k-core, where instead of the degree of a node, we examine the number of triangles that the node participates to: remove all edges $e = (u, v) \in E$ if they do not participate to at least $K - 2$ triangles, i.e., $|N(u), N(v)| \leqslant K - 2$. We denote as $\mathscr{T}$ the set of nodes with maximum node truss number (in other words, this set contains the nodes of the maximal K-truss subgraph). Algorithm 3.1 presents the K-truss decomposition of a graph [160].

Step 8 of the algorithm requires time $\mathcal{O}(d(u) + d(v))$ for each edge $e = (u, v) \in E$, giving total time complexity proportional to

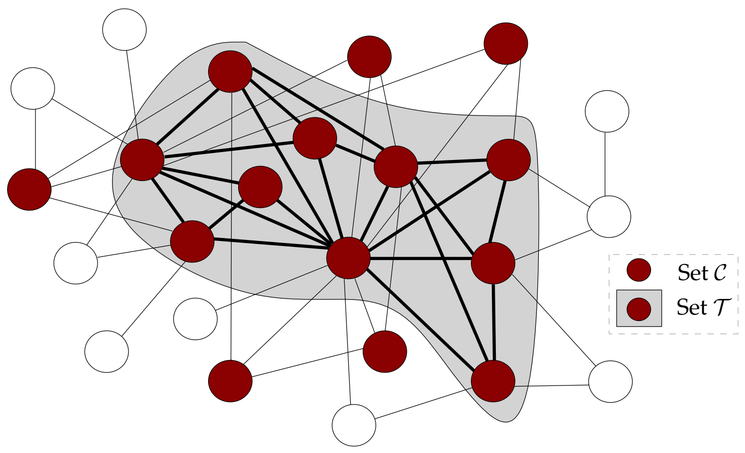


Figure 3.1: **Schematic representation of the maximal k-core and K-truss subgraphs of a graph.** The red colored nodes correspond to the 3-core subgraph of the graph (set $\mathscr{C}$); the gray shadowed region indicate the 4-truss subgraph (set $\mathscr{T}$).

$\mathscr{O}\Big(\sum_{e=(u,v)\in E} d(u)+d(v)\Big) = \mathscr{O}\Big(\sum_{v\in V} d^2(v)\Big)$. Also, the algorithm starts from an initialization that computes the support of each edge in G. Then all the edges are sorted in ascending order of their support. The computation of the support of the edges can be done in $\mathscr{O}(m^{1.5})$ time by the in-memory triangle counting algorithm [102, 149] as Wang and Cheng proposed [160]. The time complexity of the improved algorithm is $\mathscr{O}(m^{1.5})$ and the space complexity $\mathscr{O}(m+n)$.

It has been shown that the maximal k-core and K-truss subgraphs (i.e., maximum values for k, K) overlap, with the latter being a subgraph of the former; in fact, K-truss represents the core of a k-core that filters out less important information. Figure 3.1 shows an example of a graph and its k-core and K-truss subgraphs respectively. The red colored nodes correspond to the set $\mathscr{C}$ (i.e., the maximal k-core subgraph of the graph). The edges of the gray shadowed region are the edges that belong to the maximal K-truss of the original graph and the corresponding nodes are those belonging to set $\mathscr{T}$, i.e., the nodes with the maximum node truss number. In our approach, we argue that those nodes correspond to highly influential ones in the graph.

The K-truss decomposition is computationally a more difficult task compared to the one of k-core decomposition. In our approach, as we are only interested for the maximal K-truss subgraph, we take into account the structure shown in Figure 3.1. That is, we first compute the maximal k-core subgraph in linear time with respect to the total number of edges (subgraph defined by the red colored nodes in Figure 3.1) and then we extract the maximal K-truss subgraph (gray area in Figure 3.1).

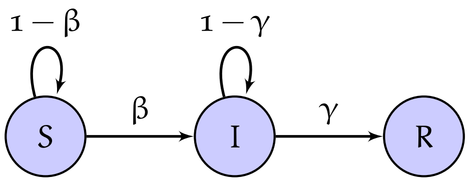


Figure 3.2: State diagram of the SIR model.

That way, the overall complexity of the task is significantly reduced.

3.2.3 *Epidemic models*

Modelling a spreading process is an active research topic that has been troubling researchers from various fields (epidemiology, social science, computer science etc.). As our approach is based on simuating a diffusion phenomenon with an epidemic model, we will be presenting a description of the Susceptible-Infected-Revovered (SIR) and Susceptible-Infected-Susceptible (SIS) models which are the most commonly used in the litterature. For a general introduction to epidemic models the reader may refer to Refs. [12, 92, 95, 130].

3.2.3.1 *The SIR model*

The SIR model [54] is a model that is used to describe an acute infectious disease. The latter refers to an infection which presents a rapid immune response after a short period of time.

The model assumes a population of N individuals, divided on the following three states.

- *Susceptible (S)*: the individual is not yet infected, thus being susceptible to the epidemic;
- *Infected (I)*: the individual has been infected with the disease and it is capable of spreading the disease to the susceptible population;
- *Recovered (R)*: after an individual has experienced the infectious period, it is considered as removed from the disease and it is not able to be infected again or to transmit the disease to others (immune to further infection or death).

Every individual that is on the I state can infect individuals with probability β, called *infection rate*, and afterwards it can recover with probability γ, called *recovery rate*. The state diagram of the model is presented in Figure 3.2.

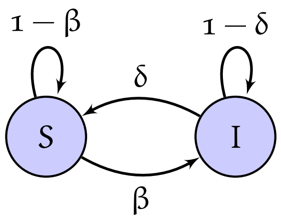


Figure 3.3: State diagram of the SIR model.

Let $S(t)$, $I(t)$ and $R(t)$ be the number of susceptible, infected and recovered individuals at time t. Then, the model can be described by the following differential equations:

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta SI}{N} \\ \frac{dI}{dt} &= \frac{\beta SI}{N} - \gamma I \\ \frac{dR}{dt} &= \gamma I. \end{aligned} \tag{3.1}$$

The last equation can be considered as redundant, since $S(t) + I(t) + R(t) = N$. In the limit of a large population of size N, analytical solutions for several quantities can be derived from these equations, such as the size of the outbreak.

3.2.3.2 *The SIS model*

The SIS model is used to simulate infections for which there is no long-lasting immunity where an individual can be infected numerous times. Examples of such diseases are rotaviruses, sexually transmitted infections and many bacterial infections.

The model assumes a population of N individuals, divided on the two states *Susceptible*(S) and *Infected*(I) as described for the SIR model. The model can be described by the following differential equations:

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta SI}{N} + \delta I \\ \frac{dI}{dt} &= \frac{\beta SI}{N} - \delta I \end{aligned} \tag{3.2}$$

We see that $\frac{dS}{dt} + \frac{dI}{dt} = 0 \Rightarrow S(t) + I(t) + R(t) = N$. δ is the fraction of the infected individuals that recover and re-enter the susceptible class per unit time. The state diagram of the model is presented in Figure 3.3.

3.2.4 *The SIR model applied in networks*

In our experiments throughout the Chapter we will be using the SIR model to simulate a spreading process. Based on its definition, the model assumes a *fully mixed* population: an infected individual can equally infect any other member of the population to which it belongs to. In order to apply the model in a population of individuals which form connections between them (i.e., a network) we follow a more realistic approach. In such a case any susceptible node can only be infected by an infected neighbor on the graph.

As we will present in Section 3.4, initially all the nodes of the network are set at the susceptible state S, except from the one that we are interested to examine its performance which is set at the infected state I. Then, at each time step t of the process, every node that is on the I state can infect its susceptible neighbors with probability β and afterwards it can recover with probability γ. Note that, a node cannot directly pass from state I to state R during the same time step t.

3.3 RELATED WORK

In this section, we present the related work for the problem of identification of individual influential spreaders in networks.

3.3.1 *Identification of Individual Influential Spreaders in Networks*

Numerous centrality criteria have been proposed in order to locate privileged spreaders in networks. Lu et al. [114] proposed LeaderRank, a random walk-based algorithm similar to PageRank [30] for identifying influential users in social networks. Later, Li et al. [109] extended LeaderRank to properly detect influential nodes in weighted networks. Chen et al. [39] proposed a semi-local centrality measure which serves as a trade-off between degree and other computationally complex measures (betweeness and closeness centrality). Additionally, Chen et al. [37] proposed ClusterRank, a local ranking method that takes into account the clustering coefficient of a node while in another approach [38], the diversity of the paths that emanate from a node was considered. The main idea was that the spreading ability of a node may be reduced if its propagation depends only on a few paths, while the rest ones lead to dead ends.

Building upon the fact that the k-core decomposition is an effective (and efficient) measure to capture the spreading properties of nodes, as introduced by Kitsak et al. [97], several extensions have been proposed. The authors of Ref. [165] introduced a modified version of the k-core decomposition in which the nodes are ranked taking into account their connections to the remaining nodes of the graph as well as to the removed nodes at previous steps of the process. They showed that the proposed node ranking method is able to identify nodes with better spreading properties compared to the traditional k-core decomposition. Bae et al. [7] extended the metric of k-core number of each node by considering the core number of its neighbors. That way, the ranking produced by the method is more fine-grained in the sense that the effect of assigning the same score (i.e., k-core number) to many nodes is eliminated. Basaras et al. [13] proposed to rank the nodes according to a criterion that combines the degree and the k-core number of a node within an μ-hop neighborhood. In Ref. [83] the authors introduced a criterion that combines three previously examined measures, namely degree, betweenness centrality and core number. The intuition was that, most of the widely used centrality criteria produce highly correlated rankings of nodes; combining them in a proper way, we are able to achieve a more accurate indicator of influential nodes. Zhang et al. [167] proposed a method to locate influential nodes taking into account the existence of community structure in networks. In Ref. [28], the authors considered real social media data, in order to examine to what extend the structural position of a user in the network allows us to characterize the ability of an individual to spread rumors effectively. Their results indicate that although the most appropriate feature is the degree of a node, only a few such highly-connected individuals exist; however, by considering the k-core number metric, we are able to locate a larger set of individuals that are likely to trigger large cascades. For a detailed review in the area, we refer to the article by Pei and Makse [136]. It is important to stress out that most of the above mentioned extensions can also be applied to the proposed K-truss decomposition-based approach.

3.4 K-TRUSS DECOMPOSITION FOR IDENTIFYING INFLUENTIAL NODES

In this Section the experimental results of our proposed method for identification of influential nodes are presented.

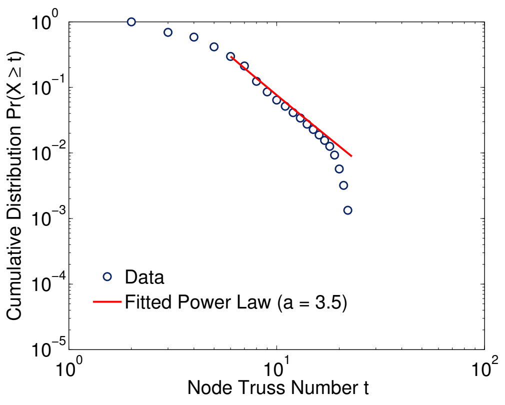
DATASETS We study real-world networks arising from online social networking and communication platforms. In particular, we investigate the following network datasets: (i) EMAIL-ENRON, (ii) EMAIL-EUALL, (iii) EPINIONS, (iv) WIKI-VOTE, (v)

Wiki-Talk and (vi) Slashdot. All datasets are considered undirected and unweighted; also, the largest connected component was used in the experiments. High level characteristics of the networks are shown in Table 3.1. A more detailed description of the datasets is presented in Section 2.4 of Chapter 2.

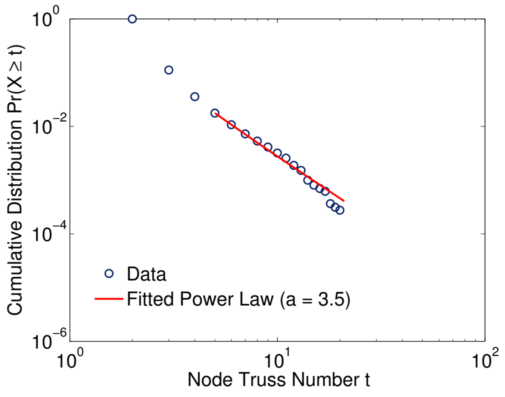
PROPERTIES OF THE K-TRUSS SUBGRAPHS We have examined the distribution of the node truss numbers t_{node} of the graphs presented in Table 3.1 and the results are depicted in Fig. 3.1. Each plot shows the complementary cumulative distribution function (CCDF) of the nodes' truss number in log-log scale. As we can observe, in most of the cases the distribution is skewed, indicating that very few nodes have high truss number; the majority of the nodes belong to "low" K-truss subgraphs, i.e., small values of parameter K of the decomposition. We have also fitted a power-law distribution [45] to the data (red colored line) and the exponent is shown in Fig. 3.1. Here, we do not claim that the truss number distribution is fully captured by a power-law; nevertheless, it corresponds to heavy-tailed distribution and this fact can help us to better understand the underlying properties of the data. In our case, this means that we can reduce the graph into a subgraph with exponentially smaller size and try to locate influential spreaders within this subgraph.

We have additionally examined the maximum level of the K-truss decomposition, i.e., value K_{max}, for the various graphs. As we can observe from Table 3.1, K_{max} values vary from dataset to dataset, but compared to the k_{max} values of the k-core decomposition, they tend to be much smaller. This is rather expected since the K-truss decomposition relies on triangle participation, which is a more strict criterion compared to node degree. This last point is also a justification for the differences on the number of nodes belonging to the truss set $\mathscr{T}$ and core set $\mathscr{C}$ (i.e., the set of nodes belonging to the maximal k-core subgraph of the graph). Although these sets are overlapping, the one that corresponds to K-truss has significantly smaller size compared to the maximal k-core subgraph. This was also one of the motivations of the proposed work; since the nodes of the maximal k-core subgraph perform well in information spreading, how to further refine this set by selecting a small subset that is characterized by even better spreading properties.

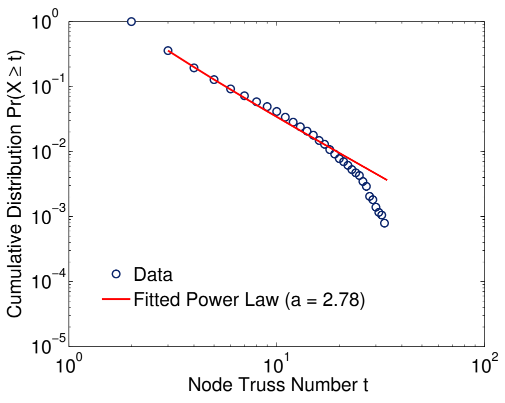
METHODOLOGY In the experimental results that follow, we are comparing the spreading performance of the nodes belonging to the set $\mathscr{T}$ (**truss** method), to those belonging to the set $\mathscr{C} - \mathscr{T}$ (**core** method), i.e., the nodes belonging to the maximal



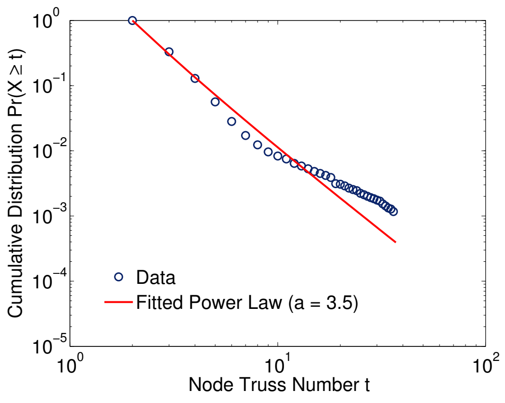
(a) Email-Enron



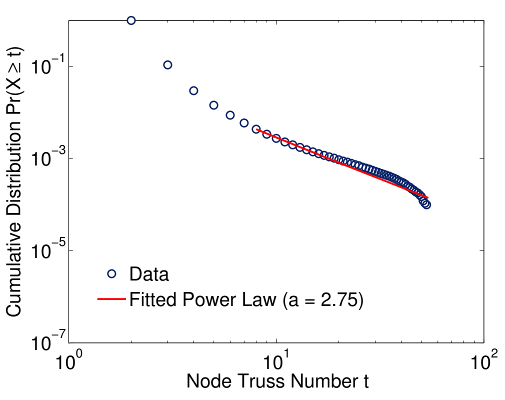
(b) Email-EuAll



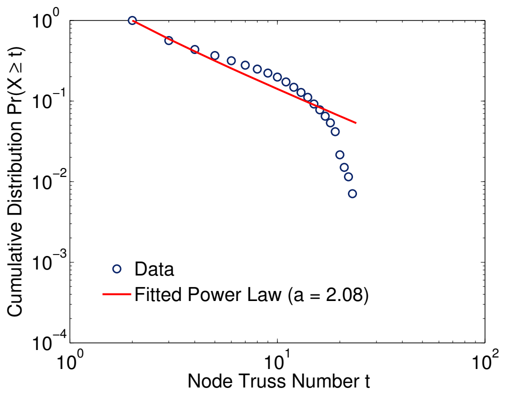
(c) Epinions



(d) Slashdot



(e) Wiki-Talk



(f) Wiki-Vote

Figure 3.1: Complementary cumulative truss number distribution function. Each plot depicts the distribution of the truss numbers for the nodes of the graph on log-log scale. The red line corresponds to the fitted power-law distribution.

Network Name	Nodes	Edges	k_{max}	K_{max}	$\lvert\mathscr{C}\rvert-\lvert\mathscr{T}\rvert$	$\lvert\mathscr{T}\rvert$	τ
Email-Enron	33,696	180,811	43	22	230	45	0.00840
Epinions	75,877	405,739	67	33	425	61	0.00540
Wiki-Vote	7,066	100,736	53	23	286	50	0.00720
Email-EuAll	224,832	340,795	37	20	230	62	0.00970
Slashdot	82,168	582,533	55	36	38	96	0.00074
Wiki-Talk	2,388,953	4,656,682	131	53	463	237	0.00870

Table 3.1: Properties of the real-world graphs used in this study (Table 3.1 provides definitions of the symbols). k_{max} and K_{max} denote the maximum k-core and K-truss numbers respectively (as produced by the decompositions); $|\mathscr{T}|$ represents the number of nodes belonging to set $\mathscr{T}$; $|\mathscr{C}|-|\mathscr{T}|$ represents the number of the nodes belonging to set $\mathscr{C}$, excluding the nodes that belong to set $\mathscr{T}$; τ is the epidemic threshold of the graph.

k-core excluding those that belong to the maximal K-truss of the graph – since $\mathscr{T}$ is subset of $\mathscr{C}$, as discussed above.

The **core** method constitutes the basic baseline approach, since it has been shown that outperforms other well known node importance criteria such as betweenness centrality [97]. For completeness in the experimental evaluation, we also compare the spreading capabilities of the nodes that belong to the maximal K-truss subgraph to those belonging to the set $\mathscr{D}$ that contains the highest degree nodes in the graph (**top degree** method); we choose $|\mathscr{C}|-|\mathscr{T}|$ high degree nodes to achieve fair comparison between the different methods.

3.4.1 *Evaluating the spreading performance*

To study the spreading process and evaluate the performance of the nodes extracted by the K-truss decomposition method, we apply the SIR epidemic model (defined in Section X). Algorithm 3.2 presents the steps of the proposed framework for (i) selecting the initial node that will trigger the epidemic (cascade) and (ii) evaluate the impact of this individual node with respect to the epidemic spreading under the SIR model.

Initially, we choose a node that belongs to $\mathscr{T}$ (i.e., maximal K-truss subgraph) and set it to the infected (I) state. In general, the initial node can be any node of the graph; the same procedure is also performed for the baseline methods. The rest of the nodes are assigned to the susceptible state S. Notice that, we keep track of the infected, susceptible and recovered nodes for each time step of the process. At each time step, an infected node can infect a susceptible neighbor with probability β. Additionally, any node that got infected at previous time steps of the process, can recover with probability γ. The process is repeated until no more infected nodes left. Finally, the algorithm returns M_v which is the number of the infected individuals under the cascade triggered by node v.

Algorithm 3.2 Identify nodes and evaluate spreading process

```
1: Input: Undirected graph $G = (V, E)$, parameters $\beta, \gamma$
2: Output: Size of infected population $M_v$ for cascade triggered by node $v$
3: Select node $v \in \mathscr{T}$
4: $\mathtt{State}(v) \leftarrow I$, $\mathtt{State}(V \setminus v) \leftarrow S$        /* Initialization steps */
5: $I(0) \leftarrow \{v\}$, $S(0) \leftarrow V \setminus v$, $R(0) \leftarrow \emptyset$
6: $t \leftarrow 0$
7: repeat
8: $t \leftarrow t + 1$
9: $I(t) \leftarrow \emptyset$, $R(t) \leftarrow \emptyset$
10: for each node $w \in V$ do
11:     /* Infected (I) nodes can infect susceptible neighbors */
12:     if $\mathtt{State}(w) = I$ then
13:         for each node $z \in \{Nb(w) : \mathtt{State}(z) = S\}$ do
14:             $Pr(\mathtt{State}(z) \leftarrow I) = \beta$ (also $I(t) \leftarrow I(t) \cup \{z\}$)
15:     /* Nodes that got infected at previous time steps can recover (R) */
16:     if $\mathtt{State}(w) = I$ and $w \notin I(t)$ then
17:         $Pr(\mathtt{State}(w) \leftarrow R) = \gamma$ (also $R(t) \leftarrow R(t) \cup \{w\}$)
18: until $I(t) = \emptyset$       /* No more infected nodes left */ return $M_v \leftarrow I(1) \cup I(2) \cup \ldots \cup I(t)$
```

To evaluate the spreading efficiency of the methods, we focus on the following quantities:

(i) the number of nodes that become infected at each time step of the process and the corresponding cumulative one

(ii) the total number of infected nodes at the end of the epidemic

(iii) the time step where the epidemic fades out.

For each node, we repeat the simulation 100 times (10 times for the WIKI-TALK graph due to its large size) and report the average behavior. In each case, we repeat the above for all the respective nodes and calculate the average behavior for the nodes of each set (**truss** method versus the two baselines **core** and **top degree**).

The experimental results are shown in Table 3.2. We set β close to the epidemic threshold and parameter $\gamma = 0.8$, as used by Kitsak et al. [97]. The values of parameter β of the SIR model for each graph, are shown in Table 3.1. Tables 3.2 and 3.3 show the number of the newly infected nodes for some of the first ten time steps of the spreading process, which we consider as the outbreak of the epidemic and the cumulative number of infected nodes per step respectively. We also report the total number of nodes that were infected at the end of the process (*Final step*) and the time step where the epidemic dies out (*Max step*).

As we can observe, the **truss** method achieves significantly higher infection rate during the first steps of the epidemic. Furthermore, in almost all cases, the total number of infected nodes at the end of the process (*Final step*) is larger, while the fade out occurs earlier (*Max step*). Lastly, as we discussed above, the number of nodes in the truss set $\mathscr{T}$ is much smaller compared to the set $\mathscr{C} - \mathscr{T}$ (Table 3.1). By refining significantly the set of influential nodes in truss set $\mathscr{T}$, the "weaker" spreaders of $\mathscr{C}$ are left in core set $\mathscr{C} - \mathscr{T}$, explaining the inferior behavior of the **core** method compared to **top degree**.

Some small deviations from this behavior are observed in the Slashdot and Wiki-Talk graphs. In the Slashdot graph, the best performance is achieved by the **top degree** method, which from the very first steps is able to infect a larger amount of nodes. In the case of the Wiki-Talk graph, although the total number of infected nodes at the end (*Final step*) of the epidemic is almost the same for all methods, the proposed **truss** method performs quite effectively at the first steps of the process. In fact, it significantly outperforms both baseline methods achieving an increase of almost 23% on the cumulative number of infected nodes compared to both **core** and **top degree** methods, at the sixth step of the process.

We have also computed the cumulative difference of the number of infected nodes per step achieved by the methods. Let $I_t^{\textbf{truss}}$ be the number of infected nodes at step t achieved by the **truss** method (similar for **core** and **top degree**). We define the cumulative difference for the **truss** and **core** methods at step t as

$$D_t^{\textbf{truss-core}} = \underset{z=1\ldots t}{\text{cumsum}}(I_z^{\textbf{truss}} - I_z^{\textbf{core}}). \tag{3.3}$$

Similarly, we can define the same quantity for the **truss** vs. **top degree** methods. The results are shown in Figures 3.2 and 3.3. For each graph, we have performed experiments for two values of parameter β and $\gamma = 0.8$. We observe that the cumulative difference of the number of nodes that are being infected at every step is always larger between **truss** and **core** than between **truss** and **top degree**. Both differences increase during the outbreak of the epidemic until they stabilize to the number of nodes

		Time Step											
	Method	2	3	4	5	6	7	8	9	10	*Final step*	σ	*Max step*
Email-Enron	**truss**	8.44	18.58	46.66	104.11	204.08	328.39	418.77	425.06	355.84	2, 596.52	136.7	33
	core	4.78	12.82	31.97	73.77	152.55	264.36	367.28	403.98	364.13	2, 465.60	199.6	37
	top degree	6.89	13.87	34.13	76.67	155.48	264.13	360.89	394.37	357.08	2, 471.67	354.8	36
Epinions	**truss**	4.17	9.25	19.70	39.56	75.04	130.48	204.14	278.69	329.08	2, 567.69	227.8	37
	core	3.45	7.18	14.72	29.11	55.27	98.11	158.56	226.17	280.03	2, 325.37	327.2	43
	top degree	4.22	7.94	16.03	31.32	58.84	103.91	166.23	234.96	289.49	2, 414.99	331.7	47
Wiki-Vote	**truss**	2.92	4.37	6.92	10.43	15.27	21.63	28.73	35.93	42.46	560.66	114.9	52
	core	1.92	3.07	4.78	7.22	10.65	15.18	20.66	26.70	32.40	466.01	104.5	57
	top degree	2.43	3.53	5.46	8.17	12.05	17.04	23.05	29.49	35.55	502.88	104.5	62
Email-EuAll	**truss**	11.62	28.04	62.25	127.79	240.97	405.53	584.87	705.89	725.42	5, 018.52	487.94	36
	core	9.85	18.69	40.82	82.28	158.72	279.41	433.81	574.97	644.76	4, 579.84	498.71	38
	top degree	17.96	16.74	39.93	73.66	144.69	384.07	503.18	565.06	548.25	4, 137.56	1, 174.84	39
Slashdot	**truss**	5.36	20.57	66.21	188.52	461.35	917.2	1, 390.52	1, 571.97	1, 359.99	8, 207.46	368.37	32
	core	6.48	19.68	61.13	168.36	410.19	820.77	1, 272.29	1, 486.5	1, 344.33	8, 002.76	518.43	32
	top degree	13.95	27.88	83.29	204.60	483.95	940.49	1, 426.81	1, 616.55	1, 403.80	8, 489.45	59.01	32
Wiki-Talk	**truss**	64.21	435.79	3, 259.05	16, 227.25	34, 543.23	23, 818.06	9, 853.84	3, 487.65	1, 186.41	93, 491.81	476.22	21
	core	41.77	269.96	2, 027.69	11, 169.2	31, 223.21	28, 732.06	13, 055.45	4, 805.11	1, 664.52	93, 496.50	767.35	23
	top degree	88.84	324.11	2, 475.01	11, 718.28	29, 694.45	27, 009.05	13, 720.15	5, 396.45	1, 937.89	93, 411.18	1, 166.77	24

Table 3.2: Average number of infected nodes per step of the SIR model using β close to the epidemic threshold of each graph and $\gamma = 0.8$. At the *Final step* column, we show the total number of infected nodes at the end of the process (*Max step*), with standard deviation σ.

		Time Step											
	Method	2	3	4	5	6	7	8	9	10	*Final step*	σ	*Max step*
Email-Enron	**truss**	9.44	28.03	74.69	178.80	382.88	711.27	1,130.05	1,555.11	1,910.95	2,596.52	136.7	33
	core	5.78	18.60	50.57	124.35	276.90	541.26	908.54	1,312.52	1,676.65	2,465.60	199.6	37
	top degree	7.89	21.76	55.90	132.57	288.05	552.18	913.07	1,307.45	1,664.53	2,471.67	354.8	36
Epinions	**truss**	5.17	14.42	34.13	73.69	148.74	279.23	483.37	762.06	1,091.14	2,567.69	227.8	37
	core	4.45	11.64	26.36	55.48	110.75	208.87	367.43	593.59	873.62	2,325.37	327.2	43
	top degree	5.22	13.16	29.20	60.52	119.36	223.27	389.49	624.46	913.95	2,414.99	331.7	47
Wiki-Vote	**truss**	3.92	8.30	15.23	25.66	40.94	62.57	91.31	127.25	169.71	560.66	114.9	52
	core	2.92	5.99	10.78	18.01	28.66	43.85	64.50	91.20	123.60	466.01	104.5	57
	top degree	3.43	6.96	12.43	20.61	32.66	49.70	72.75	102.25	137.81	502.88	104.5	62
Email-EuAll	**truss**	12.62	40.66	102.92	203.72	471.69	877.22	1,462.10	2,168.00	2,893.43	5,018.52	487.94	36
	core	10.85	29.55	70.37	152.65	311.38	590.79	1,024.60	1,599.57	2,244.34	4,579.84	498.71	38
	top degree	18.96	35.71	75.64	149.30	294.00	543.45	927.52	1,430.70	1,995.77	4,137.56	1,174.84	39
Slashdot	**truss**	6.36	26.93	93.14	281.67	743.03	1,660.23	3,050.75	4,622.73	5,982.73	8,207.46	368.37	32
	core	7.48	27.17	88.31	256.67	666.86	1,487.64	2,759.93	4,246.43	5,590.76	8,002.76	518.43	32
	top degree	14.95	42.84	126.13	330.74	814.69	1,755.18	3,181.99	4,798.55	6,202.35	8,489.45	59.01	32
Wiki-Talk	**truss**	65.21	501.00	3,760.06	19,987.31	54,530.55	78,348.62	88,202.46	91,690.11	92,876.53	93,491.81	476.22	21
	core	42.77	312.74	2,340.43	13,509.64	44,732.85	73,104.92	86,160.38	90,965.49	92,630.01	93,496.5	767.35	23
	top degree	89.84	413.95	2,888.96	14,607.24	44,301.69	71,310.74	85,030.90	90,427.35	92,365.25	93,411.18	1,166.77	24

Table 3.3: Cumulative number of infected nodes per step of the SIR model using β close to the epidemic threshold of each graph and $\gamma = 0.8$. At the *Final step* column, we show the total number of infected nodes at the end of the process (*Max step*), with standard deviation σ.

which is actually the final difference of the number of nodes that got infected (i.e., entered state I of the SIR model) during the epidemic process of the two compared methods. Clearly, as in almost all cases the differences are always above zero, one can conclude to the effectiveness of information diffusion when the spreading is triggered by the nodes that belong to the maximal K-truss subgraph.

3.4.2 *Comparison to the optimal spreading*

Since we lack ground-truth information about the best spreaders in the network, to further study the performance of the proposed K-truss decomposition method, we have examined the spreading achieved by each node of the graph. More precisely, we set each node $v \in V$ at the infected state I and simulate the spreading capabilities of this node using the SIR model, as described earlier. Figure 3.4 depicts the distribution of the nodes with respect to the infection size M, for the EMAIL-ENRON and WIKI-VOTE graphs (parameter β of the SIR model was set to $\beta = 0.01$ for this experiment). In both cases, the axes of the plot have been set to logarithmic scale. As we can observe, the distribution of the infection size M is skewed; only a small percentage of nodes are highly influential, while the majority of the nodes are able to infect only a small portion of the graph (small values of infection size M). Thus, our goal is to examine how the nodes detected by the K-truss decomposition are distributed on this small subset of spreading-efficient nodes. Note that, similar observations have been made for the rest of the graphs described at Table 3.1.

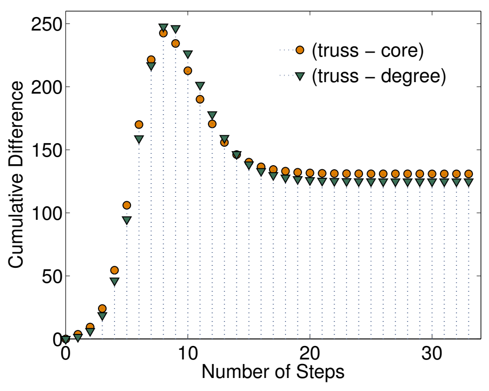
To that end, we rank the nodes $v \in V$ of the graph, according to the infection size $M(v)$. Let

$$\mathrm{OPT}_1 = \underset{v \in V}{\operatorname{argmax}}\ M(v) \tag{3.4}$$

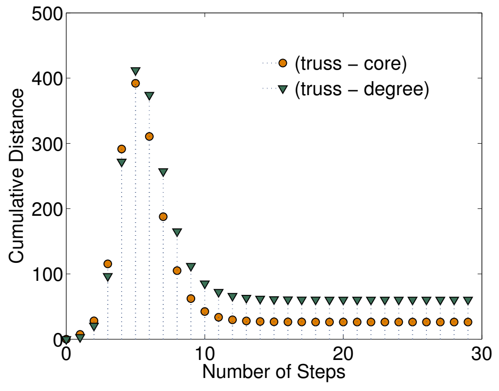
be the node that achieves that highest infection size M among all nodes in the graph, i.e., $\mathrm{OPT}_1 \geqslant \mathrm{OPT}_2 \geqslant \ldots \geqslant \mathrm{OPT}_{|V|}$. In order to examine how the nodes detected by the K-truss decomposition are distributed among the most efficient (optimal) spreaders, we consider a variable size window W over the ranked nodes and define $P_W^{\mathscr{T}}$ to be the fraction of nodes of set $\mathscr{T}$ that can be found within W as follows:

$$P_W^{\mathscr{T}} = \frac{|T_W|/|\mathscr{T}|}{|W|/|V|}, \tag{3.5}$$

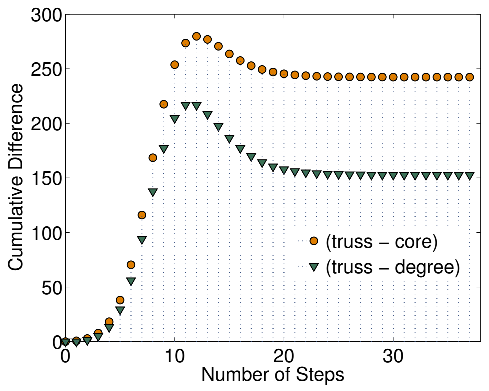
where T_W is the set of nodes $v \in \mathscr{T}$ that are located in the window W of size $|W|$ (in a similar way, we can define $P_W^{\mathscr{C}}$ for



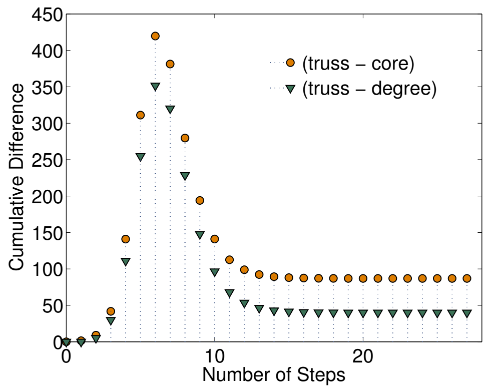
(a) Email-Enron: β = 0.01



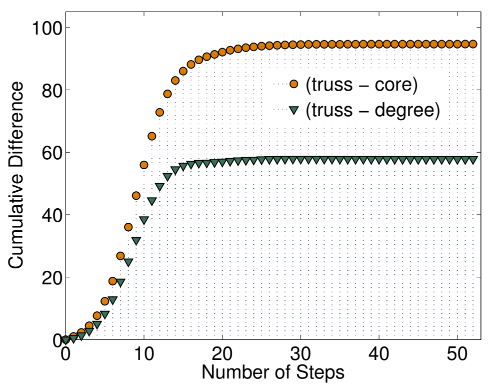
(b) Email-Enron: β = 0.03



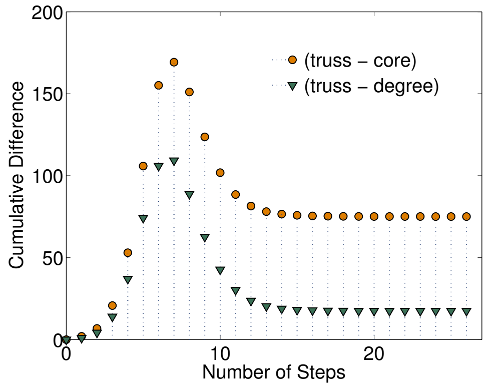
(c) Epinions: β = 0.007



(d) Epinions: β = 0.01



(e) Wiki-Vote: β = 0.009



(f) Wiki-Vote: β = 0.01

Figure 3.2: Cumulative difference of the infected nodes per step achieved by the **truss** method vs. the **core** (truss - core) and **top degree** (truss - degree) methods. Parameter γ of the SIR models is set to $\gamma = 0.8$. Continued in Fig. 3.3.

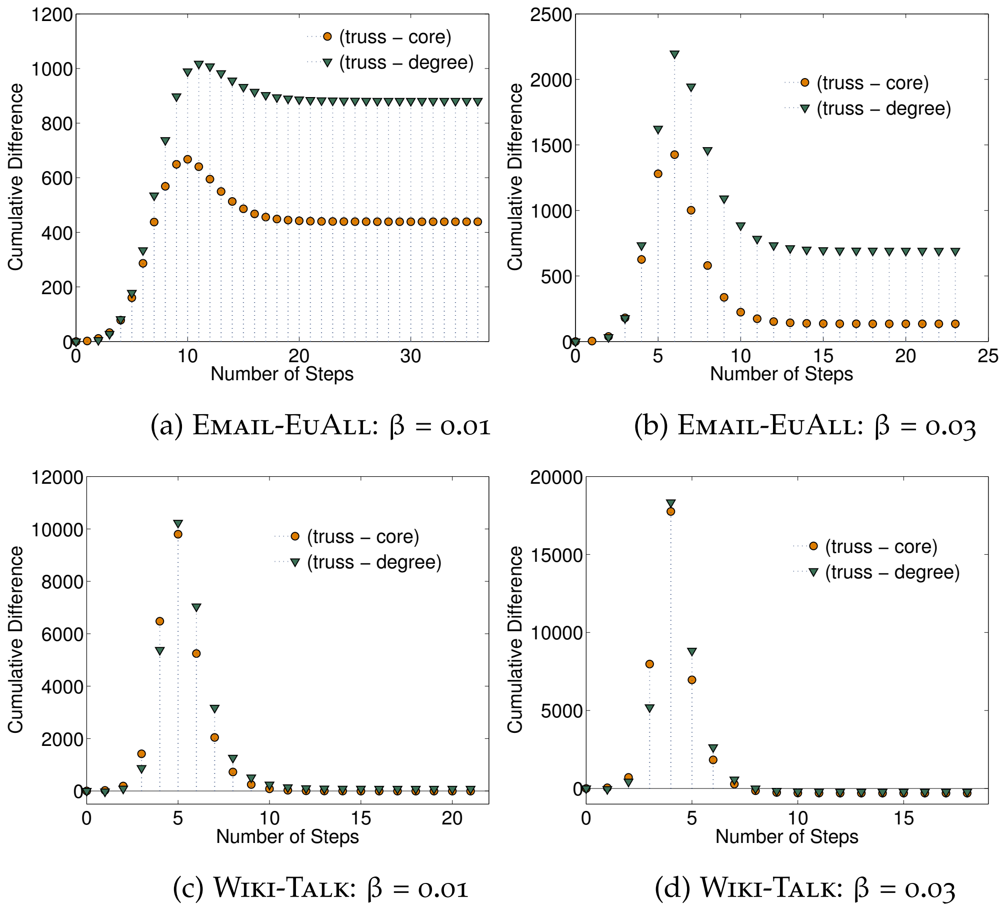
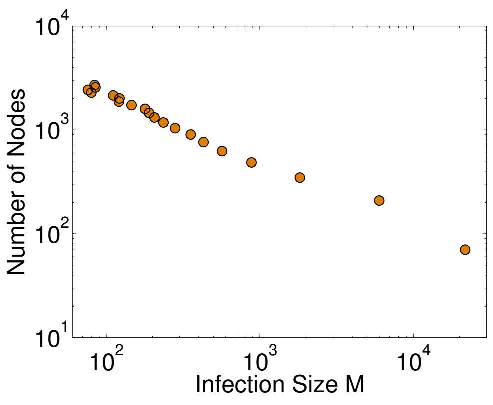


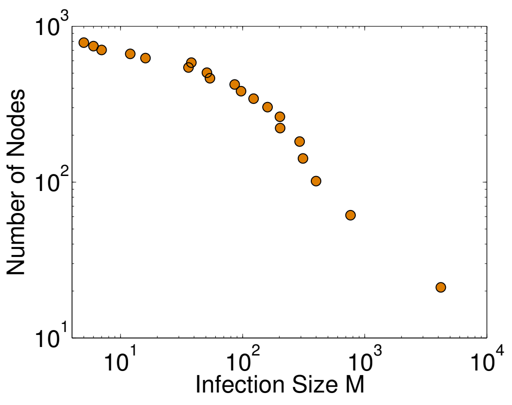
Figure 3.3: Cumulative difference of the infected nodes per step achieved by the **truss** method vs. the **core** (truss - core) and **top degree** (truss - degree) methods. Parameter γ of the SIR model is set to $\gamma = 0.8$.

the nodes of the maximal k-core subgraph). We are interested in examining how the quantities $P_W^{\mathscr{T}}$ and $P_W^{\mathscr{C}}$ behave with respect to the size of the window W.

Figure 3.5 depicts the distribution of the top-truss $P_W^{\mathscr{T}}$ and top-core $P_W^{\mathscr{C}}$ nodes, for various sizes of window W (i.e., fractions of the most efficient spreaders). As we can observe, for almost all datasets, $P_W^{\mathscr{T}}$ reaches the maximum value (i.e., 100%) relatively early and for small window sizes, compared to $P_W^{\mathscr{C}}$. The maximum value of $P_W^{\mathscr{T}}$ indicates that we have found all the nodes belonging to set $\mathscr{T}$ in the window of fractional size W. An early and intense upward trend of the curve implies that a large fraction of the nodes belonging to the set of interest ($\mathscr{T}$ or $\mathscr{C}$), corresponds to nodes with the best spreading properties on the graph. For example, in the Email-EuAll graph, the maximum for the nodes of set $\mathscr{T}$ is reached in window $W = 1.7\%$, while in the case of set $\mathscr{C}$ in window $W = 2.8\%$. Thus, the nodes detected by the K-truss decomposition method (set $\mathscr{T}$) are bet-



(a) Email-Enron



(b) Wiki-Vote

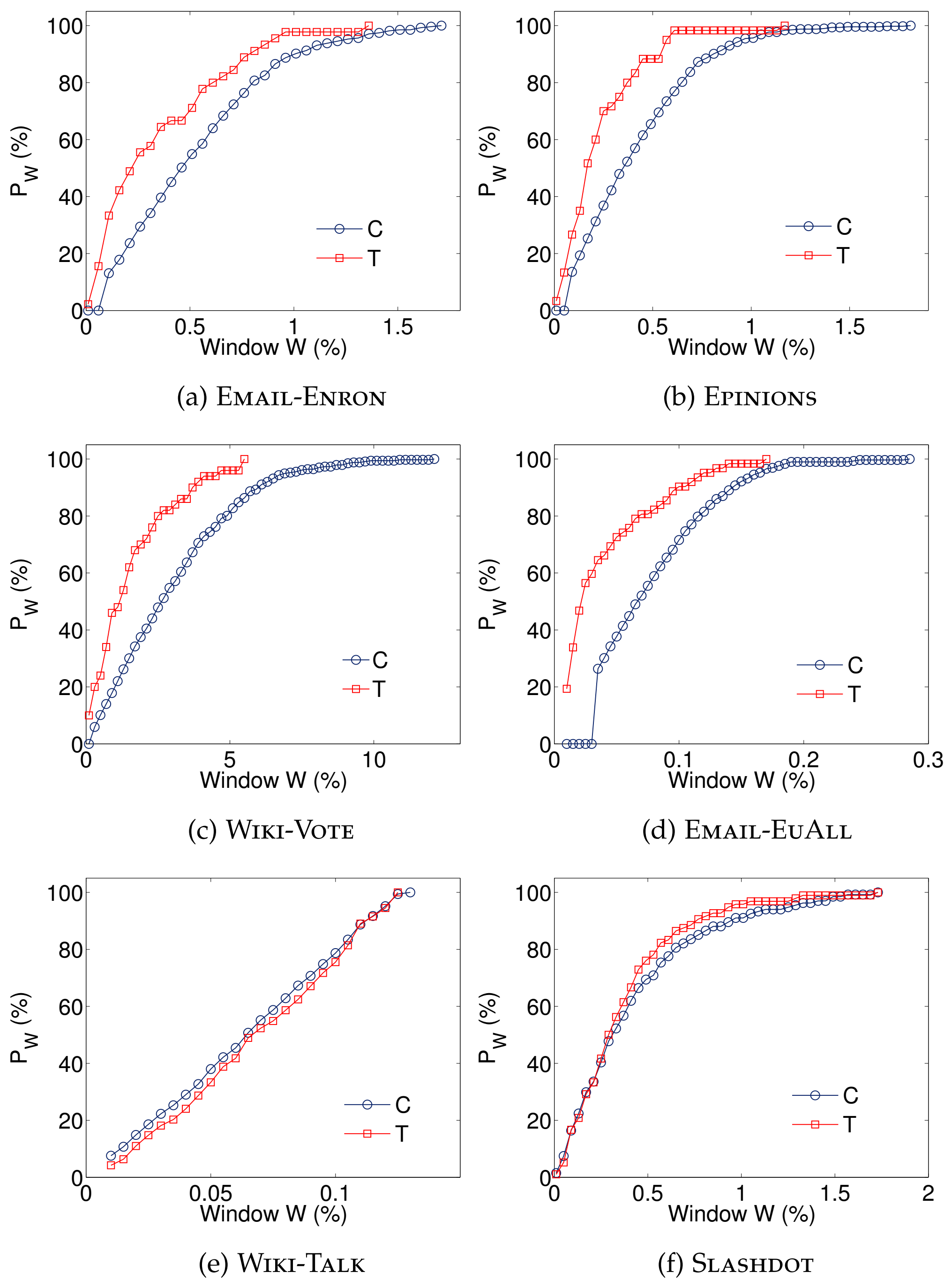
Figure 3.4: Spreading distribution of the nodes in the network, in log-log scale. The horizontal axis corresponds to the infection size M achieved by each node in the graph, after a binning process. The vertical axis captures the number of nodes that fall on each bin. Observe that only a small percentage of nodes achieves high spreading. In both cases, we have set $\beta = 0.01$ in the SIR model.

ter distributed among the most efficient spreaders, compared to those located by the k-core decomposition (set $\mathscr{C}$). A slightly different behavior is observed in the Wiki-Talk and Slashdot graphs; in both graphs, the values of $P_W^{\mathscr{T}}$ and $P_W^{\mathscr{C}}$ are very close to each other for almost all choices of window W, indicating that both sets have almost the same overlap with the set of optimal spreaders. Nevertheless, as we have already presented in Tables 3.2, for those two datasets the spreading performance of the truss nodes achieved during the first steps of the epidemic is much better.

Furthermore, we are interested to study the distribution of the nodes' truss number t_{node} with respect to window W. Similar to what described above, we consider a fraction of the best spreaders in the graph (as specified by W) and we examine the distribution of all truss numbers (and not only the maximum one) within it. Since nodes with high truss number are of particular importance here, we have considered groups of nodes as follows:

(i) Individual groups for each of the top five truss numbers, i.e., K_{max} to $K_{\text{max}-4}$. That way, the first group contains nodes with truss number equal to K_{max}, the second group nodes with truss number $K_{\text{max}-1}$ and so on.

(ii) The rest of the groups concern truss numbers in the range $K_{\text{max}-5}$ to $K = 2$, grouping together five consecutive truss numbers each time. For example, the sixth group contains



(a) Email-Enron (b) Epinions

(c) Wiki-Vote (d) Email-EuAll

(e) Wiki-Talk (f) Slashdot

Figure 3.5: Distribution of the top-truss $P_W^{\mathscr{T}}$ and top-core $P_W^{\mathscr{C}}$ nodes among the nodes with optimal spreading properties under a window of size W. Observe that for small values of window size W (i.e., closer to the optimal spreading), the number of top-truss nodes is always higher compared to the number of top-core nodes.

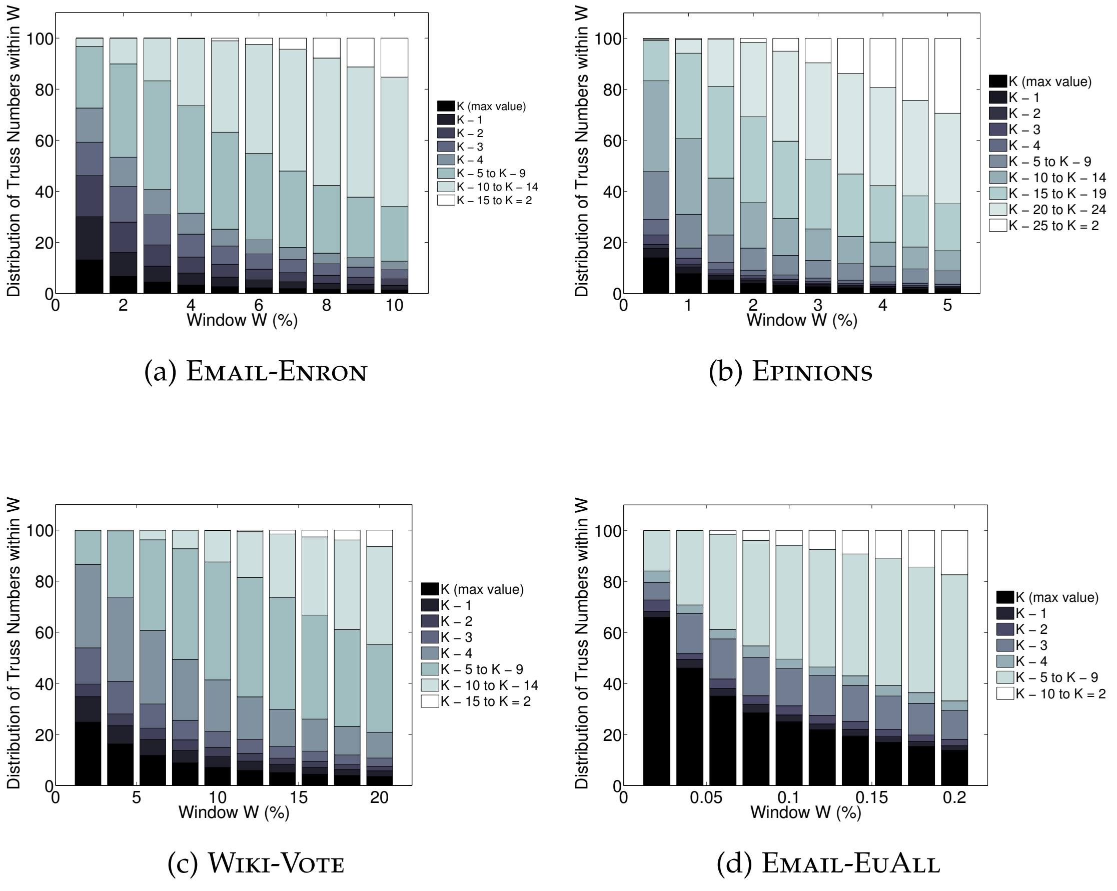


Figure 3.6: Distribution of node's truss number with respect to the ranking of the nodes under their spreading properties. The nodes are classified in groups (different colors) depending on their truss number; for each window size W, we plot the distribution of truss numbers observed within it. Observe that, for small window sizes a large number of the nodes belong to the first group, i.e., their truss number is K_{max}. When the window is enlarged, the groups of lower truss numbers involve a large percentage of the considered nodes.

nodes with truss number in the range $K_{\max-5}$ to $K_{\max-9}$. Note that, the last group may contain less than five truss numbers.

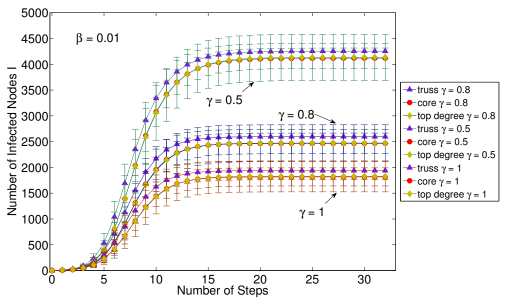
Figure 3.6 depicts the distribution of truss numbers for various values of window W. The colors on each bar correspond to the groups of truss number (darker colors for truss numbers closer to the maximum one). As we can observe in most of the datasets, for small values of window W, a large number of the nodes belong to the first group, i.e., their truss number is the maximum one. Since in most of the cases only a tiny fraction of the nodes of the graph belong to the very first groups (i.e., close to $K_{\max}$), even for small window sizes we also observe nodes from groups that correspond to smaller truss numbers. As the window W increases, i.e., deviate from the optimal spreading behavior, groups of smaller truss numbers start to evolve. From these results, it is evident that the truss number is related to the spreading capabilities of the nodes. Until now, we had only examined the effect of the nodes that belong to the maximal K-truss subgraph. However, from this experiment we can conclude that, in general, nodes with high truss number tend to have good spreading properties – with the truss number being highly related to the spreading effect.

3.4.3 *Impact of infection and recovery rate on the spreading process*

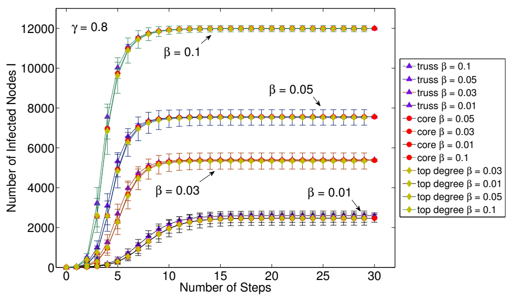
In the experiments that have been already presented, parameters β and γ of the SIR model have been set to some constant values; the infection rate β is typically set close to the epidemic threshold of the graph (as defined by the maximum eigenvalue of the adjacency matrix of the graph), while the recovery rate is considered constant and always set to $\gamma = 0.8$. Here, we examine the impact of the infection and recovery rate on the epidemic spreading achieved by the proposed method (**truss**) and the two baseline methods (**core** and **top degree**). To that end, we simulate the spreading process for each of the above methods, setting parameters β and γ as follows:

(i) Parameter β is set close to the epidemic threshold of the graph, while varying parameter $\gamma \in \{0.5, 0.8, 1\}$. Parameter $\gamma = 1$ implies that each infected node moves to the recovered (R) state with probability one, in the next step of the model.

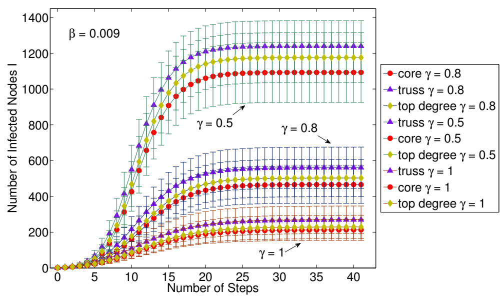
(ii) The recovery rate is set to $\gamma = 0.8$, while considering different values of parameter β, always above the epidemic threshold of the graph. As we discussed in the main text,



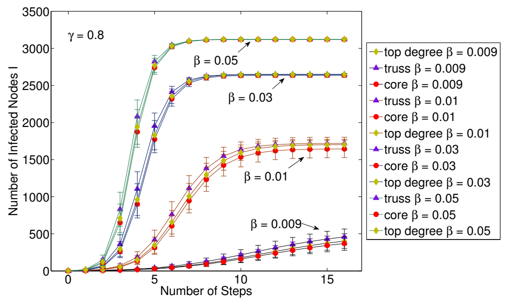
(a) EMAIL-ENRON: $\beta = 0.01$



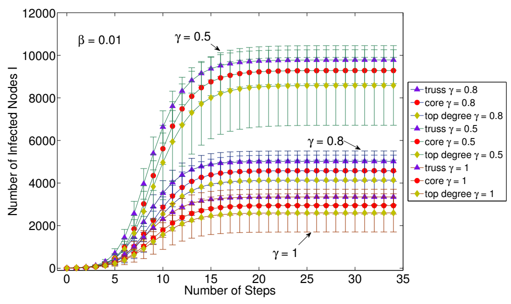
(b) EMAIL-ENRON: $\gamma = 0.8$



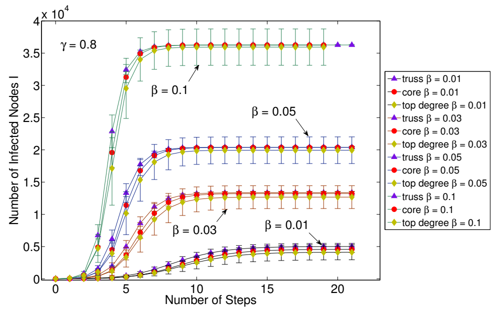
(c) WIKI-VOTE: $\beta = 0.009$



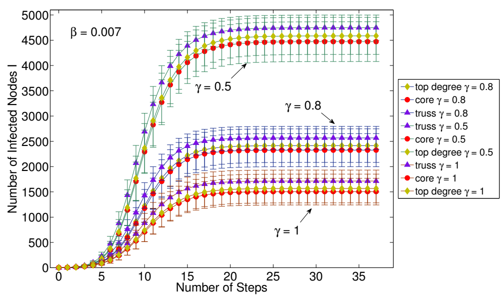
(d) WIKI-VOTE: $\gamma = 0.8$



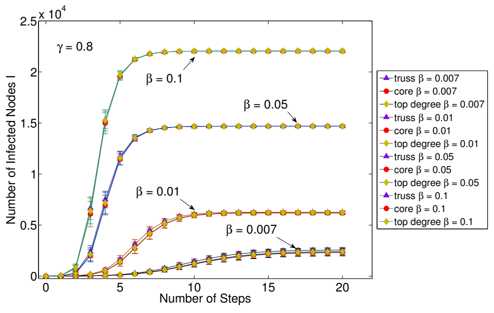
(e) EMAIL-EUALL: $\beta = 0.01$



(f) EMAIL-EUALL: $\gamma = 0.8$



(g) EPINIONS: $\beta = 0.007$



(i) EPINIONS: $\gamma = 0.8$

Figure 3.7: Impact of infection and recovery probabilities of the SIR model on the spreading process: (i) parameter β is set close to the epidemic threshold of the graph, while varying parameter $\gamma \in \{0.5, 0.8, 1\}$; (ii) setting parameter $\gamma = 0.8$ and considering different values of parameter β (always above the epidemic threshold of the graph).

Network	**Nodes**	**Edges**	d_{max}	k_{max}	K_{max}
EmailEnron	33,696	180,811	1383	43	22
Epinions	75,877	405,739	3044	67	33
Higgs	456,626	14,855,842	51386	125	72

Table 3.1: Network datasets used in Section 3.5.

if we consider high values of the infection rate β, a relatively high fraction of nodes will be infected, and thus, the spreading capabilities of individual nodes is diminished.

Fig. 3.7 shows the results. In all cases, we have computed the cumulative fraction of infected nodes I_t per step of the process, for each of the three methods, along with the standard deviation (depicted as error bars in the plot). As we can observe, while the recovery probability γ decreases, the number of infected nodes increases both during the first time steps of the process, as well as at the end of the epidemic. This behavior is expected since, as we discussed above, with high recovery rate γ most of the nodes will move to the R state, thus being inactive in subsequent iterations of the model. Regarding the performance of the methods, it is evident that the proposed **truss** outperforms both baselines for all different settings of parameter γ.

In the second case where the recovery rate γ is constant, while the infection probability is increasing, the number of infected nodes naturally increases. However, for higher values of β, the total number of infected nodes is almost the same for all methods. This behavior is rather expected; by increasing the infection rate, the importance of individual nodes in the epidemic process is reduced. For these values of β, the difference between the methods can be observed during the outbreak of the epidemic (i.e., first steps of the process), where the **truss** method performs qualitatively better.

3.5 EXPLORATION OF NETWORK CENTRALITIES IN SPREADING PROCESSES

In this Section we explore the centralities of the entities that are involved in a spreading process which is triggered by different groups of influential spreaders of a network. We analyze the patterns that occur by simulating the spreading process with the SIR epidemic model [130]. We also compare the simulated diffusion process with real influence, in terms of the evolution of the centralities of the infected nodes.

DATASETS We have performed experiments with the following real-world networks: EmailEnron, Epinions and Higgs. All graphs are considered undirected and unweighted. High level characteristics of the networks are shown in Table 3.1. A more detailed description of the datasets is presented in Section 2.4 of Chapter 2.

DISTRIBUTION OF THE EXAMINED CENTRALITIES We have examined the distribution of the node degree (d_v)), core number (c_v) and truss number (t_v) of these networks and the results for the Epinions dataset are depicted in Figure 3.1. The rest of the networks show a similar behavior. The plot shows the complementary cumulative distribution function of the nodes' aforementioned centralities in log-log scale. We observe that all three distributions are skewed, indicating that few nodes have high centralities and the majority of them have "low" degree and participate in "low" k-core and K-truss subgraphs.

METHODOLOGY To simulate the spreading process, we use the SIR model. Initially, we set a single node to be at the infected state I and the rest of the nodes at the susceptible state S. We set the parameter β close to the epidemic threshold and the parameter $\gamma = 0.8$ same as in Section 3.4.1. In our experiment, we compare three node centralities:

(i) degree (d_v),

(ii) core number (c_v) and

(iii) truss number (t_v).

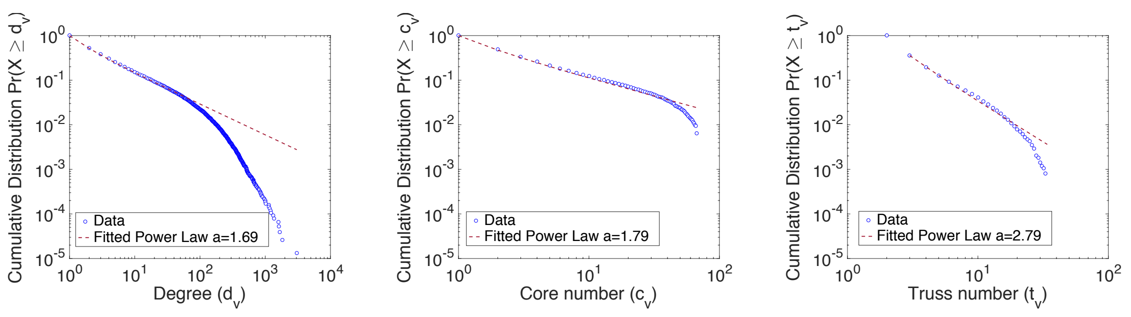
Those are the centralities of the nodes that are being infected at every time step of the process while the epidemic was triggered from three different groups of nodes:

(a) **group** D denoting the set of nodes with the highest degree in the graph,

(b) **group** C' denoting the set of nodes belonging to the k-core excluding those that belong to the K-truss of the graph,

(c) **group** T denoting the set of nodes with the maximum node truss number.

For every node of the group, we simulate the process 100 times and get the average behavior of the node. In order to get the average behavior of all the nodes of each group, we repeat the above for all respective nodes.



(a) DEGREE d_v (b) CORE NUMBER c_v (c) TRUSS NUMBER t_v

Figure 3.1: Complementary cumulative distribution function of nodes' (a) DEGREE d_v, (b) CORE NUMBER c_v and (c) TRUSS NUMBER t_v of the EPINIONS dataset in log-log scale. The red line corresponds to the fitted power-law distribution.

3.5.1 *Evaluation of Results*

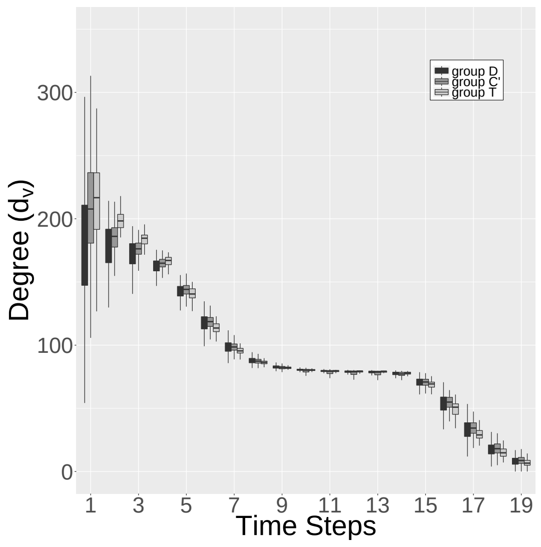
The results from the experiments are depicted in Figure 3.2. We can observe that the behavior of all three centralities is divided in three distinctive periods:

i) the outburst of the epidemic,

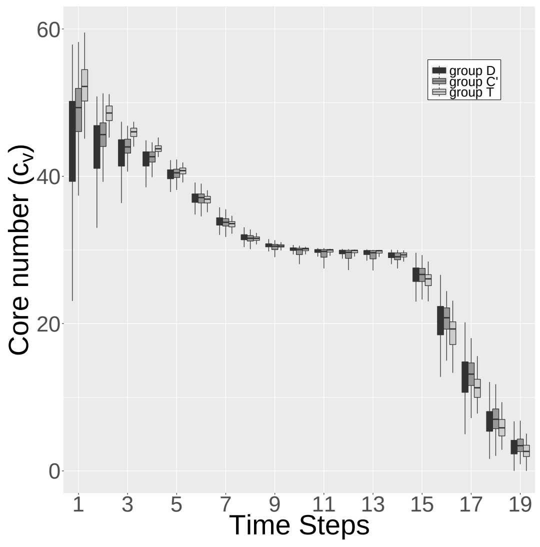
ii) the "plateau" period and

iii) the fadeout of the epidemic.

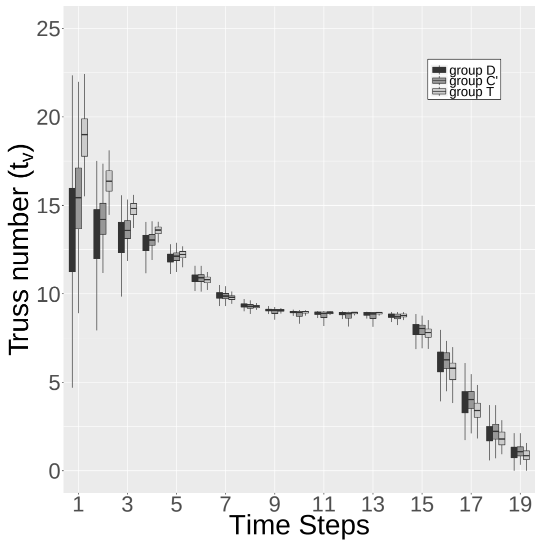
We can observe that in case of the Epinions dataset, nodes originating from group T, achieve to influence on average nodes with higher degree, core and truss centralities during the outburst of the epidemic – specifically during the first four steps. In case of the Email-Enron dataset, group T and C' seem to influence nodes with similar centralities during the first timesteps. In both datasets though, the superiority of the latter groups compared to group D is easily recognizable during the first period of the spreading process. After the outburst of the epidemic (after the 6th time step), we observe in both datasets that nodes being infected are characterized by similar centralities for all the three compared behaviors. It should be noted that the centralities of the nodes infected during this "plateau" period are quite high considering the fact that most of the nodes of the network are characterized by low centralities. We realize that most of the nodes infected in all cases during such an epidemic are characterized by the centralities observed during the "plateau" period. For example for the EPINIONS dataset, from the 8th until the 14th timestep, we observe that nodes being infected have a degree ranging from 77 to 87, a core number ranging from 29 to 31 and a truss number ranging from 8 to 9. Finally, during the fadeout (during the 5 last time steps), the



(a) DEGREE d_v



(b) CORE NUMBER c_v



(c) TRUSS NUMBER t_v

Figure 3.2: Evolution of the infected nodes' average (a) DEGREE d_v, (b) CORE NUMBER c_v and (c) TRUSS NUMBER t_v during a simulated spreading process using the SIR model for the EPINIONS dataset having triggered the epidemic from nodes of sets D, C′ and T.

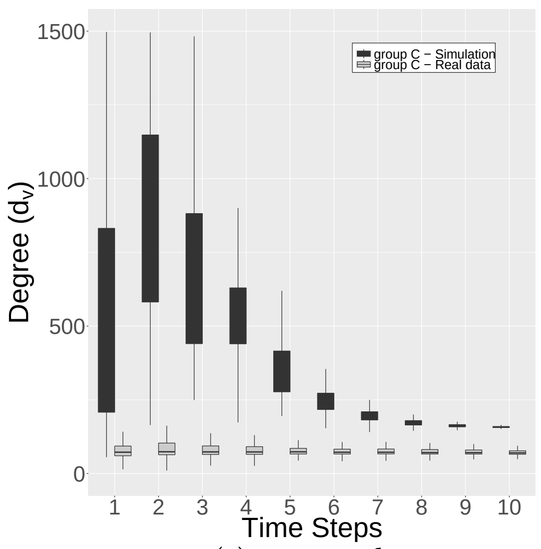
centralities of the nodes infected are severely decreased in all cases. Note that the process stops when no more nodes get infected (for the Epinions dataset this happens at time step 19).

COMPARISON TO A REAL SPREADING PROCESS. In order to explore the information spreading in a real world setting we have used the Higgs Twitter dataset [53]. The dataset is built by studying the diffusion (in means of tweets) of the announcement of the Higgs boson-like particle at CERN on the Twitter social network between the 1st and 7th of July 2012. The interactions that were considered were retweets, mentions and replies.

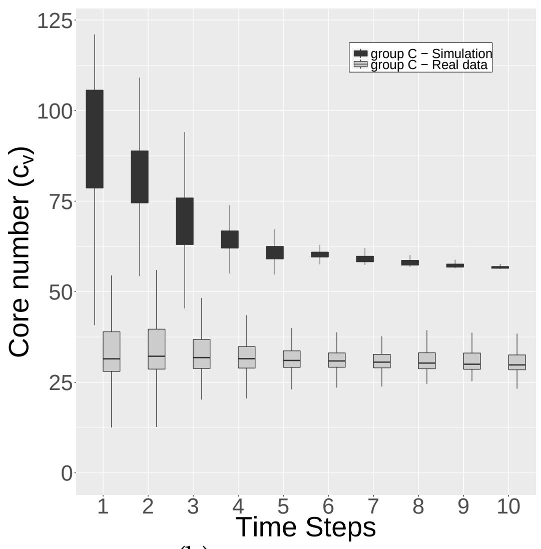
In order to fairly compare the specific spreading process with the simulation models that were previously discussed, specific assumptions have to be made [137, 138]. The spreading activity that is recorded, involves around 562,556 asynchronous timestamps during which at least one spreading interaction is recorded. We have decided to study the influence that is triggered from nodes belonging to group C (i.e., the totality of the nodes participating in the maximum k-core subgraph of the network) as they have been proven to represent a great percentage of the spreading activity in a network. The timestamp where each of the respective nodes is firstly influenced by a user of its network is considered as the first timestep of the specific node's spreading activity. The following timestep is considered after 5000 consecutively recorded timestamps. We are considering the nodes being influenced during every such period by all the nodes that were influenced during the preceding periods. We have considered for our experiments totally ten such periods which we will be refering to as timesteps.

We are specifically interested in the three centralities of those nodes that are being infected during these timesteps which we have compared with the respective centralities after running the SIR model for ten timesteps starting from the same C nodes. As in our previous experiments, the process is simulated 10 times for every node of the group (due the dataset's size) and the average behavior of the node is calculated. The above is repeated for all the nodes of the C set. Results from the experiments are shown in Figure 3.3.

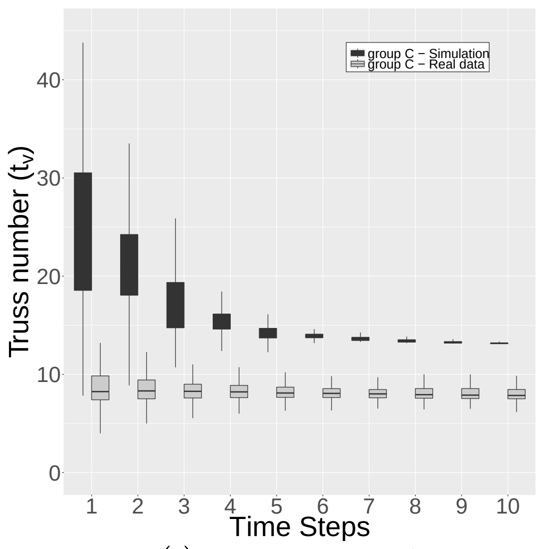
We observe that there are great differences between the two settings. While the simulation shows that during the first steps, nodes with high centralities are influenced, real data show that the nodes that are influenced do not differ much in terms of centralities during these 10 time steps that we study. It has indeed been proven that epidemic models fail to reproduce the realistic viral spreading pattern [138] in terms of i) number of



(a) DEGREE d_v



(b) CORE NUMBER c_v



(c) TRUSS NUMBER t_v

Figure 3.3: Comparison of the evolution of the infected nodes' average (a) DEGREE d_v, (b) CORE NUMBER c_v and (c) TRUSS NUMBER t_v, between a simulated spreading process using the SIR model and real influence data for the HIGGS-TWITTER dataset having triggered the epidemic from nodes of set C.

nodes being infected and ii) of the characteristics of the diffusion trees created during the process.

We prove that the model also fails to indicate the centrality characteristics of the nodes being infected during the process. This can be explained by the definition of the models. First and foremost the probability of an entity influencing a neighboring entity shouldn't be the same for all entity relations. There is an extensive literature proposing methods to modelize users' influence [80, 103, 122]. Moreover, considering the SIR model, an entity does not get "recovered" while in a spreading process such as an information diffusion in a Twitter network.

User behavior contains more complex patterns concerning the way information is disseminated. Users may stop diffusing information for some period of time but start "spreading the word" again in a later period for indefinite reasons. This resembles the SIS model where infected nodes can return to the susceptible state and with a probability can start again infecting their neighbors. But unfortunately, neither this model can be compared with the real influence data of our study. While the latter may be extremely hard to model, we believe that there exist "who-influences-whom" patterns in influence data that can help towards a better definition of the probabilities of an entity influencing a fellow neighbor. Those patterns can be found while exploring the aforementioned centralities of entities influencing their peers between steps of the spreading process.

3.6 CONCLUSIONS AND FUTURE WORK

Understanding and controlling the mechanisms that govern spreading processes in complex networks is a fundamental task in various domains, including disease propagation and viral marketing. Central to these tasks is the problem of identification of influential nodes with good spreading properties, that are able to diffuse information to a large part of the network. It has been empirically observed that widely used node centrality criteria such as degree and betweenness, have drawbacks when applied to find nodes with good spreading properties; a node may have a large number of neighbors but if it is located to the periphery of the network, its spreading capability is reduced. Kitsak et al. [97] applied the k-core decomposition method in order to locate centrally placed individuals with good spreading properties; their observations suggested that the identified nodes outperform previously used criteria with respect to the spreading effectiveness. However, the main drawback of the k-core decomposition is that its resolution is quite coarse. Depending on the structure of the network, many nodes will be assigned the same k-core number, even if their spreading capability differs from each other.

The fact that a relatively large fraction of the nodes that are extracted by the k-core decomposition method corresponds to highly influential nodes, was the motivating force behind our approach. To deal with this issue, we have considered the K-truss decomposition of a network – a triangle-based extension of the k-core structure. By setting a more strict criterion upon which nodes are assigned into layers of the graph, we have shown that the K-truss decomposition can effectively reduce the number of candidate influential spreaders in the network, as it further refines the set of nodes belonging to the maximal k-core subgraph (recall that the maximal K-truss is a subgraph of the maximal k-core). Using the SIR epidemic model, we have shown that such spreaders have the ability to influence a greater part of the network during the first steps of the process; also the total fraction of influenced nodes at the end of the epidemic is higher, compared to the performance of the rest nodes that belong to the maximal k-core subgraph and the top degree nodes of the network. Our experimental results also indicate that the K-truss decomposition filters out the best spreaders of the k-core structure; the spreading effectiveness of the remaining nodes is weakened, and those nodes show even worst behavior compared to the top degree ones.

Additionally we observed that those nodes belonging to the maximal K-truss subgraph are distributed well among the optimal spreaders of the graph, presenting better behavior compared to the remaining nodes of the maximal k-core subgraph. Furthermore, we observed that the truss number in general, is closely related to the spreading effect. The nodes of the network are distributed among the optimal spreaders (after ranking) in a way that a relationship to truss number occurs.

Finally, we explored the centralities of the entities that are involved in a spreading process which is triggered by different groups of influential spreaders of a network. We obtain interesting results by simulating the spreading process with the SIR epidemic model that let us conclude that i) degeneracy algorithms help us detect groups of nodes that will influence nodes with high centralities during the outburst of the epidemic and ii) there exists a "plateau" period during the spreading process where a significant part of the nodes are influenced and iii) the nodes influenced in this "plateau" period have relatively high degree, core and truss centralities considering the respective centrality distributions of the network. Finally, by comparing the simulated diffusion process with real influence, we observe that epidemic models cannot reproduce the real diffusion in terms of the evolution of the centralities of the infected nodes. Thus we conclude that a further research direction could be the search for a diffusion model fitting the real world process.

4

INFLUENCE MAXIMIZATION IN SOCIAL NETWORKS

INFLUENCE MAXIMIZATION has attracted a lot of attention due to its numerous applications, including diffusion of social movements, the spread of news, viral marketing and outbreak of diseases. The objective is to discover a group of users (i.e., nodes) that are able to maximize the spread of influence across the network. It constitutes an NP-hard problem, for which a simple greedy algorithm provides good approximation guarantees. Nevertheless, there are obviously serious scalability concerns. In this paper, we propose Matrix Influence, MATI, an efficient algorithm that can be used under both the Linear Threshold and Independent Cascade diffusion models. MATI is based on the precalculation of the influence by taking advantage of the simple paths in the node's neighborhood. An extensive empirical analysis has been performed on multiple real-world datasets and showing that MATI has competitive performance when compared to other well-known algorithms with regards to running time and expected influence spread.

4.1 INTRODUCTION

The interest in influence propagation has been exponentially increasing the recent years, with applications ranging from social media analytics [10] and adoption of innovations, to personalised recommendations [155], identification of influential tweeters [8] and viral marketing [14, 32, 58, 72, 73, 118]. The way entities in a social network interact with each other creates information pathways in the network, making their position be critical towards their spreading capabilities in the network. Thus, an important aspect on understanding the influence dynamics is the identification of privileged users that can diffuse information to the greatest possible part of the network. Assuming we are aware of the extent to which each individual can influence one another in a social network and that we would like to introduce a new product so that it is adopted by the largest possible fraction of the network [58, 118]. The question is how to choose those specific individuals that will trigger a cascade where their friends will recommend the product to their friends until a lot of individuals will try it.

In this Chapter we will be studying the specific problem that concerns the identification of a group of nodes that are able to maximize the total spread of influence – usually called the *influence maximization* problem [40, 41, 47, 77, 78, 93, 107]. Influence maximization can formally be described as follows: given a *social network* where the relations among users are revealed, a *diffusion model* that simulates how information propagates through the network and a parameter k, the goal is to locate those k users (represented as nodes in the graph) that maximize the spread of influence. Kempe et al. [93] formulated the problem in the aforementioned manner while adopting two diffusion models borrowed from mathematical sociology: the *Linear Threshold* (LT) and the *Independent Cascade* (IC) model. According to both, at any discrete time step a user can be either active or inactive (i.e., has adopted the product or not) and the information propagates until no more users can be activated.

Kempe et al. [93] proved that the function of the influence spread under both LT and IC models is monotone and submodular. By exploiting these properties, they presented a greedy algorithm that achieves $(1 - 1/e)$ approximation ratio. However, as the greedy algorithm repeatedly selects in every iteration the node with the maximum marginal gain by running Monte Carlo simulations, we are lead towards great performance downsides. Indeed, Feige's [63] result implied that any algorithm that guarantees a solution of at least $(1 - 1/e + \epsilon)$ times the optimum, will probably not scale with the number of seeds.

The main contributions of this work can be summarized as follows:

- *Efficient influence maximization algorithm:* We propose the Matrix Influence (MATI) algorithm, an efficient influence maximization algorithm under the two well-known diffusion models in the field: the linear threshold (LT) and independent cascade (IC) models.

- *Evaluation of our proposed approach on real graphs:* We used large scale real-world graphs while performing our experiments and showed that for both the cases of the LT and IC models, MATI performs better than the baseline methods both in terms of influence and computation time.

The rest of this Chapter is structured as follows: In Section 4.2 we present the influence maximization problem along with the diffusion models used in the field. Section 4.3 reviews the related literature on the problem of influence maximization. In Section 4.4 we describe the MATI algorithm for the linear threshold (LT) and the independent cascade (IC) diffusion models.

Notation	Description
$p_{u,v}$	Influence weight on directed edge (u,v)
$\sigma(S)$	Influence of a set of nodes S to the graph
$\mathscr{A}(u,v)$	Influence of node u to node v
$\Omega(u,v)$	Forward cumulative influence of node u to node v
$\mathscr{T}(u) = \{\tau_1, \tau_2, \dots, \tau_M\}$	Set of all possible paths starting from node u
$\tau_i = \{n_{i1}, n_{i2}, \dots, n_{iN}\}$	Path consisting of N nodes starting from node u
$\mathscr{F}(\tau_i) = \{f_{i1}, f_{i2}, \dots, f_{iN}\}$	Cumulative probability path for path τ_i
$p^{\tau}_{\ell,\ell+1}$	Influence weight between successive nodes in τ
$\Psi(u,v) = \{\psi_1, \psi_2, \dots, \psi_L\}$	Set of all possible paths between nodes u and v
$\psi_i = \{n_{i1}, n_{i2}, \dots, n_{iN}\}$	Path between nodes u and v
$\Phi(\psi_i) = \{\phi_{i1}, \phi_{i2}, \dots, \phi_{iN}\}$	Cumulative probability path for path ψ_i

Table 4.1: List of symbols used in the Chapter.

In Section 4.5 we show the results conducted in real-world datasets and in Section 4.6 we present concluding remarks.

4.2 PRELIMINARIES AND BACKGROUND

In this Section we present the problem of *Influence Maximization (IM)* as well as a description of two well-known diffusion models used in the field: the Linear Threshold (LT) and Independent Cascade (IC) models and some further extensions. A list of the symbols used in the Chapter is presented in Table 4.1.

4.2.1 *The Influence Maximization (IM) problem*

Let us define a network as $G = (V, E)$ with V being the set of nodes and E the set of edges between those nodes. If there exists an influence function $f(S) : S \to \mathbb{R}$ with $S \subseteq V$, then the problem of influence maximization (IM) is to discover this subset S that maximizes f. S has a given size k with $k \ll |V|$.

Kempet, Kleinberg and Tardos [93] proposed a general framework for IM. The spreading models they consider (see Section 4.2.2) categorize every node in two states: the *active* and the *inactive* state. Initially a set A of active nodes is considered. The latter trigger a spreading process at the end of which, $f(A)$ nodes will be active. Such a number cannot be obtained analytically, an estimation though can be given after extensive simulations of the process. In this framework, IM requires a set A of k nodes that maximizes $f(A)$.

With the IM problem being NP-hard [93], the majority of the literature provides approximate rather than the exact solution. The most common approximate algorithms would be: i) *heuristic* and ii) *greedy* algorithms. An example of a heuristic algorithm would be to rank all the nodes according to a centrality measure and select the k top-ranking nodes.

The earliest greedy algorithm was provided by Kempe, Kleinberg and Tardos [93] (see Algorithm 4.1). Let S be the subset of vertices selected to initiate the influence propagation, called the *seed set*. Let *InfModel*(S) denote a model that simulates a spreading process triggered by set S of which the output is a set of vertices influenced by S. At each round i, one node is added to set S by the algorithm. This node, together with the current set S is the one maximizing the total influence spread (Line 10). To select such a node, for each $v \in S$ the influence of $S \cup v$ is estimated by performing R simulations of *InfModel*$(S \cup v)$. Typically the number of simulations is set to 10.000.

Algorithm 4.1 Greedy Algorithm

```
1:  Input: $G = (V, E)$, $k$ and $R$        ▷ $k$: budget, $R$ : #simulations
2:  Output: $S$
3:  Initialize: $S = \emptyset$ and $R = 10.000$
4:  for $i = 1$ to $k$ do
5:      for each $v \in V \setminus S$ do
6:          $\sigma(v) = 0$
7:          for $j = 1$ to $R$ do
8:              $\sigma(v) += |InfModel(S \cup v)|$
9:          $\sigma(v) = \sigma(v)/R$
10:     $S = S \cup \{argmax_{v \in V \setminus S}\{\sigma(v)\}$
11: return $S$
```

Before presenting the approximation guarantee of the greedy algorithm, we should first introduce the definition of a submodular function.

Definition 4.1. *(Submodular function)*

Given a finite ground set V, a set function $f : 2^V \to \mathbb{R}$ is submodular if:

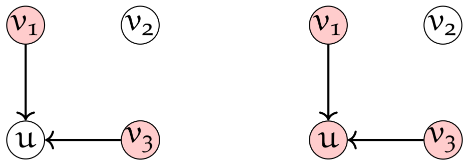


Figure 4.1: Illustration of the Linear Threshold model. The white colored nodes are in an inactive state whereas the pink colored nodes are in an active state. Node u will be activated if $p_{v_1,u}+p_{v_3,u} \geqslant \theta_u$.

$$f(S \cup v) - f(S) \geqslant f(T \cup v) - f(T)$$

for all elements $v \in V \setminus T$ *and all pairs of sets* $S \subseteq T$.

If f is monotone (i.e., $f(S \cup v)f(S)$) for all elements v and sets S, then it has been shown [51, 129] that the greedy algorithm provides an approximate solution S^* within a factor $1 - 1/e \approx 0.63$:

$$f(S) \geqslant (1 - \tfrac{1}{e})f(S^*)$$

4.2.2 *Diffusion Models*

Diffusion models are used to simulate the process of information propagation in the network. Next, we describe two of the most widely applied models, namely the LT and IC models and some extensions. As we will present later on, the proposed algorithm is designed to deal with both of those models – contrary to most of the state-of-the-art algorithms which have been designed for one of the two models.

4.2.2.1 *Linear Threshold (LT) model*

In this model, each directed link $u \to v$ is assigned a weight $p_{u,v}$ satisfying that $\sum_{u \in Nb(v)} p_{u,v} \leqslant 1$, where $Nb(v)$ is the set of node v's neighbors. Notice that $p_{u,v} \neq p_{v,u}$. Each node u chooses a threshold θ_u uniformly at random from the interval $[0, 1]$ which represents the weighted fraction of u's neighbors that must become active in order for u to become active (an example is provided in Fig 4.1). The linear threshold model starts with some active nodes (with all other nodes being inactive) and a random choice of thresholds. Then at each time step, a node v will become active if $\sum_{v \in Nb^*(u)} p_{v,u} \geqslant \theta_u$, with $Nb^*(u)$ being the set of u's active neighbors. The diffusion process unfolds in a synchronous and deterministic way in discrete steps until no further changes of nodes' states happen.

4.2.2.2 *Independent Cascade (IC) model*

In the IC model, when a node u first becomes active in timestep t, it is given a single chance to activate each neighbor v – which is currently inactive – and succeeds with a probability $p_{u,v}$. If u succeeds, then v will become active in the next timestep. If u does not succeed, it cannot further attempt to activate v in future timesteps. The process continues as long as node activations are possible.

Kempe et al. [93] proved that for both the aforementioned models, the objective functions on the expected number of active nodes $f(.)$ are submodular concluding that the greedy algorithm provides a $(1 - 1/e)$-approximation to the problem of influence maximization.

4.2.2.3 *Further extensions*

In the IC model, each weight $p_{u,v}$ that concerns link $u \to v$ is independent of the spreading process history. Nevertheless social networks have shown that information diffusion presents memory effects [35, 56, 101, 113]. Thus, Kempe et al. [94] presented the *derceasing cascade model* which extends the IC model by assigning weights at links that depend on history. Let us denote as S the set of node v's neighbors that already attempted to activate v. Then $p_{u,v}(S)$ expresses u's success probability to activate v. The model contains two constraints: i) *order-independent*: the order in which the attempts of the nodes trying to activate node v does not affect the probability of the latter being active at the end; ii) *non-increasing*: function $p_{u,v}(S)$ satisfies $p_{u,v}(S) \geqslant p_{u,v}(T)$ with $S \subseteq T$. The authors proved that the objective function $f(.)$ for the *derceasing cascade model* is also submodular concluding that the greedy algorithm provides a $(1 - 1/e)$-approximation.

4.3 RELATED WORK

Following the seminal work by Kempe et al. [93], a series of algorithms have been proposed in order to: (i) reduce the number of influence spread evaluations, (ii) make batch computations of the influence spread, and (iii) design scalable heuristics towards computing the respective spread.

Leskovec et al. [107] proposed the CELF algorithm, based on a "lazy-forward" optimization scheme, that finds near-optimal solutions guaranteed to achieve at least $1/2(1 - 1/e)$ of the optimal ones by being 700 faster than the greedy algorithm. The fact that the marginal gain of a node cannot be greater than the one achieved in previous iterations is taken into account.

A table which stores every node and its marginal gain (with regards to the influence achieved by the so far selected candidates) is stored in decreasing order. Only the marginal gain of the most profitable (top) node is re-evaluated when it is needed and the table is sorted again. The node remaining at the top of the table is selected as the next seed node. Goyal et al. [78] introduced an algorithm that further optimized the aforementioned one by $35 - 55\%$, called CELF++. By exploiting the submodularity property of the spread function for the diffusion models (e.g., LT and IC) it avoids the unnecessary re-computations of the marginal gains.

Chen et al. [41] proposed the LDAG algorithm which is tailored for the LT model and achieves to produce results by being orders of magnitude faster than the greedy algorithm. They firstly show that the computation of influence in directed acyclic graphs (DAGs) can be done in linear time. Based on that, they construct a local DAG for every node of the network and restrict the influence of the node in this local area. After constructing the DAGs the greedy seed selection approach is applied together with an accelerated solution for updating the incremental influence spread of each node.

The MIA algorithm, proposed by Chen et al. [40], is a maximum influence arborescence model based on the assumption that information diffusion occurs according to the IC model. They succeed in proposing a scalable algorithm which produces results that outperform the other so far proposed heuristics by $100 - 120\%$. They propose a) a best-effort algorithm that estimates the upper bounds of location-aware influence spread and prunes users having small influences thus achieving an approximation ratio of $(1 - 1/e)$ and b) a topic-materialization-based algorithm that estimates the bounds of influence spread and avoids computation of the actual influence of least influential users while finally achieving an approximation of $\epsilon(1 - 1/e)$. Goyal et al. [77] proposed SimPath, an algorithm tailored for the LT model which computes the influence spread by enumerating simple paths within a small neighborhood. With the help of a parameter, a balance between running time and quality of the solution can be achieved.

Tang et al. [157] designed the Two-phase Influence Maximization (TIM) algorithm which runs in near-linear time while also returning $(1 - 1/e - 1/\epsilon)$-approximate solutions. In the first phase, a lower-bound of the maximum spread is calculated in order to derive a parameter θ. In the second phase, the parameter θ is used so that random reverse reachable (RR) sets (as defined by Borg et al. [29]) are sampled. The k-sized node set that covers a large number of RR sets is the final result. TIM supports the *triggering model* which is a general diffusion model incorporating both the LT and IC models.

Another interesting approach is the one by Cohen et al. [47]. They designed a SKetch-based Influence Maximization (SKIM) algorithm which uses per-node summary structures called *combined reachability sketches* representing the node's influence coverage [46]. They introduce *influence oracles* which can answer *influence queries* in an efficient way. This algorithm is designed based on the IC diffusion model. Goyal et al. [76] proposed a new probability model, the *credit distribution model* which directly estimates influence spread by exploiting historical data. This makes the need for knowing the influence probabilities and making Monte Carlo simulations to compute the respective influence redundant thus avoiding costly computations.

4.4 MATRIX INFLUENCE (MATI) ALGORITHM

In this Section we introduce the proposed MATI algorithm, under both the LT and IC models. After presenting some preliminaries on calculating influence in social networks, we describe in detail how we compute the influence under each model and we provide the algorithms used. Similar to the case of the greedy algorithm [93], at each round of MATI, the node with the largest marginal influence estimate is chosen as the next candidate. The novelty of our algorithm lies on the fact that, by having pre-calculated all possible paths between nodes along with the respective influences of the nodes -acting individually- in the network, we are able to efficiently compute the marginal gain of adding a candidate node.

4.4.1 *Influence in Social Networks*

A social network is typically modeled as a directed graph $G = (V, E)$, consisting of $|V|$ users represented as nodes and $|E|$ edges reflecting the relationship between users. An influence weight $p_{u,v} \in [0, 1]$ is also associated with each directed edge $(u, v) \in E$, and represents the probability of node u to influence node v.

We assume that $\mathscr{T}(u) = \{\tau_1, \tau_2, \ldots, \tau_M\}$ represents the set of all possible paths that exist in the graph starting from node u and leading to "leaf" nodes (i.e., the paths that cannot be further extended). It should be noticed that in our paths no nodes are allowed to be repeated. All these paths are generated by using the Depth-first search (DFS) algorithm, with the node to be the root of the *tree*. τ_i represents each possible path and M is the number of all possible paths which start from node u. Each path τ_i consists of a sequence of nodes: $\tau_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$

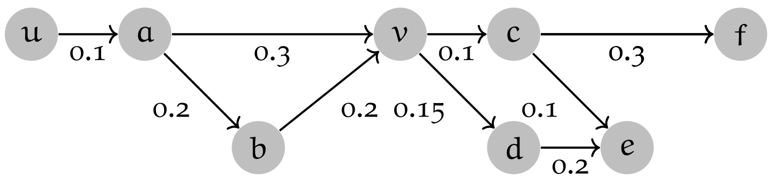


Figure 4.1: Example graph.

and N represents the number of the nodes and simultaneously the index of the terminal node of path τ_i. M and N can obviously be different for every user u and every path τ_i, respectively, but they are defined as such for the sake of the simplicity of the model.

Let $p^{\tau}_{\ell,\ell+1}$, $1 \leqslant \ell \leqslant N-1$, represent the influence weight (probability) between two successive nodes ($n_{i\ell}$ and $n_{i(\ell+1)}$) in path τ. Then $\mathscr{F}(\tau_i) = \{f_{i1}, f_{i2}, \dots, f_{iN}\}$ represents the probability path for every path τ_i starting from node u to be *a*ctive (i.e., a path is considered active if each one of its edges is active). Each f_{ij} is defined as follows:

$$f_{ij} = \begin{cases} \prod_{\ell=1}^{j-1} p^{\tau_i}_{\ell,\ell+1} & \text{if} \quad j > 1, \\ 1 & \text{otherwise.} \end{cases} \tag{4.1}$$

Let us now define as $\Psi(u,v) = \{\psi_1, \psi_2, \dots, \psi_L\}$ the set of all possible (unique) paths from a node u to a node v. ψ_i represents each possible path and L is the number of all possible paths between nodes u and v. Each path ψ_i consists of a sequence of nodes: $\psi_i = \{n_{i1}, n_{i2}, \dots, n_{iN}\}$ and N represents again the number of nodes of path ψ_i. Obviously $L \leqslant M$ and again L and N can be different for every set of paths between two nodes and every path ψ_i, respectively. We can now respectively define as $\Phi(\psi_i) = \{\phi_{i1}, \phi_{i2}, \dots, \phi_{iN}\}$ the probability for every path ψ_i between two nodes u and v and is calculated in the same way as f_{ij} (see Eq. (4.1)).

Let us illustrate the above notations with an example. For this purpose, we consider the graph illustrated in Fig. 4.1 that consists of $|V|= 8$ nodes and $|E|= 9$ edges. The set of paths starting from node u is defined as $\mathscr{T}(u) = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$ with $M = 6$ and the different paths starting from node u being the following:

$$\begin{aligned} \tau_1 &= \{u, a, v, c, f\}, & N &= 5 \\ \tau_2 &= \{u, a, v, c, e\}, & N &= 5 \\ \tau_3 &= \{u, a, v, d, e\}, & N &= 5 \\ \tau_4 &= \{u, a, b, v, c, f\}, & N &= 6 \\ \tau_5 &= \{u, a, b, v, c, e\}, & N &= 6 \\ \tau_6 &= \{u, a, b, v, d, e\}, & N &= 6 \end{aligned}$$

Therefore, the probability path for τ_1 is defined as: $\mathscr{F}(\tau_1) = \{f_{11}, f_{12}, f_{13}, f_{14}, f_{15}\}$, where $f_{11} = 1$, $f_{12} = 0.1$, $f_{13} = 0.03$, $f_{14} = 0.003$ and $f_{15} = 0.0009$.

In the same way, the set of all possible paths from node u to node v is defined as $\Psi(u, v) = \{\psi_1, \psi_2\}$ with $L = 2$ and the different paths between the two nodes being the following:

$$\begin{aligned} \psi_1 &= \{u, a, v\}, \qquad N = 3 \\ \psi_2 &= \{u, a, b, v\}, \quad N = 4. \end{aligned}$$

Then the probability path for ψ_1 is defined as: $\Phi(\psi_1) = \{\phi_{11}, \phi_{12}, \phi_{13}\}$ with $\phi_{11} = 1$, $\phi_{12} = 1 * 0.1 = 0.1$ and $\phi_{13} = 1 * 0.1 * 0.3 = 0.03$. Similarly, $\Phi(\psi_2) = \{\phi_{21}, \phi_{22}, \phi_{23}, \phi_{24}\}$ with $\phi_{21} = 1$, $\phi_{22} = 1 * 0.1 = 0.1$, $\phi_{23} = 1 * 0.1 * 0.2 = 0.02$ and $\phi_{24} = 1 * 0.1 * 0.2 * 0.2 = 0.004$.

4.4.2 *Influence Computation under the LT Model*

Kempe et al. [93] have shown the equivalence of the Linear Threshold model to the *live-edge model*. According to this model, a node $u \in V$ chooses just one of its incoming edges with probability $p_{u,v}$. If an edge is selected, it is considered *live*, otherwise *blocked*. We can deduce from the above that the nodes expected to be activated by a seed set S is the expected number of nodes that can be reached from S over all possible worlds. As it has been shown by Goyal et al. [77], the expected spread of seed set S can be calculated as follows:

$$\sigma(S) = \sum_{v \in V} \sum_{X} \Pr[X] I(S, v, X) = \sum_{v \in V} \mathscr{A}(S, v), \tag{4.2}$$

where X is a possible *live-edge* graph, $\Pr[X]$ is the sampling probability of graph X, $I(S, v, X)$ is an indicator function which equals to 1 if there exists a live path in X from S to v and 0 otherwise, and $\mathscr{A}(S, v)$ is the probability the single node v to be activated (influenced) by S. In the special case of a single node u, its expected spread to a node v ($u \neq v$) is defined as:

$$\begin{aligned} \mathscr{A}(u, v) &= \sum_{\psi_i \in \Psi(u,v)} \Pr[\psi_i] \\ &= \sum_{\psi_i \in \Psi(u,v)} \prod_{\ell=1}^{N-1} p^{\psi_i}_{\ell,(\ell+1)} \\ &= \sum_{\psi_i \in \Psi(u,v)} \phi_{iN} = \sum_{i=1}^{L} \phi_{iN} \end{aligned} \tag{4.3}$$

where $\Pr[\psi_i]$ is the probability of path ψ_i being *live* and $\Psi(u, v)$ is the set of all possible paths between nodes u and v. It becomes apparent that the influence of a node u to itself is equal to 1 (i.e., $\mathscr{A}(u,u) = 1$). We can now define the expected total influence spread of a single node u to the network:

$$\sigma(u) = \sum_{v \in V} \mathscr{A}(u,v) \approx \sum_{v \in I(u)} \mathscr{A}(u,v), \tag{4.4}$$

where I(u) represents the nodes in the graph that can be influenced by node u depending on a threshold θ which is set in order to limit the calculations of probability and cumulative probability paths. Roughly speaking, a node v belongs to set I(u), *iff* $A(u,v) \geqslant \theta$. Therefore, the lower the parameter value θ is, the higher the accuracy that can be achieved.

The *forward cumulative influence* $\Omega(u,v)$ is another quantity of interest, that corresponds to the influence of node u to the nodes that can be found right after v in the paths $\mathscr{T}(u)$ of node u. It is computed as follows:

$$\Omega(u,v) = \sum_{i=1}^{M} \sum_{j=\text{index}(v)+1}^{N} f_{ij}. \tag{4.5}$$

According to the above, the influence and forward cumulative influence of node u to node v in our example graph (see Fig. 4.1 in Section 4.2) are equal to:

$$\begin{aligned}
\mathscr{A}(u,v) &= \sum_{i=1}^{2} \phi_{iN} = [\phi_{13}]_{\psi_1} + [\phi_{24}]_{\psi_2} = 0.034, \\
{}^{\cdot}(u,v) &= \sum_{i=1}^{6} \sum_{j=\text{index}(v)+1}^{N} f_{ij} \\
&= [f_{14} + f_{15}]_{\tau_1} + [f_{24} + f_{25}]_{\tau_2} + [f_{34} + f_{35}]_{\tau_3} \\
&+ [f_{45} + f_{46}]_{\tau_4} + [f_{55} + f_{56}]_{\tau_5} + [f_{65} + f_{66}]_{\tau_6} = 0.01428.
\end{aligned}$$

Revisiting the case of a set of nodes, Goyal et al. [77] showed that the spread of a set S of nodes is the sum of the spread of each individual node $u \in S$ on the subgraphs induced by the set $V - S + u$:

$$\sigma(S) = \sum_{u \in S} \sigma^{V-S+u}(u), \tag{4.6}$$

where $\sigma^{V-S+u}(u)$ denotes the total influence of u in the subgraph induced by $V - S + u$. Similar to [77], we write $V - S$ to denote the difference of sets V and S, $V \setminus S$, and $V - S + u$ to denote $((V \setminus S) \cup \{u\})$.

By taking advantage of Equations (4.5) and (4.6) we get the following key result that helps towards the calculation of the influence gain after the addition of a node x to a set of nodes S. This result constitutes the basis of the proposed MATI algorithm under the LT diffusion model.

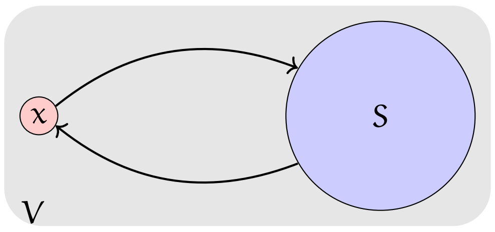


Figure 4.2: Illustration of Theorem 1.

Theorem 1. Under the LT model, to calculate the influence after adding a node x to a set of nodes S, one has to subtract from the sum of the individual spread of S and x the forward cumulative influence ˙ of all the nodes that belong to set S which contain node x in paths connecting the latter to nodes in set S. That is,

$$\sigma(S+x) = \sigma(S) + \sigma(x) - \sum_{y \in S} \Omega(x,y) - \sum_{y \in S} \Omega(y,x). \tag{4.7}$$

Proof.

$$\begin{aligned}
\sigma(S+x) &\overset{(1)}{=} \sum_{u \in S+x} \sigma^{V-S-x+u}(u) \\
&\overset{(2)}{=} \sigma^{V-S}(x) + \sum_{u \in S} \sigma^{V-S-x+u}(u) \\
&\overset{(3)}{=} \sigma^{V-S}(x) + \sigma^{V-x}(S) \\
&\overset{(4)}{=} \sigma(x) + \sigma(S) - \sum_{y \in S} \Omega(x,y) - \sum_{y \in S} \Omega(y,x).
\end{aligned}$$

Equality (1) is a direct application of Eq. 4.6 (see [77] for its proof). (2) and (3) can easily be verified by making simple calculations. (4) comes from set theory (see Fig. 4.2 for an illustration). Roughly speaking, (4) expresses that the influence gain after adding node x to a set S, is equal to the summation of the influence gain of x and S independently, subtracting from this value the nodes that can be influenced by x or S, through paths that pass via nodes on set S or node x, respectively. For instance, if both x and S influence nodes that do not cross each other, we get that $\sigma(S+x) = \sigma(x) + \sigma(S)$. □

Algorithms 4.2 to 4.4 show the complete structure of MATI algorithm under the LT model. Routine CalcStatsLT (Algorithm 4.3) computes $\mathscr{A}$ and Ω, and routine CalcInf (Algorithm 4.4) returns the influence of all nodes $v \in V$, as was described in this section. We use a CELF queue which is a queue storing the nodes' marginal gains in decreasing order, as in [78]. At each iteration, we add the top node of the CELF queue at the seed set, until the budget k is reached (see Algorithm 4.2).

The influence gain of every node to be selected is calculated by subtracting the respective influence for which the candidates selected so far are responsible for, through common paths as shown in Theorem 1.

Algorithm 4.2 MatiLT

```
1:  Input: $G = (V, E)$, $k$        ▷ $k$: budget (number of seed nodes)
2:  Initialize: $S = \emptyset$
3:  $\mathscr{A}, \Omega$ = CalcStatsLT($G$)
4:  $Q$ = CalcInf($\mathscr{A}, V$)
5:  for $i = 1$ to $k$ do
6:      $s, \sigma(s) = Q.top()$
7:      $S = S \cup s$
8:      $U = V \backslash S$
9:      for each $u \in U$ do
10:         $\sigma(u) = Q(u)$
11:         for each $v \in S$ do
12:             $\sigma(u) -= \Omega(v, u)$
13:             $\sigma(u) -= \Omega(u, v)$
14:         $Q.add((u, \sigma(u)))$
15: return $S$
```

Algorithm 4.3 CalcStatsLT

```
1:  Input: $G = (V, E)$
2:  Initialize: $\mathscr{A}, \Omega = 0$
3:  for each $u \in V$ do
4:      Generate $\mathscr{T}(u)$ and $\mathscr{F}(\tau_i), \forall \tau_i$ using DFS
5:      $i = 0$
6:      for each $\tau \in \mathscr{T}(u)$ do
7:          $sum = 0$
8:          $i = i + 1$
9:          for each $v \in \tau$ in reverse order do
10:             $j \leftarrow index(v)$            ▷ index of $v$ in path
11:             $\Omega(u, v) += sum$
12:             $sum = sum + f_{ij}$
13:     for each $v \in V$ do
14:         Generate $\Psi(u, v)$ and $\Phi(\psi_i), \forall \psi_i$ (based on $\mathscr{T}(u)$)
15:         for each $\psi_i \in \Psi(u, v)$ do
16:             $\mathscr{A}(u, v) += \phi_{iN}$        ▷ $\phi_{iN} \in \Phi(\psi_i)$
17: return $\mathscr{A}$, $\Omega$
```

Algorithm 4.4 CalcInf

```
1: Input: $\mathscr{A}$, V
2: Initialize: $Q = \emptyset; \sigma(u) = 0, \quad \forall u \in V$
3: for each $u \in V$ do
4:     for each $v \in I(u)$ do          ▷ I(u): nodes influenced by u
5:         $\sigma(u) \mathrel{+}= \mathscr{A}(u, v)$
6:     Q.add((u, σ(u)))
7: return Q
```

4.4.3 *Influence Computation under the IC Model*

In the IC diffusion model, the activation probability of a node to another one in a path can be calculated by multiplying the influence weights $p^{\tau}_{\ell,\ell+1}$ leading to it in path τ. That is, in a path $\psi_i = \{n_{i1} = u, n_{i2}, ..., n_{iN} = v\}$, the influence of node u to node v can be calculated as follows:

$$\mathscr{A}_{\psi_i}(u, v) = \prod_{\ell=1}^{N-1} p^{\psi_i}_{\ell,\ell+1} = \phi_{iN} \tag{4.8}$$

The total activation probability of node v from node u, while taking into consideration the L different (unique) paths ¯$(u, v) = \{\psi_1, \psi_2, ..., \psi_L\}$ that lead from u to v, can be computed as:

$$\mathscr{A}(u, v) = 1 - \prod_{\psi_i \in \Psi(u,v)} (1 - \sigma_{\psi_i}(u, v)) = 1 - \prod_{i=1}^{L}(1 - \phi_{iN}), \tag{4.9}$$

where $\prod_{i=1}^{L}(1 - \phi_{iN})$ is equal to the probability that node u does not influence v (i.e., none of the paths from u to v is active).

In the case of the IC model, $\Omega(u, v)$ cannot be calculated only according to the influence weights $p_{k,k+1}$ in a specific path. In fact, the calculation of the influence in the case of the addition of a node u will change the so far calculated influence of a set of seed nodes S. Therefore, we use the following heuristics to compute the additional influence of node u, i.e., $\sigma(S + u)$:

$\sigma(S + u)$ **computation:**

1. For every path originating from node u (i.e., $\mathscr{T}(u)$) or a node belonging to seed set S, we keep the subpaths before falling into a node belonging to $S \cup \{u\}$.
2. The $\sigma(S + u)$ is equal to the sum of the influence probabilies that correspond to each of these subpaths.

Algorithm 4.5 shows the structure of the MATI algorithm under the IC model. Initially, routines CalcStatsIC (Algorithm

Algorithm 4.5 MatiIC

```
1: Input: $G = (V, E), k$        ▷ k: budget (number of seed nodes)
2: Initialize: $S = \emptyset, \sigma(S) = 0$
3: $\mathscr{A}$ = CalcStatsIC(G)
4: Q = CalcInf($\mathscr{A}$, V)
5: for $i = 1$ to $k$ do
6:     $s, \sigma(s) = Q.top()$
7:     $S = S \cup s$
8:     $U = V \setminus S$
9:     $\sigma(S) = \sigma(S) + \sigma(s)$
10:    for each $u \in U$ do
11:        $\sigma(S + u) = |S \cup \{u\}|$
12:        $\sigma(S + u)$+= AdditiveInf($\mathscr{T}(u)$, $\mathscr{F}(u)$, S)
13:        $\sigma(S + u)$+= AdditiveInf($\mathscr{T}(S)$, $\mathscr{F}(S)$, $S \cup \{u\}$)
14:        $Q.add((u, \sigma(S + u) - \sigma(S)))$        ▷ Order is maintained
15: return S
```

Algorithm 4.6 CalcStatsIC

```
1: Input: $G = (V, E)$
2: Initiaze: $\mathscr{A} = 0$
3: for each $u \in V$ do
4:     Generate $\mathscr{T}(u)$ and $\mathscr{F}(\tau_i)$, $\forall\ \tau_i$ using DFS
5:     for each $v \in V$ do
6:         $pr = 1$
7:         Generate $\Psi(u, v)$ and $\Phi(\psi_i), \forall \psi_i$ (based on $\mathscr{T}(u)$)
8:         for each $\psi_i \in \bar{}(u, v)$ do
9:             $pr = pr * (1 - \phi_{ij})$
10:        $\mathscr{A}(u, v) = 1 - pr$
11: return $\mathscr{A}$
```

Algorithm 4.7 AdditiveInf

```
1: Input: $\mathscr{T}, \mathscr{F}, S$        ▷ $\mathscr{T}$: set of paths, S: set of nodes
2: Initialize: $i = 0; inf = 0$
3: for each $\tau \in \mathscr{T}$ do
4:     $i = i + 1$
5:     for each $u \in \tau$ do
6:         $j \leftarrow index(u)$
7:         if $j == 1$ then
8:             continue
9:         else if $u \notin S$ then
10:            $inf$ += $f_{ij}$
11:        else
12:            break
13: return $inf$
```

4.6) and CalcInf (Algorithm 4.4) are called to compute $\mathscr{A}$ and the contents of CELF queue Q, respectively. While calculating the marginal influence for every candidate node u, routine AdditiveInf (Algorithm 4.7) computes the additional influence as previously described (see $\sigma(S + u)$ computation).

4.5 EXPERIMENTAL EVALUATION

In this Section, we present experimental results concerning the performance of the proposed algorithm for influence maximization. We have conducted experiments in real-world datasets in order to evaluate the performance of the MATI algorithm and compare it to state-of-the-art influence maximization algorithms on the quality of results and efficiency. The algorithm has been implemented in Python and all experiments are run on a Linux machine with a 3.00GHz CPU Intel Xeon CPU and 64GB memory.

4.5.1 *Datasets*

We have used four publicly available graph datasets: NetHept, WikiVote, Epinions and Email-EuAll. High level characteristics of the networks are shown in Table 4.1. A more detailed description of the datasets is presented in Section 2.4 of Chapter 2. To generate influence weights on all edges, we adopt the classical uniform method by [75, 93]. More precisely, we set the weight of every incoming edge of a node v to be equal to $\frac{1}{d_v}$, where d_v is the in-degree of node v. It has to be noted that the datasets were transformed from an undirected format to a directed one by simply assuming that if an edge between two nodes exists the one can influence the other. Thus, the number of the edges used in the experiments is twice the one that is reported.

4.5.2 *Baseline Algorithms*

In order to evaluate the performance of the MATI algorithm, we compare the respective results with those of four baseline algorithms which are described below:

- **Degree**: A heuristic based on the concept of "degree centrality", considering high-degree nodes as influential [93]. The seeds are nodes with k highest *out-degrees*.
- **Greedy**: The original greedy algorithm with Monte-Carlo Simulations. Following the literature [93], we run $10,000$

Dataset	NetHEPT	WikiVote	Epinions	Email-EuAll
# Nodes	15K	7K	75K	225K
# Edges	62K	103K	405K	341K

Table 4.1: Properties of the real-world graphs used.

Monte Carlo (MC) simulations to estimate the spread of any seed set.

- **LDAG**: The algorithm using locality properties as proposed in [41]. Influence parameter θ is set to $\frac{1}{320}$ as used by the authors.
- **SimPath**: The algorithm proposed in [77]. The pruning threshold is set to 10^{-3} and the look-ahead value l is set to 4 as proposed by the authors.

Unless noted otherwise, the threshold θ for the proposed MATI algorithm is set to 0.0001. The value was chosen experimentally, based on performance observation.

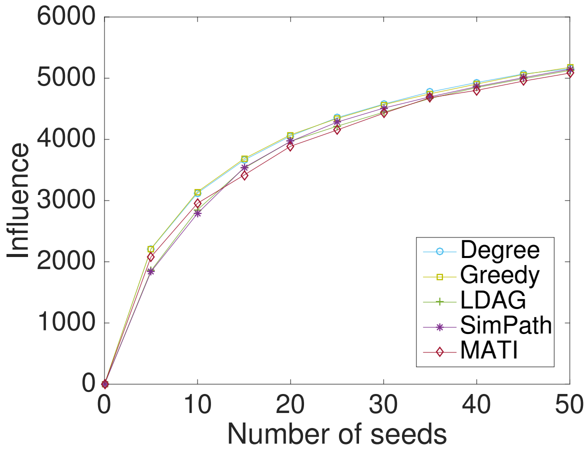
4.5.3 *Experimental Results*

We compare the performance of the aforementioned algorithms, with respect to the quality of seed sets and efficiency aspects.

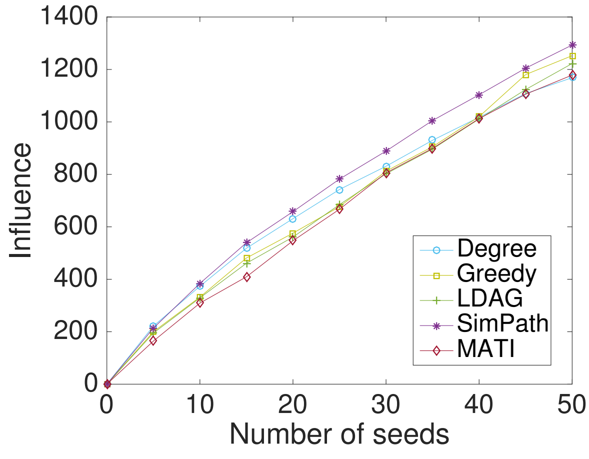
QUALITY OF SEED SETS. The quality of the seed sets obtained by different algorithms is evaluated based on the expected spread of influence measured in number of nodes. Figures 4.1 and 4.2 show the spread of influence versus the size of seed set, under the LT and IC models respectively.

Under the LT model, the seed sets obtained via MATI are quite competitive in quality compared to those of the Greedy, LDAG and SimPath algorithms. For all four datasets, the influence loss for up to 50 seeds is less than 2%. Under the IC model, our algorithm is still efficient, despite the heuristics involved in the influence estimation.

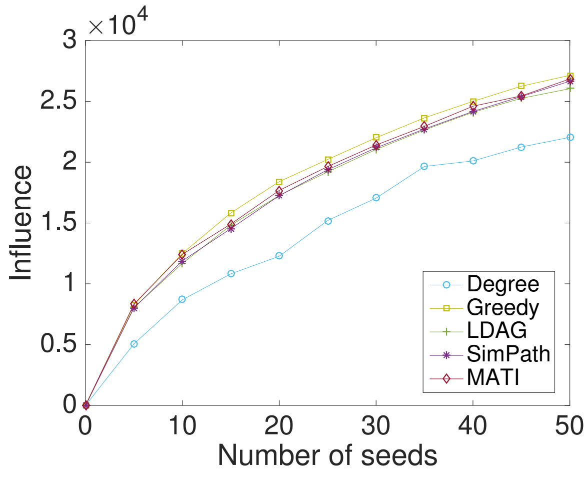
We have also performed experiments for different values of the parameter θ of our algorithm for the NetHEPT dataset, in order to observe the running time of our algorithm with respect to the influence that is achieved. The results are depicted in Table 4.2. As θ decreases, the running time is always increasing. This is justified by the fact that a smaller θ allows computation of influence in a greater neighborhood around each node. In most of the cases, the influence achieved also increases. This is justified by the fact that the foration of paths of greater length provide a more accurate computation of a node's influence.



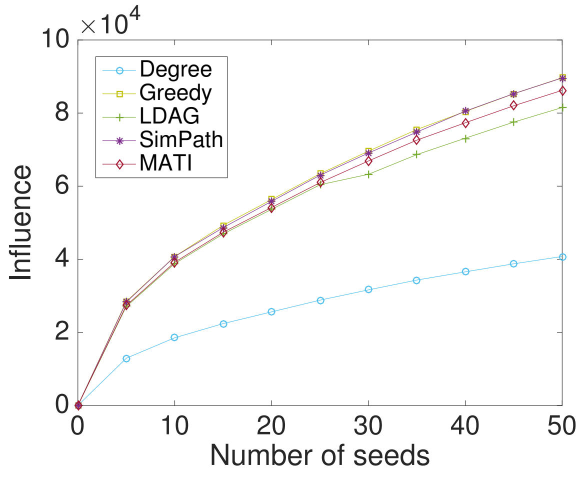
(a) WikiVote



(b) NetHEPT

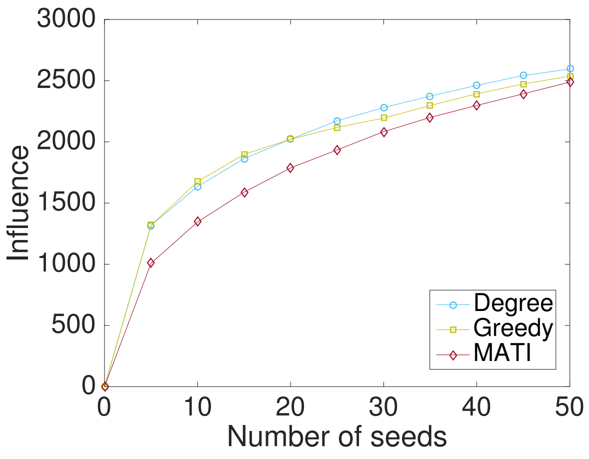


(b) Epinions

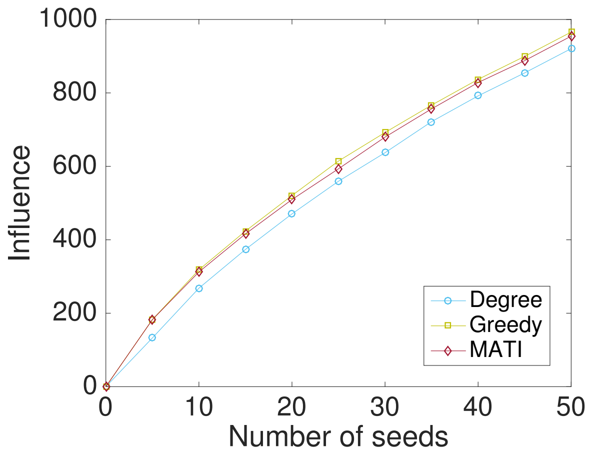


(c) Email-EuAll

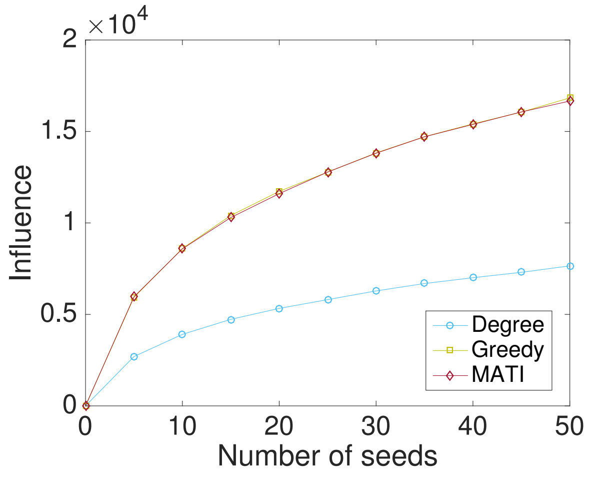
Figure 4.1: **Influence spread in number of nodes for the different algorithms, under the LT model.** We show results for the following networks: **(a)**WikiVote; **(b)**NetHEPT; **(c)**Epinions; **(d)**Email-EuAll. Each plot depicts the influence in number of nodes achieved by the different methods: Degree, Greedy, LDAG, SimPath and MATI. Each point shows the number of nodes that the respective number of seed nodes - given by the different methods - achieves to influence.



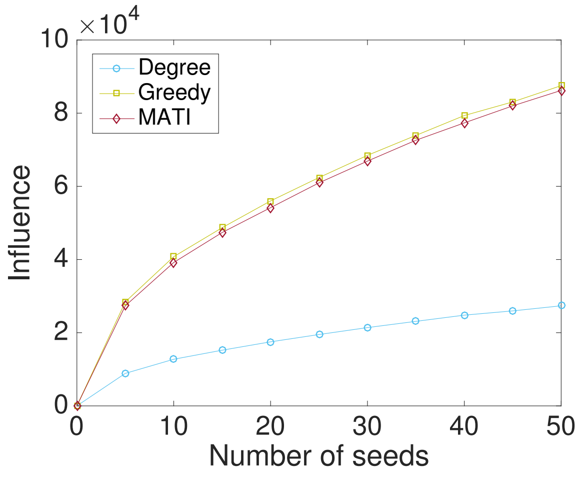
(a) WikiVote



(b) NetHEPT



(c) Epinions



(d) Email-EuAll

Figure 4.2: **Influence spread in number of nodes for the different algorithms, under the IC model.** We show results for the following networks: **(a)**WikiVote; **(b)**NetHEPT; **(c)**Epinions; **(d)**Email-EuAll. Each plot depicts the influence in number of nodes achieved by the different methods: Degree, Greedy and MATI. Each point shows the number of nodes that the respective number of seed nodes - given by the different methods - achieves to influence.

θ	Running Time (s)	Influence
0.1	1.2	984.7
0.01	6.5	1162.3
0.001	88.6	1209.6
0.0001	708.2	1190.8

(a)

θ	Running Time (s)	Influence
0.1	1.2	867.77
0.01	6.8	959.42
0.001	106.5	938.21
0.0001	820.6	950.32

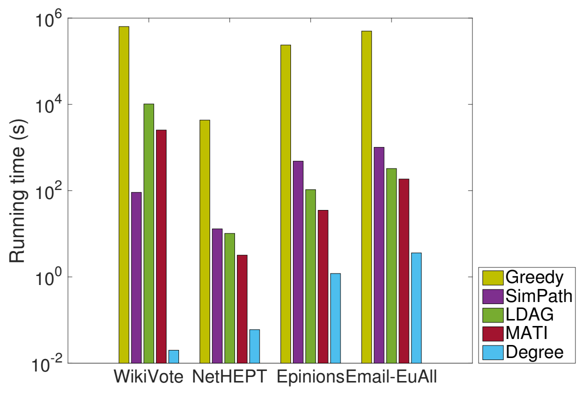
(b)

Table 4.2: Comparison of running times in seconds and influence spread in number of nodes for different values of the parameter θ under (a) the LT and (b) the IC model for the NetHEPT network.

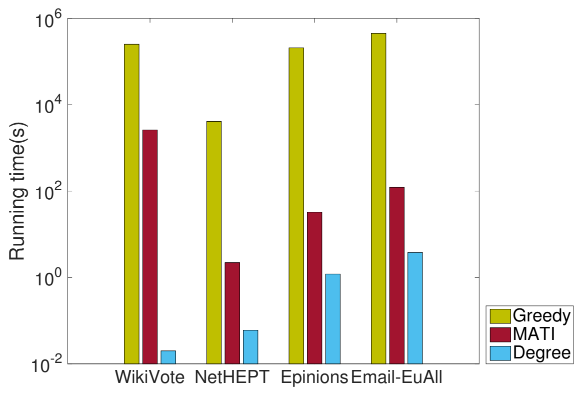
efficiency of mati. We have also examined the running time of the proposed algorithm. Figure 4.3 reports the execution time required by various algorithms for the LT and IC models respectively. The figures have a logarithmic scale on the y-axis. In all cases, MATI is faster than the Greedy and LDAG algorithms. In all datasets except WikiVote, MATI also performs better that SimPath. It takes Greedy more than one week to select 50 seed nodes for datasets such as Epinions. We should also mention here that, although the Degree heuristic is time efficient, it fails to output a seed set of high quality.

4.6 conclusions and future work

Identifying vital nodes in networks which are associated with some certain structural or functional objectives is a very significant task with various applications in numerous domains. In the previous Chapter the problem of identifying individual vital nodes was introduced. Nevertheless, many real-world applications require a small set of nodes that play a crucial role in information diffusion. A common marketing strategy when the budget is limited, is to show advertisements and provide small samples or even discounts to the specific set of customers that are highly probable to buy the product and inluence many people to buy it too. For some military applications, it is re-



(a) Running time - LT model



(b) Running time - IC model

Figure 4.3: **Comparison of running times in seconds of the different algorithms under the (a) LT and (b) IC models.** We show results for the following networks: WikiVote; NetHEPT; Epinions; Email-EuAll. Each plot depicts the running time in seconds that each different algorithm requires to produce a group of k = 50 seed nodes. Results for the following methods are shown: Degree, Greedy, LDAG, SimPath and MATI.

quired that some few critical nodes of the enemy are destroyed in order to the greatly reduce their communication capacity.

A greedy algorithm has been proposed for the so-called problem of influence maximization which is proven to provide good approximation guarantees. The serious drawback of the greedy algorithm is that it is very time-consuming. Many algorithms have been proposed in order to surpass greedy's main drawback. They focus on reducing the computation time either by reducing the number of spread evalutations, either by making batch computations of the influence spread or by designing heuristics that will efficiently compute the respective spread.

In this Chapter, we proposed MATI, an efficient influence maximization algorithm under both the LT and IC diffusion models. By taking advantage of the possible paths that are created in each node's neighborhood, we have designed an algorithm that succeeds in locating the users that can maximize the influence in a social network while also being scalable for large datasets.

Specifically we take advantage of the fact that when we want to calculate the influence gain that a node will add to the influence of a group of nodes, we just need to subtract the influence of the common paths of the new node and those of our already existing seeds from this new node's individual influence to the network. Our algorithm can be seen an extension of the SimPath algorithm in what concerns its formulation for the LT model. Taking advantage of the possible paths created and precalculating the influence gain of a node for the IC model though has not beed proposed, in the best of our knowledge, by any related work. The methods used to precalculate each node's potential influence depends on the creation of matrices which may on one hand increase the memory consumption of the algorithm while at the same time facilitating the re-computation of the seeds in the case that some nodes and edges are deleted. In order to limit the computation of the possible paths and the respective probabilities of them being "active", we use a pruning threshold θ which reduces the running time but also the accuracy of the influence computation. Extensive experiments show that for both the cases of the LT and IC models, MATI performs better than the baseline methods both in terms of influence and computation time.

As future work, we plan to further evaluate our algorithm by doing more experiments with larger datasets and comparing it with more baseline methods. Additionally we are experimenting with heuristics that can speed up more the running time and efficient data structures that may reduce the memory consumption of our algorithm. It would be interesting to experiment with the structural centralities that have been proven efficient in identifying individual influential spreaders in order to see whether those centralities can also provide useful insights for the influence maximization problem. Finally we are studying how our method can be extended for the case of a dynamic graph where nodes and edges are added or deleted from the network.

5 PRIVATE, SECURE AND DISTRIBUTED COMPUTATION OF K-CORES

The focus of the dissertation until now was on identifying those specific nodes in the network that would, as individuals or by acting all together as a group, maximize the spread of influence across the network. This information can be proven invaluable specifically for advertisers, for building on-line services and viral marketing campaigns. However, sharing such social networks raises severe privacy concerns. The goal of this Chapter is to calculate a metric which measures the influence of each node of the network in a secure and privacy-preserving way. To that end, we capitalize on the k-core decomposition which has been proved to locate higly influential spreaders. We build a distributed Peer-to-peer (P2P) algorithm that securely calculates the k-core numbers and therefore the spreading properties of the nodes in a network. We show that our algorithm can succesfully calculate the specific metric for dynamic graphs while limiting the calculations and the number of the messages exchanged among peers. Finally we show that our algorithm can run on anonymized graphs while maintaining good quality of results.

5.1 INTRODUCTION

Identification of influential spreaders in networks has been the focus of many researchers from various different scientific fields. Numerous metrics and algorithms have been introduced in order to locate those "important" nodes in complex network structures as presented in the previous chapters. Indeed, it has been shown [98, 120, 143] that the identification of dense subgraphs can lead to a good estimation of the "best" initial nodes (called *influential nodes*), both in terms of spread speed and total number of nodes reached. The results of such studies have been proven significant to a great range of applications that include collective dynamics and viral marketing.

Nevertheless, the aforementioned methods require knowledge of the social network. Such practices raise serious concerns associated with the publishing of such sensitive information. A trivial solution to this problem would be to remove the identifying information from the network by removing user-specific data concerning each individual node. Unfortunately such a solution is not impenetrable to possible attacks that aim to reveal the true identities of targeted users [6]. The latter work triggered the proposal of several anonymization methods whose purpose is to limit the risk of privacy breach in shared data [23, 164]. Another trend that exists towards circumventing the privacy problem is secure computation using cryptographic techniques such as homomorphic encryption [69, 74]. The major drawbacks of the cryptographic approach is that, for the moment, it is computationnaly costly (when not unfeasible).

In this work, we adopt the *decentralization* approach to favor privacy. Distributed graph computation models have gained great attention the latest years as they can effectively perform computations over large-scale graphs which became very relevant for numerous Web-related applications.

We specifically propose a Peer-to-peer (P2P) algorithm for distributed k-core decomposition. k-coreness has been proved to be a metric that efficiently locates those nodes that –while acting individually – will disseminate informaton to a larger part of the population [98]. Montresor et al. [125] were the first to propose an algorithm devised with a Peer-to-Peer scenario in mind (i.e., *fully distributed*). They succeed in designing an algorithm that completes the k-core decomposition in $O(N)$ rounds for graph with N number of nodes. However, in real-world applications, the network evolves over time. Several nodes and/or edges may be added to the initial state of the graph. In such dynamic networks, it is crucial to have the up-to-date k-core values of the nodes which constitutes a difficult problem – usually called the *core maintenance* problem.

An edge addition or deletion may affect though not only the coreness of the two end nodes, but also that of their neighbors. The update could even spread across the network and change various k-core values. Thus we deduce that determining which node in a network should update its core number given the network changes is not a straightforward task. For a small network with few updates, a trivial alternative would be to execute the P2P k-core decomposition algorithm [125] every time an update takes place. This solution would lead to a high number of messages and computations as well as to a non-negligible convergence time which could cause issues if updates are frequent. It could possibly be an acceptable solution for small networks with low dynamism but it is inadequate in our context: fully distributed computations in a large network with frequent updates.

Our proposed approach, inspired by the one by Montresor et al., constitutes an incremental algorithm that solves the core maintenance problem while limiting the number of computations and messages exchanged for each update. The main contributions of this work can be summarized as follows:

- *P2P algorithm that solves the core maintenance problem*: We propose an incremental algorithm that efficiently solves the core maintenance problem in P2P, limiting the number of messages and computations needed to successfully update the core numbers for the vertices when an edge is inserted or deleted, thus respecting these constraints.
- *Complexity analysis on real graphs*: We provide an experimental evaluation of our proposed P2P algorithm, that shows that it can correctly scale to large-scale networks.
- *Security and privacy analysis on real graphs*: We analyse the security and privacy aspects of our algorithm using two different adversary models. We discuss the desired privacy and information quality that needs to be achieved in our scenario and perform experiments on real datasets.

The rest of the Chapter is organized as follows. Section 5.2 presents the problem statement and some background concepts that are used throughout the Chapter and Section 5.3 reviews the related literature on k-core decomposition and core maintenance algorithms. Then, in Section 5.4 we present the proposed P2P algorithm for core maintenance. Sections 5.5 and 5.6 present a detailed computational complexity and security and privacy analysis of our method respectively. Finally, in Section 5.7 we present concluding remarks.

5.2 PROBLEM STATEMENT AND PRELIMINIARIES

In this Section, we present the problem studied and the preliminary concepts upon which our approach for a P2P core maintenance algorithm is built. Initially, we recall the concept of k-core decomposition in graphs. Then, we present the theorems and definitions that were the basis towards the composition of our algorithm. Table 5.1 provides a list of symbols used in this Chapter, along with their definitions.

5.2.1 *Problem Statement*

In this work, we are interested in the following problem: *how to compute through an* ***efficient, correct, secure, and privacy-preserving*** *algorithm a metric which will measure the influence of each node of the network.*

Hypothesis 1. **(Peer-to-Peer)** The social network is considered as a P2P system. The algorithms developed must be P2P algorithms.

There have also been many attempts to propose distributed and private social networks [52]. Arguably, the most successful is Diaspora, in which *pods* represent end-points to which users connect, and constitute the distributed infrastructure. In a *fully decentralized* context, which is the case considered in this work, each user would create and manage his own pod. There is no central entity that has a global knowledge of the network.

Nevertheless, creating algorithms in a decentralized setting still raises the following question: *what is the privacy leakage of the information shared with other nodes when running the algorithm?* In our work, we consider the following privacy constraint, applied to the computation of k-cores:

Constraint 1. **(Privacy)** It must not be possible for a node (resp. a set of colluding nodes) to reconstruct partially or entirely the graph using the information it (resp. they) obtain during the execution of our algorithm.

Symbol	**Definition**
$G = (V, E)$	Undirected graph G
V, E	Node and edge set of graph G
$d_G(v)$	Degree of node $v \in V$
$\mathtt{neighbor}_G(v)$	Function returning the set of nodes u such that $(u, v) \in E$
$k_G(v)$	Coreness/ k-core number of node v of graph G
$ICS(v)$	Induced Core Subgraph of node v
$PIICS(v)$	Potential Incrementation Induced Core Subgraph of node v
$DICS(v)$	Decrementation Induced Core Subgraph ofnode v
G_{anon}, $\tilde{G}$	Anonymized and modified version of graph G
α,β	percentage of nodes to be added and deleted
q,q_N	Quality and Normalized Quality
q_1, q_2	base and randomized quality
$X\%$, z	percentage and actual number of influential nodes
V_N,$V_{i<N}$	set of nodes of the N_{th} and $N-1$ top coreness groups of G
V'_N, $V'_{i<N}$	set of nodes of the N_{th} and $N-1$ top coreness groups of $\tilde{G}$

Table 5.1: List of symbols and their definitions.

5.2.2 *Preliminaries and Background*

We assume that each individual node of the graph $G = (V, E)$ are actually the hosts of the distributed system. Each node v is initially only aware of its degree $d_G(v)$ and has access to the function $\mathtt{neighbor}_G(v)$ which returns the set of nodes u such that $(u, v) \in E$.

We briefly recall to the notion of k-core decomposition in networks (a detailed description can be found in Chapter X, Section X). Let $G = (V, E)$ be an undirected graph. C_k is defined to be the k-core subgraph of G if it is a maximal connected subgraph in which all nodes have degree at least k. Then, each node $v \in V$ has a core number $k_G(v) = k$ – also known as coreness, if it belongs to a k-core but not to a $(k+1)$-core.

Our distributed algorithm is based on the *locality theorem* proved by Montresor et al. [125] and the *k-core update theorem* [110] that follow.

Theorem 5.1 (Locality [125]). *For every node $v \in V$ of graph G, $k_G(v) = k$ if and only if a) there exist k neighbors of v whose coreness is greater than or equal to k and b) there are no $k+1$ neighbors of u whose coreness is greater than or equal to $k+1$.*

From the locality theorem, we conclude that knowing the coreness of its neighbors is sufficient for a node to compute its own coreness. Based on this theorem, Montresor et al. [125] developed a distributed peer-to-peer algorithm for the computation of the k-core decomposition of a network. Nevertheless, in real-world applications, the network evolves over time. Our distributed algorithm incorporates a first round of computations similar to the aforementioned algorithm where each individual node v is finally aware of its own coreness and that of its neighbors. Then, each time a change occurs, additional computations are triggered so that the coreness of all the nodes affected by the aforementioned change is updated. Potentially affected nodes are limited to some subgraph:

Definition 5.2 (Induced Core Subgraph (ICS)). *The core subgraph of G induced by a node $v \in V$, noted $ICS(v) = (V_I^v, E_I^v)$ is the maximal connected subgraph of the $k_G(v)$-core containing v such as:*

1. *$v \in V_I^v$; v is in H (i.e., the vertex inducing the ICS is in the ICS).*
2. *$\forall u \in V_I^v, k_G(u) = k_G(v)$; all node of $ICS(v)$ has a coreness exactly equal to (and no greater than) $k_G(v)$.*

Accordingly, the k-core update theorem limits the potential changes as follows:

Theorem 5.3 (k-core update theorem [110]). *Given our graph G and two nodes u and v in G, the insertion or deletion of an edge between u and v:*

- *if $k_G(u) > k_G(v)$, may impact V_I^v, the nodes that belong to $ICS(v)$.*
- *if $k_G(u) < k_G(v)$, may impact V_I^u, the nodes that belong to $ICS(u)$.*

- *if* $k_G(u) = k_G(v)$, *may impact* V_I^v *and* V_I^u*,the nodes that belong to the union of* $ICS(v)$ *and* $ICS(u)$*.*

The coreness of such a node may:

- *remain unchanged.*
- *in the case of an edge addition, increase by 1.*
- *in the case of an suppression, decrease by 1.*

Based on this theorem, whenever an edge between two nodes is added/deleted in G, only the nodes in the ICS induced by the node(s) with the smaller coreness may need to be updated. Note that the addition and suppression of a node don't change the coreness of the others; only edges matters since coreness is based on neighborhood.

5.3 RELATED WORK

In this section the relevant state-of-art in the areas of i) k-core decomposition and ii) core maintenance algorithms will be revised. Finally we take a look at decentralized personal data management platforms which demonstrate the soundness of the considered context and provide examples of systems where our proposed approach could be applied.

5.3.1 k-*core Computation*

The standard algorithm for k-core decomposition was proposed by Batagelj and Zaversnik [15]. They propose an algorithm which recursively deletes nodes (and the edges incident to them) that have a degree less than k. The core subgraphs are computed in increasing order (1-core,2-core, 3-core etc) and in linear time to the number of edges in the network. This algorithm runs efficiently if the entire network can fit in the main memory.

For graphs that cannot be kept in the main memory, Cheng et al. [42] proposed **EMcore**, an algorithm for massive networks. Contrary to the aforementioned bottom-up approach, they propose a top-down approach starting from the smallest-size core and recursively decreasing the search space and disk I/O cost for every core computed.

Wen et al. [163] propose a semi-external algorithm comprising optimization techniques to reduce I/O and CPU cost for core decomposition on web-scale graphs. Core decomposition is also studied in random graphs [117, 124] and uncertain graphs [24].

Montresor et al. [125] proposed a distributed k-core algorithm assuming that nodes of the network are located on separate computing nodes. They present two different models: i) one which assumes that one computational unit is associated with one node in the graph similar to Pregel, the distributed framework proposed by Google [119] and ii) one assuming one host stores a group of nodes with their local and remote edges. It is assumed that everything is held in the memories of the computing nodes.

5.3.2 *Core Maintenance*

Moriandi and Pelegrini [123] rely solely on the algorithm in [15]. If the graph is updated, the algorithm is re-executed and the core number of every node is re-calculated from scratch. This is obviously a computationally expensive solution for large graphs. Li and Yu [110] proposed a more efficient solution which limits the computations needed by determining the minimal subgraph for which the k-core decomposition might have to be updated.

Aksu et al. [1] propose a batch maintenance as changes occur dynamically in the graph. The proposed algorithms prune the search space to minimize the messages exchanged among the computing nodes that store the partitioned data. Saríyüce et al. [148] present an incremental k-core decomposition scheme. Based on the observation that when an edge is inserted or removed, the subgraph that may need to be updated is connected and reside in the subcore which the edge is in, they propose an algorithm linear in the size of the subcore. Zhang et al. [169] further improve the aforementioned approach by proposing a new order-based algorithm. By maintaining a k-order among vertices they significantly outperform the state-of-the-art algorithm up to 3 orders of magnitude.

5.3.3 *Decentralized Personal Data Management Platforms*

The work proposed in this Chapter is to be considered in the context of a distributed social network, or more generally a distributed data management platform, or personal cloud. Many companies such as OwnCloud, SandStorm, SeaFile, Tonido, Younity, CozyCloud, Lima or governmental initiatives such as the British[*] initiative, offer a *logically* decentralized approach to the management of personal data.

[*] mydex.org

Compared to Diaspora†, which is a decentralized social network, the current trend of empowering users to manage their own data goes further and continues to pursue a privacy objective, often with physical decentralization. For instance, OpenPDS and the SafeAnswers framework aim to minimize to the bear minimum the (personal) information shared with others when computing a query. Allard et al. [4] proposed an initial design for the Personal Data Server approach to use low-price secure hardware to execute local computations, while protecting the system from the user himself.

We believe that the work presented in this chapter belongs to this context and that our approach could be integrated into any logically or physically decentralized system.

In this next section, we formalize our P2P algorithm which is based on the theorems presented in the Section 5.2. It solves the core maintenance problem by triggering update mechanisms that involve the exchange of a limited number of messages and performance of computations whenever a change occurs.

5.4 P2P ALGORITHM FOR CORE MAINTENANCE

The algorithm can be divided in two parts:

i) *the static part* which involves the initial calculations so that all the nodes of the initial format of the graph are aware of their own and their neighbors' corenesses. This part is similar to the "one node, one host" algorithm introduced by Montresor et al. [125].

ii) *the dynamic part* involving isolated graph perturbations during which new coreness estimations are calculated. We consider that the system is stable prior to graph evolutions and that the maintenance algorithms are performed after the static algorithm has converged.

Before describing the algorithms that concern the coreness estimation for each node, we will be giving a description of the local variables stored in each node and the different types of messages exchanged among them.

† diasporafoundation.org/

5.4.1 *Local variables*

Each node u maintains some variables representing their local knowledge. The three first variables are similar to the static case:

- ***myCrns*** is an integer that represents node's u own estimated coreness; initially set to its degree.
- ***est***{}, the partial view of u, is a map linking node u's neighbors ids to the most up-to-date information regarding their respective coreness estimations. Initially, all the estimations are set to $+\infty$.
- ***changed*** is a Boolean flag set to true if u's state has been modified since its last communication. It is initialized to false.

The following variables are related in particular to core maintenance:

- ***isInc*** is a Boolean flag set to true if the node is handling a possible incrementation of its coreness. It is initially set to false.
- ***incrEst***{}, similar to ***est*** is a hashmap containing node u's neighbors ids and some coreness estimations. Used during an attempted incrementation, estimations are made assuming neighbors with coreness $k_G(u)$ will be affected (i.e., increased). It is initially empty.
- ***toHandle*** is an identifier of events to be handled by u. Such events are a particular kind of change in the graph topology: edge additions. It is initially empty. Since concurrent events are not considered for now, *toHandle* is supposed to be a single identifier in the remainder of the section.
- ***handled*** is a set of identifiers containing already handled events in order to avoid redundant treatments and ensure convergence. It is initially empty.

5.4.2 *Handling Messages and Events*

Each node can send and receive different types of messages. Graph evolution and message exchange trigger corresponding routines described in Algorithm 5.1 and Algorithm 5.3, respectively.

5.4.2.1 *Events and graph evolutions*

The routine execute by each peer (i.e., each node u) that handles graph evolutions is described in Algorithm 5.1.

- During ***Initialization*** the variables listed in the previous section are set to their initial values. The **creation** of some node u (i.e., its addition to the network) simply triggers the initialization routine on u. Note that when added, a node has no neighbor and therefore has no particular impact on other nodes' coreness.

- The ***Suppression*** routine triggers the suppression of all edges incident to u.

- The ***Edge Suppression*** routine concerns the deletion of an edge (u, v). It is a unilateral decision taken by v, one of u's extremity nodes. v sends a message **<v>** to u, terminating its relation with the receiver.

- An ***Edge Addition*** is a bilateral event. Accordingly, we suppose that during such addition, extremities exchange their coreness estimation, agree on an identifier $eventID$, and trigger the corresponding routine: *on edge addition <v,k,eventID>* where v is the identifier of the other extremity and k its coreness estimation. For the addition of an edge (u, v) the identifier is assumed to be generated by a hashing function $hash$ taking as input u, v and the local time of u and/or v. In concordance with theorem 5.3, the extremity with the lowest coreness may have to increment its estimation.

5.4.2.2 *Attempting coreness incrementation*

The *tryIncrement* procedure, described in Algorithm 5.2, initializes a coreness incrementation attempt of node u. It sets the $isInc$ flag to true and stores the id of the event to handle. Most importantly, it builds an alternate partial view $incEst$ integrating the hypothesis that all neighbors of u that could be impacted by the event $eventID$ will be; i.e. all neighbors belonging to the ICS will increment their coreness. In the next round of computations, the node will determine whether it is possible for its coreness to be incremented with this hypothesis. Depending on the outcome, the node will send a message to its neighbors either propagating the potential incrementation or notifying its failure to increment.

Algorithm 5.1 Handling graph evolutions in P2P k-core decomposition with maintenance; routine executed by node u.

```
1: on Initialization/creation do
2:     myCrns ← d(u)
3:     isInc ← false
4:     changed ← true
5:     handled ← ∅
6:     toHandle ← NULL
7:     for each $v \in neighbor(u)$ do
8:         est[v] ← +∞
9:         send $< u, myCrns >$ to $v$
10: on Suppression do
11:     for each $v \in neighbor(u)$ do
12:         send $< u >$ to $v$
13: on Edge Suppression $(u, v)$ do
14:     send $< u >$ to $v$
15:     changed ← true
16:     est[v] ← 0
17:     if isInc then
18:         incrEst[v] ← 0
19: on Edge Addition <v,k,eventID> do
20:     est[v] ← k
21:     if myCrns $\leqslant$ k then
22:         tryIncrement($eventID$)
```

Algorithm 5.2 Initializing an incrementation attempt on node u.

```
1: procedure TRYINCREMENT(eventID)
2:     toHandle ← eventID
3:     isInc ← true
4:     changed ← true
5:     incEst ← est
6:     for each $v \in neighbor(u)$ do
7:         if incEst[v] = myCrns then
8:             incEst[v] ← incEst[v] +1
```

5.4.2.3 *Messages*

Here we provide a description of the messages that are exchanged between the nodes. The first two types of messages serve to notify for an update of the coreness estimation while the third notifies for an edge suppresion. The protocol is described in Algorithm 5.3.

- **<u,k>**: the sender, node u, sends an update of its own coreness estimation, k. This message is similar to the message exchanged in the static case. The receiving node will update its local knowledge (i.e., its own and immediate neighbors coreness estimations) and switch the $changed$ flag to true accordingly. Note that, in the dynamic case, this message is also used to notify for a failure of coreness incrementation.

- **<u,k,eventID>**: the sender, node u, sends an update of its own coreness estimation, k, and propagates the event with id $eventID$. The receiver, node v, updates its local knowledge accordingly and triggers a potential coreness incrementation. In concordance with theorem 5.3, this incrementation may occur only if v belongs to the *Induced Core Subgraph (ICS)* of u, and therefore only if its coreness is $k-1$ (i.e., the coreness of v before its incrementation). In addition, the incrementation may only occur if v did not already handle the event. If it already did but is still of coreness $k-1$, it means that v's coreness could not increment after the event occured. Since u, during its incrementation attempt, made the hypothesis that v's coreness could increment, v notifies u of its failure through the message: $< u, k-1 >$ in order to invalidate this hypothesis.
- **<u>**: the sender, node u, notifies that an edge has been deleted. The receiver deletes u as its neighbor, which is equivalent to considering u having a coreness of 0. The est map is updated accordingly and the $changed$ flag is set to true.

5.4.3 *Computing coreness estimations*

Periodically, each node re-estimates its coreness according to its new local knowledge. The periodic behaviour and the process leading to the estimation are described in Algorithms 5.4 and 5.5.

5.4.3.1 *Periodic behaviour*

Similar to the static case [125], the protocol execution is divided in periodic rounds to limit computations and the number of exchanged messages. Every δ time units, the variable *changed* is checked; if the local knowledge has been modified, a new coreness estimation – $newCrns$ – is computed. Flags are set back to false and, if relevant, the event handled is pushed to the $handled$ set. The result of this computation may be broadcasted to neighbors.

Algorithm 5.3 Handling messages in P2P k-core decomposition with maintenance; routine executed by node u.

```
1: on Receive <v,k> do
2:     if est[v] ≠ k then
3:         changed ← true
4:         est[v] ← k
5:         if isInc then
6:             incrEst[v] ← k
7: on Receive <v,k,eventID> do
8:     if est[v] ≠ k then
9:         changed ← true
10:        est[v] ← k
11:        if isInc then
12:            incrEst[v] ← k
13:    if myCrns=k-1 then
14:        if eventID ∈ handled then
15:            send $< u, myCrns >$ to $v$
16:        else
17:            tryIncrement($eventID$)
18: on Receive <v> do
19:    changed ← true
20:    est[v] ← 0
21:    if isInc then
22:        incrEst[v] ← 0
```

- During the static case ($isInc$=false):
  - Node u is not handling an incrementation. If the newly estimated coreness $newCrns$ has decreased, $myCrns$ takes its value and the message $< u, newCrns >$ is sent to u's neighbors to notify them for the update of the coreness estimation.
- When the node is handling a potential incrementation ($isInc$=true):
  - If the node's coreness estimation has been increased, the message $< u, newCrns, toHandle >$ is sent, notifying for the update of the coreness estimation and propagating the potential incrementation corresponding to the event $toHandle$. The local knowledge of u, stored in est, is replaced by $incEst$: u maintains its hypothesis that all of its neighbors belonging to the same ICS will manage to increment their coreness estimation.

Algorithm 5.4 Estimating local coreness; routine executed by node u.

```
1: repeat every $\delta$ time units (round duration)
2:     if changed then
3:         if ! isInc then
4:             newCrns $\leftarrow$ computeEstimate(est)
5:             if newCrns < myCrns then
6:                 myCrns $\leftarrow$ newCrns
7:                 for each $v \in neighbor(u)$ do
8:                     send $< u, newCrns >$ to $v$
9:         else
10:            newCrns $\leftarrow$ computeEstimate(incEst)
11:            if newCrns>myCrns then
12:                est $\leftarrow$ incEst
13:                for each $v \in neighbor(u)$ do
14:                    send <$u, newCrns, toHandle$> to $v$
15:            else
16:                for each $v \in neighbor(u)$ do
17:                    send $< u, newCrns >$ to $v$
18:            handled $\leftarrow$ handled $\cup$ {toHandle}
19:            toHandle $\leftarrow$ NULL
20:            isInc $\leftarrow$ false
21:        changed $\leftarrow$ false
22:        myCrns $\leftarrow$ newCrns
```

Algorithm 5.5 Computation of node u's upper bound coreness.

```
1: procedure COMPUTEESTIMATE(estimation)
2:     for i from 1 to myCrns do
3:         count[i] $\leftarrow$ 0
4:     for each $v \in neighbor(u)$ do
5:         j $\leftarrow$ min(myCrns, estimation[v])
6:         count[j] $\leftarrow$ count[j]+1
7:     for i from myCrns downto 2 do
8:         count[i-1] $\leftarrow$ count[i-1] + count[i]
9:     i $\leftarrow$ myCrns
10:    while i > 1 and count[i] < i do
11:        i $\leftarrow$ i - 1
       return i
```

– If the node's coreness estimation has not been increased, a message $< u, newCrns >$ is sent. Even if its estimation remains the same, the message will serve to invalidate the hypothesis made by one or several of its neighbors that u could increment its coreness. The local knowledge of u remains stored in est while the incremented coreness estimations of the neighbors included in incEst are discarded.

5.4.3.2 *Estimating coreness*

The procedure computeEstimate described in Algorithm 5.5 is used to estimate an upper-bound of a node's coreness. It is similar to the static case [125] and stems directly from the locality theorem (Theorem 5.1). The only difference is that, in the case of an incrementation attempt, the local knowledge used to estimate coreness is incEst rather than est; the node then supposes that its neighbors in the same ICS will increment their coreness.

5.4.4 *Example*

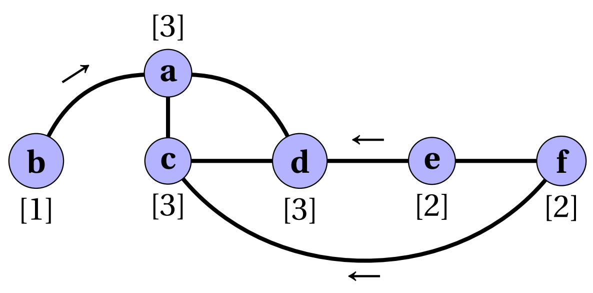
In this section we are describing a run of the algorithm on a sample graph. At first, with no perturbation, the algorithm runs similarly to its static counterpart as depicted in Figure 5.1. Then, its dynamic behaviour when confronted to the addition of an edge is detailed and illustrated in Figure 5.2.

5.4.4.1 *Static run*

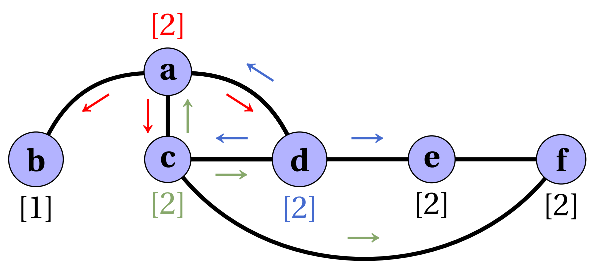
Initially all nodes have a coreness equal to their degree and the flag isChanged is set to true.

- **At t=0.** Nodes *a*, *c*, and *d* propagate their coreness estimation (MyCorness) of 3 by sending their neighbors the messages $< a, 3 >$, $< c, 3 >$, and $< d, 3 >$, respectively. Their neighbors update their local knowledge est accordingly. These updates do not however cause any change to their own estimations. Similarly, node *b* notifies node *a* that its $myCrns = 1$, so that the latter will update its coreness estimate to 2 at the end of the round when re-computation occurs. At the same time, nodes *f* and *e* notify node *c* and *d* respectively about their estimation being equal to 2 which will cause the myCrns value of the latter nodes to change from 3 to 2 during the next computation (see Figure 5.1a).

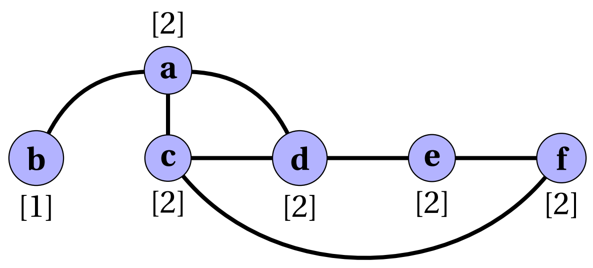
- **At t= δ.** All nodes re-compute their estimation and set their flag **isChanged** to false. As seen previously, nodes a, c, and d modify their estimation and notify their neighbors {b,c,d}, {a,d,f} and {a,c,e}, respectively (see Figure 5.1b). All nodes therefore change their local knowledge est and switch their isChanged flag to true.



(a) At t= 0.



(b) At t= δ.



(c) At t=2*δ.

Figure 5.1: **A simple example describing the static part of the algorithm.**

- **At t=2*δ.** All nodes re-compute their estimation. Nevertheless, myCrns estimates of the nodes do not change from now on; the algorithm has converged for this initial state of the graph (see Figure 5.1c).

5.4.4.2 *Dynamic run - One perturbation*

- **At (k-1)*δ $\leq$ t < k $*$ δ.** An event occurs: an edge is added between nodes a and e (see Fig. 5.2a). They exchange messages in order to notify each other about their current coreness estimation, agree on an event identifier – eventID– and trigger the corresponding routines *On edge addition <e,2,eventID>* and *On edge addition <a,2,eventID>*, respectively. Since they have the same coreness, it is possible for the coreness of both to be incremented according to theorem 5.3. Both therefore trigger tryIncrement. Flags are set to true, eventID is stored in toHandle, and incEst is built by making the hypothesis that all neighbours in the same ICS will increment their coreness. Note that the ICS induced by e and a are the same graph, which is the sub-graph induced by $\{a, c, d, e, f\}$. Consequently, e considers, in $incEst$, that all its neighbors have coreness 3. On the other hand, the $incEst$ of node a maps c, d, and e with coreness 3 and b with 1 (b's coreness remains unchanged since it does not belong to the ICS induced by a).

- **At $t = k * \delta$.** Nodes a and e re-compute an estimation using $incEst$. Both have three neigbors of coreness 3 and therefore change their estimated coreness to 3. Flags are set back to false, est takes the values of $incEst$ (the hypothesis regarding neighbors incrementation are not discarded), and the event is propagated. a and e send $< a, 3, eventId >$ and $< e, 3, eventId >$ to $\{b, c, d, e\}$ and $\{a, d, f\}$, respectively. a and e are unaffected by the messages, b updates its knowledge of a, and c,d, and f update their knowledge and trigger $tryIncrement$. The situation is depicted in Figure 5.2b.

- **At $t = (k + 1) * \delta$.** b recomputes its coreness but no change occurs. Nodes c, d and e handle the potential incrementation. c and d manage to increment and behave similarly to a and e in the previous step. Node f, however, having only two neighbors, can not increment its coreness to 3. The local knowledge remains est, $incEst$ is discarded, and f sends $< f, 2 >$ to $\{e, c\}$. e and c update their estimation of f: the hypothesis according to which f would have incremented its coreness is invalidated – see Figure 5.2c.

- **At $t = (k + 2) * \delta$.** Nodes e and c re-compute their estimation due to the modification of their knowledge regarding node f. They both have exactly two neighbors with an estimated coreness of 3 and one of 2 –node f. They both change their estimation to 2 and notify about this modification by sending $< e, 2 >$ and $< c, 2 >$ to $\{a, d, f\}$ –see Figure 5.2d.

- **At $t = (k + 3) * \delta$.** Node f recomputes its estimation but it remains unchanged. Nodes a and d, however, update their estimated coreness to 2 and notify their neighbors with the messages $< a, 2 >$ and $< d, 2 >$, respectively – see Figure 5.2e.

- **At $t = (k + 4) * \delta$.** Nodes a, b, c, d, and e recompute their estimation due to the modification of a and/or d. Nevertheless, all estimations remain unchanged; the algorithm has converged and the event $eventID$ has been fully handled as depicted in Figure 5.2f.

5.5 ANALYTICAL AND EXPERIMENTAL STUDY

In this Section, we perform an extensive study of our P2P algorithm both analytically and experimentally. Firstly, and for each possible type of perturbation, we show that the algorithm converges. We then provide upper and lower bounds for its analytical complexity in terms of time, exchanged messages and local computations and prove its correctness. Finally, we transpose the complexity study in the real world by performing an analysis on real online social networks and datasets.

5.5.1 *Analytical study*

5.5.1.1 *Complexity: remarks and notations*

In the following, we provide an analytical complexity analysis for the maintenance routines that correspond to each of the four events presented in Sec. 5.4.2.1: node addition (na), node deletion (nd), edge addition (ea) and edge deletion (ed) whose complexity is noted $C_{na}(v)$, $C_{nd}(v)$, $C_{ea}(e)$, and $C_{ed}(e)$, respectively. v corresponds to a node $v \in V$ and e to an edge $e \in E$ of graph $G = (V, E)$.

For each kind of event, the complexity is studied w.r.t. three aspects:

1. *time* – $C^{t}_{event}(el)$: it refers to the number of rounds necessary for the algorithm to converge, the duration of a single round being δ. In this section, we assume that the transmission time of a message from a neighbor to another is less than δ, so that a message sent in round i will always be treated in round $i + 1$;

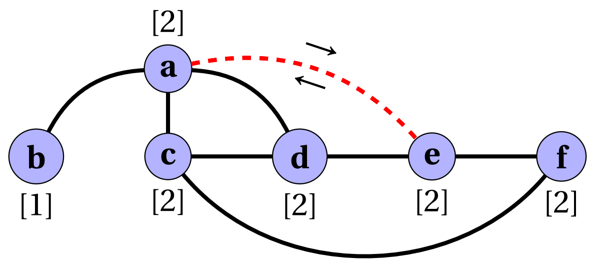
2. *exchanged messages* – $C^{m}_{event}(el)$: refering to the sum of all exchanged messages. Considered messages are those presented in Sec. 5.4.2.3, of type $< v >$, $< v, k >$, or $< v, k, eventId >$;

3. *local computations* – $C^{c}_{event}(el)$: refering to the sum of local coreness re-estimation through $\mathtt{computeEstimate}$.

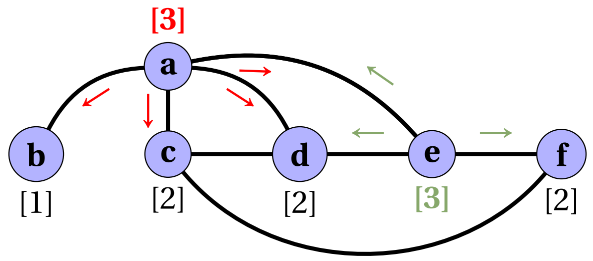
with $event \in \{na, nd, ea, ed\}$ and el being the concerned element either a node v or an edge e.

5.5.1.2 *Edge suppression*

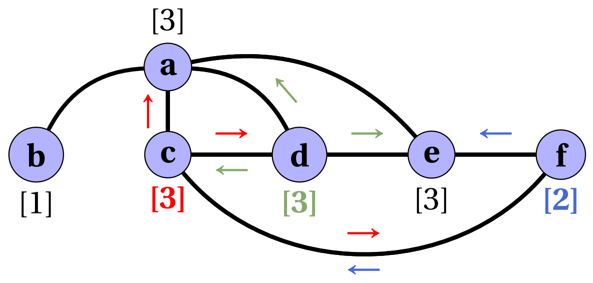
PRELIMINARIES.



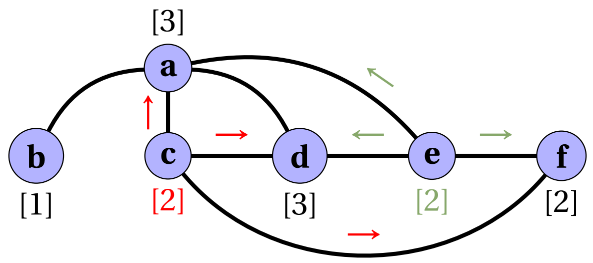
(a) At (k-1)*$\delta \leq t < k * \delta$.



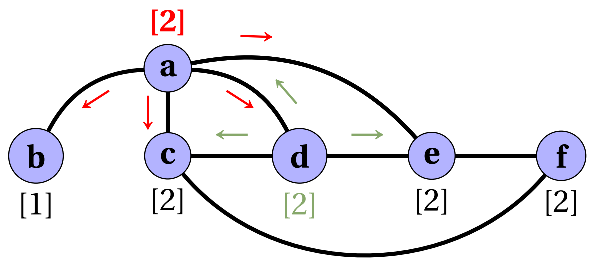
(b) At $t = k * \delta$.



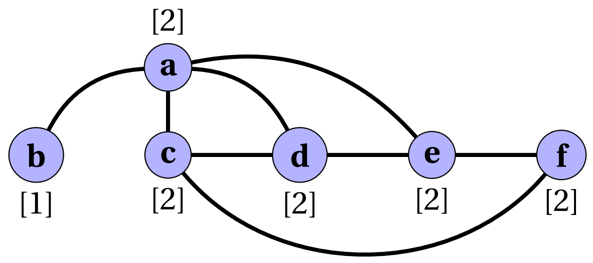
(c) At $t = (k + 1) * \delta$.



(d) At $t = (k + 2) * \delta$.



(e) At $t = (k + 3) * \delta$.



(f) At $t = (k + 4) * \delta$.

Figure 5.2: **A simple example describing the dynamic part of the algorithm.**

Let us consider the deletion of some edge e between nodes v and u. Let G and G' be the considered graph before and after the addition of e, respectively. In the complexity analysis, and so as to isolate consequences of the perturbation, we consider that the system is stable prior to the perturbation.

According to the k-core update theorem (see Theorem 5.3), the nodes that may be affected by the deletion of e belong to $ICS(v)$ or $ICS(u)$ or $ICS(v) \cup ICS(u)$. In particular, and by design of the algorithm, the nodes that will decrement their coreness belong to some sub-graphs:

Definition 5.4 (Decrementation Induced Core Subgraph (DICS))**.** *The* $DICS = (V_D^v, E_D^v)$ *induced by some vertex* v *for a graph* $G = (V_G, E_G)$*, noted* $DICS(v, G)$*, is the maximal (induced) connected subgraph of the core subgraph induced by* v *on* G *(*$ICS(v, G)$*) such as:*

1. $\forall u \in V_I^v$*,* $d_{ICS(v)} + |\{ (u' \in V_G) \mid (u' \notin V_I^v) \wedge ((u, u') \in E_G) \wedge (k(u') > k(v)) \}| = k(v)$*; all nodes of DICS(v) have exactly* $k(v)$ *neighbors that are either in* G *with a coreness greater than* $k(v)$ *or in* $ICS(v)$*.*
2. $v \in V_{PI}^v \vee V_{PI}^v = \emptyset$*.*

Indeed, the extremity with the lowest coreness decrements its estimation if it previously had exactly $k_G(v)$ neighbors of coreness greater or equal to $k_G(v)$ since it now has $k_G(v)$-1 such neighbors. In turn, its neighbors that had exactly $k_G(v)$ neighbors of coreness greater or equal to $k_G(v)$ now have $k_G(v) - 1$ such neighbors and will decrements their estimation before propagating the perturbation.

Example. Let us illustrate the above with an example. For this purpose, we consider the graph G illustrated in Fig. 5.1. Suppose that the edge between nodes c and h is deleted. According to the k-core update theorem and as $k_G(h) < k_G(c)$, the nodes that may be affected belong to $ICS(h)$. $ICS(h)$ contains 7 nodes: $V_I^h = \{g, h, i, j, k, l, m\}$. In reality, the nodes that will be affected belong to $DICS(h, G) = \{h, j\}$. Node h has exactly one neighbor in $ICS(h)$ (node j) and one neighbor in G of coreness 3 (node c). Similarly, node j is a neighbor of h in $ICS(h)$ and has exactly $k_G(h) = 2$ neighbors that are both in $ICS(h)$ (i.e., nodes h and k). Node k, however, has 3 neighbors in $ICS(h)$ and is therefore not in $DICS(h, G)$. After the edge deletion, node h has $k_G(h) - 1 = 1$ neighbor and will decrement its estimation before propagating the perturbation. The perturbation reaches j which now has $k_G(h) - 1 = 1$ neighbors of coreness $k_G(h) = 2$. It will also decrement its estimation and propagates the perturbation that will not lead to further decrementation (as h has already been impacted and k will not be).

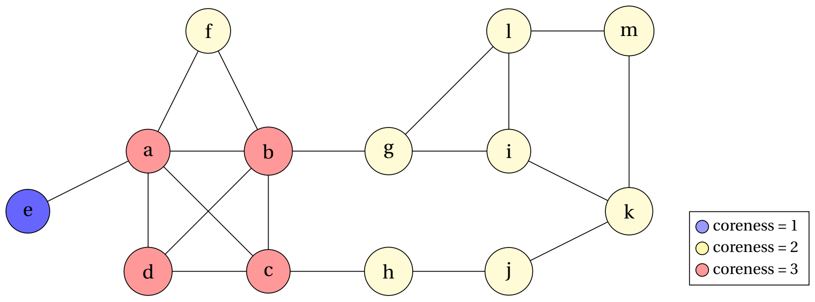


Figure 5.1: Example graph.

COMPLEXITY.

The deletion of an edge creates a perturbation that is propagated within the concerned DICS as nodes in the DICS broadcast the perturbation until reaching nodes outside of the DICS whose coreness remains unchanged.

Time. Here, the time required for a perturbation to be propagated is equivalent to the distance it travels in the graph thanks to the hypothesis that δ is greater than the communication time between neighbors. Thus, with $\epsilon(v, G)$ the eccentricity of v in G, the number of round taken by initial perturbation to reach all concerned nodes is:

$$C^t_{ed}((v,u)) = \epsilon(v, DICS(v,G)) + 1 \;\; \text{if } k_G(v) < k_G(u)$$
$$C^t_{ed}((v,u)) = max(\epsilon(v, DICS(v,G)), \epsilon(u, DICS(u,G))) + 1 \;\; \text{if } k_G(v) = k_G(u) \tag{5.1}$$

Messages. All nodes in the concerned DICS(s) will be affected by the decrementation and will broadcast their new coreness. Thus, the number of messages exchanged in order to calculate the updated coreness after an edge deletion is:

$$\begin{aligned} C^m_{ed}((v,u)) &= \sum_{n \in V^v_D} d_G(n) \;\; \text{if } k_G(v) < k_G(u) \\ C^m_{ed}((v,u)) &= \sum_{n \in V^v_D \cup V^u_D} d_G(n) \;\; \text{if } k_G(v) = k_G(u) \end{aligned} \tag{5.2}$$

Local computations. By design of the algorithm:

- Upon deletion of an edge, both extremities re-compute their coreness estimation.
- Apart from that, no computation can occur in absence of messages.

We thus trivially have:

$$C^c_{ed}((v,u)) \leqslant 2 + C^m_{ed}((v,u)) \tag{5.3}$$

CORRECTNESS.

From the locality theorem (see Thm 5.1), it is obvious that, for the computation of one's coreness, a non-existant neighbor is equivalent to a neighbor of coreness 0[*]. Therefore, both extremities of the deleted edge have an accurate representation of their neighborood after setting their coreness estimate of the other extremity at 0.

From there, the process can be seen as regular event of the static algorithm where a node is notified by one of its neighbor of a decrease in the latter's coreness estimation. In particular, the liveness property of the algorithm is conserved. Therefore, the edge deletion process is correct as the static algorithm is itself correct [125]. Note that this is true even in the presence of concurrent perturbations and regardless of the state of the system (stable or not) when the event occurs.

5.5.1.3 *Node suppression*

COMPLEXITY.

A node suppression is equivalent to multiple edge deletion as all edges incident to it are suppressed. We thus trivially have:

$$\begin{aligned} C^t_{nd}(v) &\leqslant max_{\{u \in V_G | (v,u) \in E_G\}} C^t_{ed}((v,u)) \\ C^m_{nd}(v) &\leqslant \sum_{\{u \in V_G | (v,u) \in E_G\}} C^m_{ed}((v,u)) \\ C^c_{nd}(v) &\leqslant \sum_{\{u \in V_G | (v,u) \in E_G\}} C^c_{ed}((v,u)) \end{aligned} \tag{5.4}$$

CORRECTNESS.

The node suppression process is correct for the exact same reason that the edge suppression process is. It is similar to a regular event of the static algorithm where a node notifies all of its neighbors of a decrease in its coreness estimation.

5.5.1.4 *Edge addition*

PRELIMINARIES.

Let's consider the addition of some edge e between nodes v and u. Let G and G' be the considered graph before and after the addition of e, respectively. In the complexity analysis, and

[*] Note that it is not normally possible for a node to have a neighbor of coreness 0 as only isolated nodes have a coreness equal to 0.

so as to isolate consequences of the perturbation, we consider that the system is stable prior to the perturbation.

According to the k-core update theorem (see Theorem 5.3), the nodes that may be affected by the addition of e belong to $ICS(v)$ or $ICS(u)$ or $ICS(v) \cup ICS(u)$. In particular, in our algorithm, the nodes that can possibly manage to increment their coreness are those belonging to the following subgraph:

Definition 5.5 (Potential Incrementation Induced Core Subgraph (PIICS)). *The* $PIICS = (V^v_{PI}, E^v_{PI})$ *induced by some vertex* v *for a graph* $G = (V_G, E_G)$*, noted* $PIICS(v, G)$*, is the maximal (induced) connected subgraph of the core subgraph induced by* v *on* G *(*$ICS(v, G)$*) such as:*

1. $v \in V^v_{PI}$.
2. $\forall u \in (V^v_I \backslash v)$, $d_{ICS(v)} + |\{ (u' \in V_G) \mid (u' \notin V^v_I) \wedge ((u, u') \in E_G) \wedge (k(u') > k(v)) \}| > k(v)$*; all nodes of PIICS(v) have at least* $k(v) + 1$ *neighbors that are either in* G *with a coreness greater than* $k(v)$ *or in* $ICS(v)$*.*

Indeed, nodes of the PIICS are the one that can increment their coreness estimation through the computeEstimate routine. Note that the incrementation is not necessarily stable: they may go back to their initial coreness afterwards.

Example. Let us assume that an edge is added between nodes b and l in our example graph (see Fig. 5.1). According to the k-core update theorem and as $k_G(b) < k_G(l)$, the nodes that may be affected belong to $ICS(l, G)$. $ICS(l, G)$ contains 7 nodes: $V^l_I = \{g, h, i, j, k, l, m\}$. In reality the nodes that will be affected belong to $PIICS(l, G)$ and are $\{g, i, k, l\}$. Nodes i and k have 3 neighbors that are in $ICS(l, G)$ and node g has 2 neighbors in $ICS(l)$ and one neighbor (node b) in G with a coreness greater than $k_G(b) > k_G(l)$.

COMPLEXITY.

The best case scenario is quite trivial. It happens when extremities have different coreness numbers and the PIICS induced by the extremity with the lowest coreness is limited to said extremity. Let's assume that v is the extremity with the lowest coreness. This case is characterized by $k_G(v) < k_G(u) \wedge PIICS(v) = (\{v\}, \emptyset)$.

After edge addition, tryIncrement is triggered on v. As isChanged is set to true, v re-computes its coreness, fails to increment it, and notifies its $d_{G'}(v)$ $(= d_G(v) + 1)$ neighbors by broadcasting the message $< v, k_G(v) >$. Neighbors won't be affected as none did any hypothesis regarding a potential incrementation of v's coreness.

Therefore:

$$\begin{aligned} C^t_{ea}((v,u)) &\geqslant 2 \\ C^c_{ea}((v,u)) &\geqslant 1 \\ C^m_{ea}((v,u)) &\geqslant d_{G'}(v). \end{aligned} \tag{5.5}$$

Worst-case scenario. Similarly to edge deletion, to maximize the number of potentially affected nodes, both extremities should have the same coreness in G so that both ICS(v, G) and ICS(u, G) will be affected. Hence, the worst case happens when $k_G(v) = k_G(u)$. Note that, in this case, ICS(v, G′) = ICS(u, G′).

The addition of an edge creates a perturbation that is propagated within the concerned PIICS. Nodes in the PIICS propagate the perturbation that eventually reach some nodes in the ICS (but not in the PIICS) that fail to increment. If enough nodes fail to increment, secondary, inverse, perturbations are propagated within some subgraph of the PIICS as nodes will decrement their estimations and propagate the secondaty perturbations.

To maximize the number of messages and computations, all nodes that can possibly affected should indeed be. They should also be affected in the worst possible way: successfully increment during one round, and then decrement.

Time. As noted previously, the time required for a perturbation to be propagated is equivalent to the distance it travels in the graph. Thus, regarding time, the initial perturbation (incrementation) will take at most $max(\epsilon(v, \text{PIICS}(v, G)), \epsilon(u, \text{PIICS}(u, G))) + 1$ rounds to reach all concerned nodes.

In the worst case scenario, the node being the furthest away from the initial source of the perturbation will trigger a secondary perturbation that will:

- reach the PIICS in 1 round.
- be propagated within the whole PIICS. Let $diam(G)$ be the diameter of the graph G. The perturbation traverses the PIICS in $diam(\text{PIICS}(u, G')$ rounds.
- be broadcasted one last time by the last node to be reached in the PIICS.

Therefore:

$$\begin{aligned} C^t_{ea}((v,u)) &\leqslant \epsilon + diam(\text{PIICS}(u, G')) + 3 \\ with\ \epsilon &= max(\epsilon(v, \text{PIICS}(v, G)), \epsilon(u, \text{PIICS}(u, G))) \end{aligned} \tag{5.6}$$

Messages. A node n belonging to PIICS may be affected by an edge modification and will, in a worst case scenario:

- broadcast a message $< n, k_G(v) + 1, eventID >$ to its neighbor to propagate the incrementation;
- broadcast a message $< n, k_G(v) >$ to its neighbor to propagate the decrementation;
- upon reception of an incrementation message sent by a neighbor in the PIICS reply with $< n, k_G(v) >$ to signify that the event has been handled but the incrementation failed. Since n had to handle the perturbation before sending such messages, this can occur:
  - at most once per neighbor in PIICS minus one (the one whose message lead to the event being handled) if n is neither u nor v,
  - at most once per neighbor in PIICS if n is neither u nor v (the sources of the perturbation).

Neighbors of n that are not in the PIICS will receive messages and may also send messages in response. Let $N_{PI}(v, G)$ be the neighbors of nodes in PIICS(v, G) that are not themselves in the PIICS(v, G):

$$N_{PI}(v, G) = \{n \in (V_G \backslash V_{PI}(v, G)) | (n' \in V_{PI}^{v} \wedge ((n, n') \in E_G)\}.$$

Example. Considering again an edge addition between nodes b and l in our example graph. As discussed PIICS(l, G) is the graph induced by $\{g, i, k, l\}$ which makes $N_{PI}(v, G) = \{b, m, j\}$.

Nodes in N_{PI} but not in the ICS will not send any message. However, nodes $n \in N_{PI} \cap V_I$ will in the worst case:

- broadcast a message $< n, k_G(n) >$ after failing to increment.
- upon reception of an incrementation message sent by a neighbor in the PIICS reply with $< n, k_G(n) >$ to signify that the event has been handled but the incrementation failed. This can occur at most once per neighbor in PIICS minus one (the one whose message lead to the event been handled).

Thus, we conclude the upper bound of the number of messages exchanged in order to calculate the updated coreness after an edge modification :

$$\begin{aligned} C^m_{ea}((v,u)) \leqslant & (\sum_{n \in V^v_{PI} \cup V^u_{PI}} 2 * d_G(n) + (d_{PIICS(v,G')}(n) - 1)) + 2 \\ & + \sum_{n \in S} d_G(n) + |\{n' \in V^v_{PI} \cup V^u_{PI} | (n,n') \in E_G\}| - 1 \\ & with\ S = (N_{PI}(v,G) \cap V_I(v,G)) \cup (N_{PI}(u,G) \cap V_I(u,G)) \end{aligned} \quad (5.7)$$

Local computations. By design of the algorithm:

- Upon addition of an edge, the extremity with the lowest coreness re-computes its coreness estimation. If both extremities have the same coreness, they both re-compute their coreness estimation.
- Apart from that, no computation can occur in absence of messages.

We thus trivially have:

$$C^c_{ea}((v,u)) \leqslant 2 + C^m_{ea}((v,u)) \quad (5.8)$$

A finer upper bound can however be found. Indeed:

- For nodes in the PIICS, the initial perturbation triggers a single computation, no matter how many messages of type $< v, k, eventID >$ are received, as long as they all contain the same $eventID$.
- As seen previously, when a node broadcasts a message of type $< v, k, eventID >$, it may get replies of type $< v, k >$ from neighbors that already handled the event but failed to increment. These replies are received during the same round and can therefore trigger a single re-computation.

Therefore:

$$\begin{aligned} C^c_{ea}((v,u)) \leqslant\ & 2 \\ & + \sum_{n \in V^v_{PI} \cup V^u_{PI}} 2 * d_G(n) + 1 \\ & + \sum_{n \in S} d_G(n) + |\{n' \in V^v_{PI} \cup V^u_{PI} | (n,n') \in E_G\}| - 1 \\ & with\ S = \{n \in V^v_I \cup V^u_I | (n' \in V^v_{PI} \cup V^u_{PI}) \wedge ((n,n') \in E_G)\} \end{aligned} \quad (5.9)$$

CORRECTNESS

We have seen that the edge addition process has a bounded complexity. It is therefore terminating. Let's demonstrate that it stops with the correct coreness. We assume here that the system was stable prior to edge addition (i.e., all vertices did estimate their coreness correctly). By contradiction, let's assume that the process does not accurately update the estimation of each node. Nodes affected by the process have their estimated coreness either unchanged or incremented by one. According to the update theorem, an edge addition may lead to the incrementation of some nodes. Therefore, there are basically two possibilities; (1) some node(s) have an incremented coreness estimation while their actual coreness is unchanged, (2) some node(s) have an incremented coreness estimation while their actual coreness is unchanged. Let $k-1$ be the coreness of the potentially affected node prior to the edge addition.

(1) Erroneous equality. Let's assume that some vertices did not increment their estimation while they should have and let $V^=$ be the set of such vertices. They either (i) did not initially increment their estimations or (ii) did increment their estimations but decrement it at some point during the process. Estimation updates occurs at the end of a round, for each $u \in V^=$ one of these two events thus occurred at the end of some round r_u. Let v be a node $V^=$ such as $\forall u \in V^=, r_v \leqslant r_u$. *(i) No initial incrementation.* Let's assume that v receives a message of type $< u, k, eventID >$ in round r_v but did not increment its coreness. Since we consider the initial incrementation attempt, $eventID$ should not be in $handled$. Since we assumed that v's coreness increments due to the edge addition, according to the update theorem, its coreness should be $k-1$ prior to the perturbation. Therefore, due to the stability hypothesis, v's estimation at the beginning of r_v is $k-1$ and $tryIncrement$ is triggered on v. At the end of the round, v recomputes its coreness with $incEst$ and, by hypothesis, v's estimation remains unchanged. Therefore, it has less than k neighbor with an estimated coreness of at least $k-1$ (neighbor of coreness $k-1$ are considered of coreness k in $incEst$). According to the core update theorem, it means that after the edge addition v has necessarily less than k neighbor of coreness k. According to the locality theorem, this contradicts the hypothesis that v's coreness increments after the edge addition. *(ii) Erroneous decrementation.* Let's assume that v decrements its estimation at the end of round r_v. Necessarily, it has received some message(s) $< u_i, k-1, eventID >$ from a set of neighbors u_i in that round. These messages notify an estimation update calculated at the end of round $r_v - 1$. Since v decrements its coreness at the end of r_v, its set est contains (after its update according to these messages) less than k neighbors of estimated coreness at least k due to the update. By definition of v and r_v, coreness updates emitted before the end of round r_v are legitimate and v has indeed less than k neighbors of coreness at least k after the edge addition. According to the locality theorem, this contradicts the hypothesis that v's coreness increments after the edge addition. Therefore, it is not possible for a node to erroneously *not increment* its coreness estimation during the edge addition process.

(2) Erroneous incrementation. Let's assume that some vertices did increment their estimation while they should not have. Let V^+ be the set of such nodes and let v be a node in V^+. Note that, in spite of the incrementation hypotheses, each node has a consistent view of its neighbors' estimations at the end of the process (i.e., the values stored in est are consistent with local estimations). By hypothesis, v's estimation is k. Since each time est is modified the node re-computes its coreness, v has at least k neighbors of estimated coreness greater or equal to k. Let K be the largest connected subgraph containing v and whose nodes all have an estimated coreness greater or equal to k. Forall $v_k \in K$, either

- v_k is of coreness k. It has at least k neighbors of coreness greater or equal to k according to the locality theorem who are in K by definition of K.
- $v_k \in V^+$. As seen before, it therefore has at least v_k neighbors of estimated coreness greater or equal to k who are in K by definition of K.

Therefore, K is contained in a k-core by definition. Since v is in K, v is of coreness at least k. This is a contradiction . Therefore, it is not possible for a node to erroneously *not increment* its coreness estimation during the edge addition process.
Thus, the edge addition process is correct.

5.5.1.5 *Node addition*

In the case of node addition, the study is quite trivial: obviously, the addition of some node v with no edge does not impact the coreness of other nodes as v's neighborhood is empty.

A simple initialization is performed on v, taking a single round. During the initialization, $\mathsf{isChanged}$ is set to false and the procedure $\mathsf{computeEstimate}$ is thus triggered once. The coreness of v equals its degree, 0, and the estimated coreness

Dataset	NetHEPT	WikiVote	Email-Enron	Epinions
# Nodes	15K	7K	34K	75K
# Edges	62K	103K	180K	405K
k_{max}	31	53	43	67

Table 5.1: Properties of the real-world graphs used.

is not modified. No messages are exchanged as v has no neighbours. Therefore:

$$\begin{aligned} C^t_{na}(v) = C^c_{na}(v) &= 1 \\ C^m_{na}(v) &= 0 \end{aligned} \tag{5.10}$$

5.5.2 *Complexity: Experimental Study*

In this subsection we perform an analysis on real online social networks as it is important to see how our analytical results translate in the real world. We specifically focus on *edge addition* as it is the most costly event and also the one that differs the most from the static version of the algorithm.

DATASETS We have performed experiments with several real-world networks. Table 5.1 presents the datasets used in our study, along with some relevant properties. A detailed description of the datasets is presented in Section 2.4 of Chapter 2.

RESULTS Tables 5.2 to 5.5 present for every of the aforementioned datasets the *average ICS size*, *average PIICS size*, *average ICS diameter* and *average PIICS diameter* for every coreness group. We specifically focus on the aforementioned values as those are the ones related to the complexity of an edge addition event. The size of the aforementioned subgraphs define the number nodes that will be impacted while their diameter is related to the upper bound of the time complexity for such an event.

We have seen that the total number of messages that will be sent in case of an edge addition is approximately bounded by three times the sum of the degree of nodes in the PIICS induced by the respective vertex. We observe that for the NetHept dataset the largest *average PIICS* detected is the one for nodes in the 5-core subgraph and consists on average of 206.91 nodes. This is just a 1.36% fraction of the total number of nodes in the dataset. Similar values are observed for the rest of the datasets, even for the largest of the four chosen datasets for this study, Epinions, the the largest *average PIICS* detected is the one for nodes in the 53-core subgraph and consists on average of 324.04 node which

coreness	average ICS size	average PIICS size	average ICS diameter	average PIICS diameter
1	1.79	1.14	0.75	0.14
2	3.44	1.69	1.37	0.55
3	17.33	6.20	4.05	2.03
4	109.58	22.59	14.81	3.00
5	376.35	206.91	12.64	9.48
6	256.05	152.94	11.57	9.26
7	147.37	99.27	7.80	8.12
8	191.58	154.45	5.19	4.22
9	10.00	1.00	1.00	0.00
18	19.00	1.00	1.00	0.00
20	21.00	1.00	1.00	0.00
23	24.00	1.00	1.00	0.00
31	32.00	1.00	1.00	0.00

Table 5.2: ICS and PIICS statistics for NetHept network

is just a 0.63% fraction of the total number of nodes. From this analysis we conclude that it is a lot less costly to use our P2P algorithm, in terms of number of messages that will be sent, than Montresor's algorithm [125] which in the case of an edge addition, requires that all nodes broadcast a message at least once in order to calculate the new coreness estimations.

In terms of temporal complexity, again for the case of an edge addition, the number of steps that will be performed by our algorithm is approximately bounded by three times the diameter of the PIICS. We observe that for the WikiVote dataset the largest *average PIICS diameter* detected is the one for nodes in the 51-core subgraph and equals to 13.76. For the Email-Enron dataset the largest *average PIICS diameter* detected is the one for nodes in the 39-core subgraph and equals to 10.45.

5.6 SECURITY AND PRIVACY ANALYSIS

Whenever confronted with sensitive data, such as social networking data, security and privacy questions come to mind. Most importantly, using pseudonyms (i.e. removing node identifiers) is not sufficient to provide privacy to social network users [128]. In this section, we perform a security and privacy analysis of our approach, by overviewing existing privacy models for graphs, and by showing how such models can be used in our context to anonymize the data. We then perform a quality analysis to show that our coreness computation remains robust even when running on anonymized graphs. We thus discuss the impact of security and privacy, which depends on anonymization parameters, on the quality of the output, measured by comparing the results of our algorithm executed on the initial and anonymized graphs.

First of all, we will describe the general anonymization process for graphs, and a state of the art methodology to recon-

coreness	average ICS size	average PIICS size	average ICS diameter	average PIICS diameter
1	1.05	1.02	0.03	0.02
2	1.58	1.15	0.48	0.13
3	3.24	1.67	1.03	0.47
4	3.85	2.00	1.05	0.56
5	3.59	1.91	1.08	0.52
6	4.76	2.35	1.38	0.68
7	9.95	6.33	1.77	1.22
8	5.61	3.72	1.34	0.84
9	6.72	3.90	1.94	1.10
10	3.72	2.62	1.28	0.72
11	4.48	3.35	1.18	0.77
12	3.73	2.67	1.20	0.64
13	3.16	1.93	0.96	0.42
14	5.86	4.02	2.21	1.25
15	2.66	1.60	1.20	0.39
16	3.37	1.89	1.37	0.49
17	2.74	1.79	1.25	0.55
18	2.61	1.75	1.28	0.63
19	2.09	1.30	0.98	0.24
20	4.26	2.44	2.04	0.82
21	2.59	1.73	1.29	0.65
22	12.44	5.68	5.17	1.69
23	5.61	2.22	2.31	0.61
24	9.00	2.77	4.70	1.13
25	11.91	2.85	6.01	0.89
26	3.47	1.79	2.12	0.58
27	25.05	14.30	6.83	4.46
28	10.96	4.03	4.68	1.51
29	18.74	5.34	7.34	1.37
30	44.68	22.87	6.87	4.14
31	44.25	11.13	11.63	1.73
32	75.04	28.39	13.57	3.70
33	8.27	3.48	4.19	1.40
34	140.40	101.53	10.54	9.00
35	54.30	33.55	9.63	8.00
36	58.51	36.62	7.04	7.25
37	26.80	17.24	6.49	5.20
38	76.00	59.22	6.00	7.00
39	58.03	42.03	9.83	10.45
40	86.00	76.12	6.00	6.00
41	37.28	24.86	5.58	5.40
42	50.04	42.35	4.90	4.90
43	275.00	267.03	3.00	3.00

Table 5.3: ICS and PIICS statistics for the EmailEnron network

coreness	average ICS size	average PIICS size	average ICS diameter	average PIICS diameter
1	1.02	1.01	0.01	0.01
2	1.01	1.00	0.01	0.00
3	1.06	1.02	0.06	0.02
4	1.01	1.00	0.01	0.00
5	1.08	1.03	0.08	0.02
6	1.04	1.00	0.04	0.00
7	1.21	1.12	0.18	0.09
8	1.19	1.06	0.19	0.04
9	1.18	1.10	0.18	0.10
10	1.14	1.06	0.14	0.06
11	1.26	1.08	0.26	0.07
12	1.90	1.67	0.58	0.44
13	1.09	1.03	0.09	0.03
14	1.20	1.09	0.20	0.09
15	1.36	1.15	0.36	0.12
16	1.37	1.15	0.37	0.15
17	1.23	1.08	0.23	0.08
18	2.03	1.29	0.91	0.20
19	1.17	1.02	0.17	0.02
20	2.26	1.61	0.89	0.47
21	5.15	2.04	2.78	0.65
22	3.63	2.11	1.49	1.06
23	1.58	1.24	0.58	0.24
24	2.09	1.42	0.70	0.28
25	1.67	1.19	0.56	0.15
26	1.61	1.06	0.61	0.06
27	1.30	1.11	0.30	0.11
28	2.82	1.49	1.38	0.27
29	2.62	1.77	1.54	0.60
30	2.18	1.39	1.08	0.33
31	9.43	5.83	3.22	2.41
32	6.68	2.39	3.18	0.73
33	12.09	5.35	4.74	1.56
34	4.14	3.27	1.95	1.43
35	4.50	1.79	2.33	0.79
36	8.91	2.93	4.52	0.87
37	14.33	5.49	4.97	2.74
38	25.39	10.33	10.33	2.79
39	11.00	4.05	5.00	1.40
40	58.45	22.68	12.21	4.84
41	18.62	2.53	8.30	0.74
42	7.47	4.03	2.82	1.68
43	4.45	3.00	3.05	1.23
44	9.93	4.05	4.21	1.77
45	40.60	21.66	9.00	4.60
46	38.63	22.18	11.23	8.43
47	73.08	26.40	8.77	8.91
48	39.31	19.18	7.51	9.64
49	139.00	112.19	6.00	6.99
50	45.24	28.59	7.53	7.94
51	52.04	31.43	6.87	13.76
52	144.00	132.08	5.00	5.00
53	336.00	324.04	3.00	3.00

Table 5.4: ICS and PIICS statistics for the WikiVote network

coreness	average ICS size	average PIICS size	average ICS diameter	average PIICS diameter
1	1.33	1.12	0.24	0.12
2	1.73	1.24	0.56	0.21
3	1.92	1.38	0.60	0.24
4	1.87	1.35	0.63	0.27
5	2.02	1.43	0.66	0.28
6	2.11	1.54	0.67	0.35
7	2.07	1.44	0.68	0.29
8	2.13	1.52	0.73	0.34
9	2.80	1.83	1.01	0.44
10	2.23	1.49	0.80	0.27
11	3.39	2.63	0.83	0.46
12	2.26	1.68	0.70	0.33
13	1.80	1.34	0.57	0.28
14	2.71	1.87	0.92	0.48
15	2.01	1.40	0.73	0.28
16	2.79	1.62	1.32	0.40
17	1.72	1.33	0.56	0.25
18	4.38	3.02	1.46	0.80
19	4.86	3.71	1.18	0.48
20	2.77	1.54	1.19	0.36
21	2.97	1.89	1.35	0.55
22	8.71	6.40	1.95	1.23
23	2.29	1.33	1.07	0.27
24	5.60	1.89	3.37	0.64
25	2.14	1.38	1.14	0.38
26	2.46	1.48	1.18	0.33
27	2.44	1.70	1.08	0.50
28	4.04	1.89	2.13	0.72
29	2.78	2.01	1.09	0.52
30	3.13	1.78	1.56	0.56
31	3.47	2.00	1.88	0.80
32	9.23	2.65	3.55	0.77
33	5.97	2.73	2.52	0.99
34	5.71	2.48	3.23	1.19
35	3.60	2.09	1.52	0.75
36	6.58	2.29	3.40	0.54
37	46.31	8.00	10.72	1.62
38	20.65	5.50	5.73	1.01
39	3.83	1.93	2.07	0.70
40	13.52	3.65	5.43	1.05
41	28.64	15.62	7.99	3.73
42	7.98	3.95	3.70	1.43
43	31.39	10.65	7.90	3.18
44	28.77	9.88	7.77	2.20
45	114.53	94.06	6.64	7.95
46	19.67	8.77	6.33	2.68
47	19.82	6.68	5.76	1.71
48	11.24	3.62	5.74	1.21
49	23.81	10.70	6.91	3.77
50	81.93	41.35	11.84	13.36
51	20.57	5.98	7.02	1.00
52	36.80	13.97	10.43	3.38
53	51.25	9.52	10.14	1.47
54	63.91	43.86	7.84	4.84
55	96.02	56.90	7.92	7.19
56	53.68	23.08	9.11	5.32
57	80.07	53.38	6.83	11.48
58	72.03	42.64	6.91	10.88
59	11.91	4.88	6.18	2.33
60	56.19	39.13	8.56	6.39
61	40.14	15.59	11.45	6.36
62	77.17	55.92	6.77	5.58
63	38.19	19.86	6.71	4.29
64	73.00	52.75	6.00	5.84
65	36.15	21.95	6.65	7.17
66	231.00	217.06	4.00	4.00
67	486.00	477.02	3.00	3.00

Table 5.5: ICS and PIICS statistics for the Epinions network

struct the original while using its anonymized version. Our contribution in this chapter is to propose a secure protocol to apply state of the art graph anonymization techniques and subsequently to study the influence of state of the art graph anonymization on the quality of the results of our coreness computation algorithm.

5.6.1 *Attack Model*

The social network de-anonymization attacks considered in this work are those described in the work of Narayanan et al. [128]. In this context, the attacker $\mathscr{A}$ is assumed to know G_{aux} which represents some knowledge of the real graph topology G (partial or complete), and the attack consists in trying to map nodes of G_{aux} to nodes in $\tilde{G}$ which is the observed graph that the attacker can obtain during the execution of the algorithm. In a worst case scenario, one can assume that $G_{aux} = G$.

Obviously, in a real context, $G_{aux} \subset G$ and nodes in $\tilde{G}$ contain additional information that $\mathscr{A}$ wishes to append to his own knowledge of G_{aux}.

We consider the following kind of attackers in our scenario:

1. $\mathscr{A}_{int}$: The attacker is a member of the network. He knows his own neighbors and receives their messages. Therefore he knows the degree and the evolution of estimated coreness of its neighbors.

2. $\mathscr{A}_{ext}$: The attacker is outside of the network but manages to have access to all the exchanged messages. He knows which nodes are communicating with one another, as well as the degree and the evolution of estimated coreness of each node.

Our approach to security and privacy for both these attackers is similar : is the attacker able to reconstruct (part of) the original graph from the information that he has? Although exact graph reconstruction simply from the knowledge of the degrees of the graph is difficult, Narayanan and Shmatikov [128] have shown that it is possible to reconstruct parts of it, based on finding some "seed" nodes who have rare degrees, and progress from that point to de-anonymize a large portion of the graph. Thus, it is not acceptable for any attacker to obtain the exact degrees of the graph.

In case of $\mathscr{A}_{ext}$, one can easily see that if the algorithm runs on the real topology, then $\mathscr{A}_{ext}$ can reconstruct G without even using G_{aux} since $\tilde{G} = G$.

Therefore, in order to have $\tilde{G} \neq G$, our algorithm should obviously not run on the original graph but on a modified version, and any observation of the network should lead to seeing the anonymized topology and in no case G itself. In this case, the attacker knows $\tilde{G} \neq G$ and G_{aux} and tries to reconstruct G by mapping vertices from $\tilde{G}$ to G_{aux}. This model is equivalent to the **global surveillance attacker model** defined in [128].

In order to secure the coreness computation process, two complementary techniques are used: messages are all encrypted, so that no information may actually leak from the contents of the messages for an external observer (such as temporary coreness values) and 2) communications of the nodes must be modified so that the actual topology observed by the attackers is different from the real topology. Casas-Roma et al. [34] proposed a technique to produce such a graph $\tilde{G}$ by **switching, adding** and **deleting** edges of G. In particular they aim at preserving information quality, meaning that algorithms should have similar results whether they are run on $\tilde{G}$ or G. Different algorithms are more or less robust to such an anonymization process. We will show in what follows that our distributed coreness computation algorithm is very robust to anonymization.

In practice, we need to implement this anonymization model in a distributed context. This can be achieved by i) not broadcasting to a set of random neighbors (which corresponds to *deleting* an edge from the graph) and ii) by sending messages to random nodes of the graph which are not neigbours (which corresponds to *adding* edges to the graph). In a social network, these random nodes can be selected by simply contacting non-neigbour nodes in a connected component.

In the rest of this Section, we will assume that $\tilde{G} \neq G$ in other words, that the graph that can be observed by a global attacker has been anonymized with a specific process which we will describe next, and whose precise parameters will be explained in section 5.6.3. By running our algorithm on such a graph, we achieve the security and privacy objective. We will next show that our algorithm continues to produce good results in terms of coreness ranking, thus achieving the quality objective.

5.6.2 *Privacy and Information Quality*

In our case, the quality (i.e. robustness) of an algorithm is linked to the similarity of nodes' corenesses when computed on G and $\tilde{G}$. The objective is that the distribution of nodes with regards to their coreness is similar when running on both G and $\tilde{G}$.

Indeed, the applications that we consider are interested in selecting nodes with highest coreness values, regarless of this actual value, since coreness is studied in particular to identify highly influential individuals. Specifically, the nodes that belong to the densest k-core subgraph are proved to be more efficient information spreaders [98]. To assess the quality of $\tilde{G}$, we will therefore compare the resulting nodes when influential individuals are chosen from the different versions of the graph. Those influential individuals are obviously chosen from the top k-core subgraph (or subgraphs depending on the number of individuals required) as previously suggested.

In other words, we do not necessarily want nodes to have the *exact same coreness values* when computing our algorithm on G and $\tilde{G}$, but we want sets of nodes ranked by coreness to be the same.

5.6.3 *Experimental results*

In this Section we present experimental results concerning the quality of the influential entities acquired from anonymized versions of a graph.

5.6.3.1 *Datasets and methodology*

DATASETS For the experiments of this Section we have used the NetHept, WikiVote and Email-Enron datasets, some relevant properties of which are presented in Table 5.1. A detailed description of the datasets is presented in Section 2.4 of Chapter 2.

ANONYMIZATION PROCESS We produce anonymized versions of the datasets by randomly adding and/or deleting edges from the initial graph. The edges are added/deleted based on two parameters α and β. They represent the percentage of the total edges to be added and deleted respectively. We specifically consider two different settings:

- *Setting* I: Only random deletions of edges occur. In this case α is obviously equal to 0 whereas β can take values from 0.1 to 0.9.
- *Setting* II: Both random additions and deletions of edges occur. Both α and β take values from 0.1 to 0.9.

Definition 5.6 (*Anonymization Security* AS)**.** *The anonymization security* AS *of an algorithm is measured by* α *in Setting* I *and by the couple* (α, β) *in Setting* II.

Obviously in Setting II, values of AS are not necessarily comparable, but form a lattice.

METHODOLOGY Experimentally, as the edges are added/deleted in a random way, for each anonymized version (i.e., for a specific choice of an AS value) we repeat the process 10 times and report average behavior.

In order to evaluate the quality of influential entities that are acquired from the anonymized versions of the graphs we define the following metrics.

Definition 5.7 (*Quality* q)**.** *The Quality* q *of the influential nodes computed using an anonymized version of a graph is defined as the number of nodes returned that are indeed influential in* G*. In other words, it is the number of influential nodes that are common whether they were computed using the anonymized graph* $\tilde{G}$ *or the original graph* G*.*

The most common practice in respective applications, is to require a specific amount of influential nodes of the graph. Let us define as X% the percentage of influential nodes required out of the total number of nodes of the graph. We note z the actual number of influential nodes required, hence the following definition of Normalized Quality q_N.

We note as G_z the top z nodes computed using our algorithm running on G. We note $\tilde{G}_z$ the top z nodes computed using our algorithm running on $\tilde{G}$. As some nodes may have the same coreness value, it is possible that $|G_z|> z$ and/or $\tilde{G}_z > z$.

Definition 5.8 (*Normalized Quality* q_N)**.** *The Normalized Quality* q_N *of the influential nodes acquired from an anonymized version of the graph, is the number of influential nodes that are common when chosen from either version of the graph (*$\tilde{G}$ *or* G*) averaged over the number* z *of influential nodes required.*

Obviously q_N takes values between 0 and 1. It should be noted that the value that will be reported in the results is the average of the values produced by the different iterations performed for the production of each anonymized version of the graph.

A first idea would be simply to compute q_N by using the Jaccard index :

$$q_N = \frac{|G_z \cap \tilde{G}_z|}{|G_z \cup \tilde{G}_z|} \tag{5.11}$$

However, in fact our algorithm does not return a set of z nodes, but instead a *partially ordered list*. We should take this order into account when deciding which nodes to select. Thus in fact, q_N can be better calculated as follows:

$$q_N = \frac{q_1 + q_2}{z} \tag{5.12}$$

where q_1 is the *base quality* and q_2 the *randomized quality*. In order to define q_1 and q_2, let us denote as V_N and $V_{i<N}$ the set of nodes that belong in the N_{th} and the $N-1$ top coreness groups of the original graph respectively. The equivalent sets of nodes of a modified version of the graph are denoted as V'_N and $V'_{i<N}$. Let Y be the natural number such that $|V_{i<Y}| < z \leqslant |V_{i<Y+1}|$. Equivalently Y' is the natural number such that $|V'_{i<Y'}| \leqslant z < |V'_{i<Y'+1}|$. Then the *base* and *randomized quality* can be calculated as follows:

$$q_1 = |V'_{i<Y'} \cap V_{i<Y+1}| \tag{5.13}$$

$$q_2 = |S| \frac{|P \cap V_{i<Y+1}|}{|P|} \tag{5.14}$$

where $|S|$ is the number of nodes remaining to be picked after having selected $|V'_{i<Y'}|$ nodes (i.e., $|S| = z - |V'_{i<Y'}|$). In this case where $|S|>0$, $|V'_{i<Y'+1}| > |V_{i<Y+1}| > |V'_{i<Y'}|$. Let $A = V'_{i<Y'} \cap V_{i<Y+1}$ which defines the set of correct influential spreaders that we need to select based on their ranking. Then $P = V'_{i<Y'} \setminus A \cup V'_{i=Y'}$ which defines the set of all nodes that have not already been considered and that will be considered now.

q_2 actually represents the mean of a hypergeometric distribution [141]. The latter represents a discrete probability distribution describing the probability of ℓ successes in m draws, without replacement, from a finite population of size M that contains exactly L successes, wherein each draw is either a success or a failure. The mean of such a distribution is calculated as follows:

$$Mean_{hg} = m\frac{L}{M} \tag{5.15}$$

In our case we are interested in knowing the mean of the probability of $m = z - |V'_{i<Y'}|$ draws without replacement from a population of size $|P|$ that contains exactly $|P \cap V_{i<Y+1}|$ successes (i.e., nodes that are indeed the correct top spreaders in the original graph). We could have simply considered the hypergeometric mean of sampling z individuals from $V'_{i<Y'+1} with |V'_{i<Y'+1} \cap V_{i<Y+1}|$ correct individuals. However, as we have a ranking of spreaders, we initially take all the $V'_{i<Y'}$ (computation of q_1), and simply sample the $V'_{i<Y'+1}$ ones (computation of q_2).

A toy example is presented in 5.1 in order to demonstrate the calculation of q_1, q_2 and q_N. The example depicts the nodes belonging to the top 3 coreness groups in the (a) original and the (b) modified graph respsectively. The $'$ was added to the notation of the coreness groups of the modified graph as the

coreness	nodes
k	b, h, j
k-1	a, c, d
k-2	n, k, l
⋮	

(a) ORIGINAL GRAPH

coreness	nodes
k'	a, b, h
k'-1	c, d, j, n
k'-2	k, l
⋮	

(b) MODIFIED GRAPH

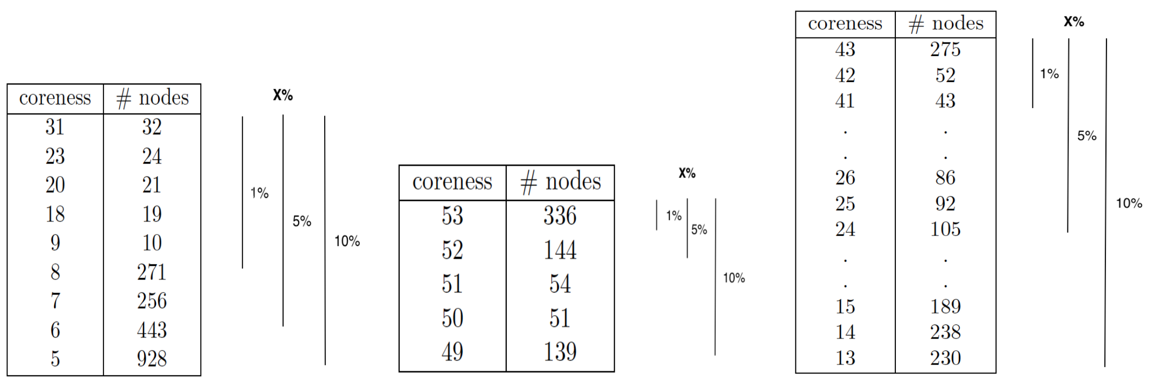
Figure 5.1: **Toy example to demonstrate the computation of q_1, q_2 and q_N which represent the *base*, *randomized* and *normalized quality* of influential nodes acquired from an anonymized version of a graph.** The top 3 corenesses (left column) and the nodes characterized by them (right column) are depicted for the (a) ORIGINAL GRAPH and the (b) MODIFIED GRAPH When $z = 3$ - when 3 influential nodes are required - $q_1 = 2$, $q_2 = 0$ and $q_N = 0.667$. Whereas when $z = 4$, then $q_1 = 2$, $q_2 = 0.75$ and $q_N = 0.6875$.

value k of the densest k-core subgraph may change after addition and/or deletion of edges. Let us consider the case where 3 influential nodes (i.e., $z = 3$) need to be picked from the modified graph. Then $V_{i<Y+1} = \{b, h, j\}$, $V'_{i<Y'} = \{a, b, h\}$ and $V'_{i<Y'} \cap V_{i<Y+1} = \{b, h\}$. Which means that $q_1 = 2$. As $|S| = z - |V'_{i<Y'}| = 0$ also $q_2 = 0$. Finally the *normalized quality* equals to $q_N = 2 + 0/3 = 0.667$. If $z = 4$, then $V_{i<Y+1} = \{b, h, j, a, c, d\}$, $V'_{i<Y'} = \{a, b, h\}$ and $V'_{i<Y'} \cap V_{i<Y+1} = \{b, h\}$ which results in $q_1 = 2$. In this case $|S| = 1$, $P = \{c, d, j, n\}$. Then the *randomized quality* equals to $q_2 = 1 * \frac{3}{4}$ which results in a *randomized quality* $q_N = 0.6875$.

5.6.3.2 *Evaluating the quality of influential spreaders on anonymized graphs*

Figure 5.3 depicts the *normalized quality* q_N of top influential nodes selected from anonymized versions of real datasets produced as *Settings* I and II suggest. The q_N values presented are calculated for different numbers of nodes - X% represents the percentage of the nodes required out of the total number of nodes of the dataset. It should be noted that even if the proportions of edges added are the same with those that are deleted in *Setting* II, the actual edges deleted are different from those that were added in order to create the anonymized versions of the graph.

We observe that for the NetHept dataset, the q_N values are greater than 65% up until there are 50% of edge deletions (*Set-*



coreness	# nodes
31	32
23	24
20	21
18	19
9	10
8	271
7	256
6	443
5	928

(a) NetHept

coreness	# nodes
53	336
52	144
51	54
50	51
49	139

(b)WikiVote

coreness	# nodes
43	275
42	52
41	43
.	.
.	.
26	86
25	92
24	105
.	.
.	.
15	189
14	238
13	230

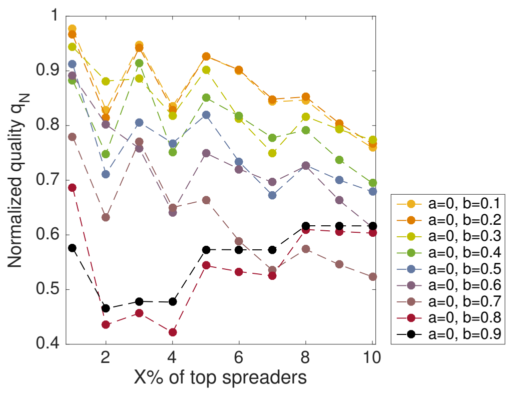
(c) EmailEnron

Figure 5.2: **Distribution of nodes in the top k-core subgraphs in the original fomat of the real datasets tested.** The data reported for the following datasets: **(a)** NetHept, **(b)**WikiVote and **(c)** EmailEnron. The coreness values of X%=1%, 5% and 10% of top spreaders selected are shown.

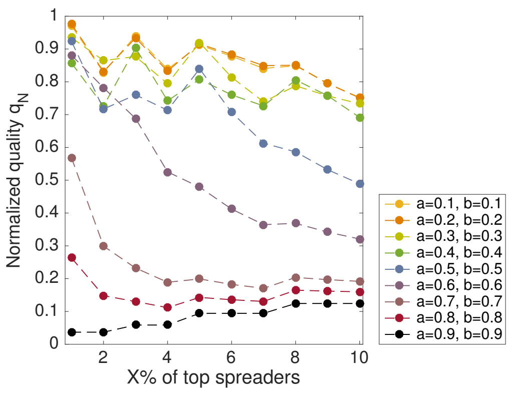
ting I, a=0, b=0.5). Whereas when both edges are added and deleted, the normalized quality is greater than 70% for the first four anonymized versions (i.e., until there are a=0.4 and b=0.4 additions and deletions respectively). In the case of the WikiVote dataset, the quality results are more promising even for large perturbations. We observe that even for 90% of edge deletions (*Setting* I, a=0, b=0.9), q_N is approximately 78% when 10% of top spreaders are to be selected. In general for both *Settings* I and II the normalized quality is greater than approximately 70% and 60% respectively. For the Email-Enron dataset the results are still convincing, the q_N values are greater than 70% up until there are 80% of edge deletions (*Setting* I, a=0, b=0.8) and 70% of edge additions and deletions (*Setting* II, a=0.7, b=0.7). It is important to note that Narayanan et al. have shown in [128] that graph de-anonymiza-tion attacks are ineffective when $b > 0.75$.

In consequence, experimental results show that even for the case when the original graph is subject to great perturbations, the normalized quality is sufficiently high. Thus, we can indeed run our algorithm on a modified version of the graph $\tilde{G}$ for high AS values thus preventing reconstruction of the original graph from possible attackers while preserving the desired information quality.

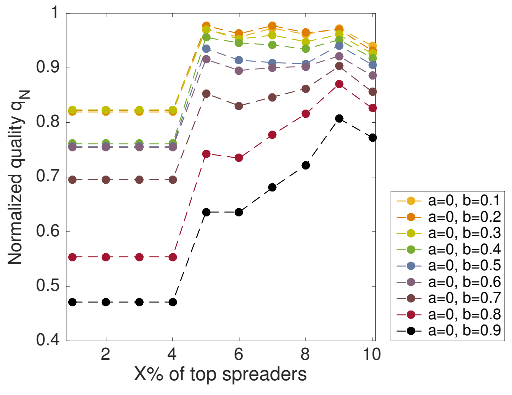
What is interesting to mention, is that the q_N values presented for a range of perturbations below 50% or 70% for NetHept and WikiVote respectively show a similar behavior. Under such perturbations the graph has not yet lost the required



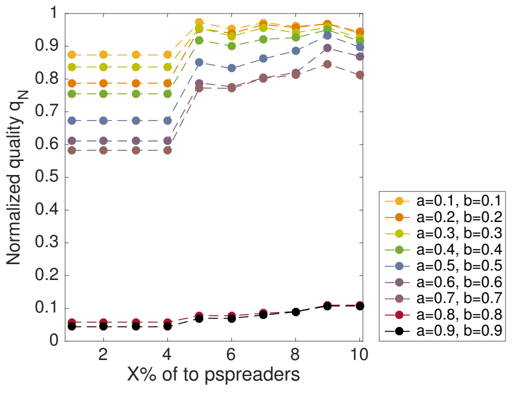
(a) NetHept: Setting I



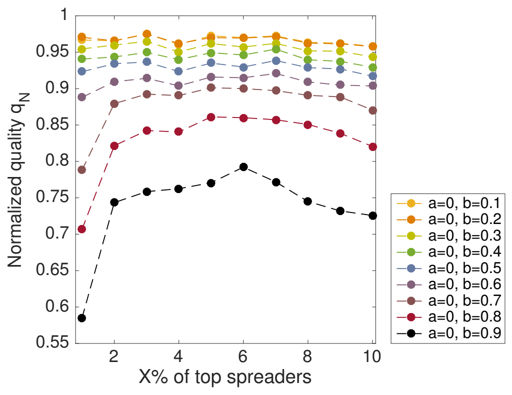
(b) NetHept: Setting II



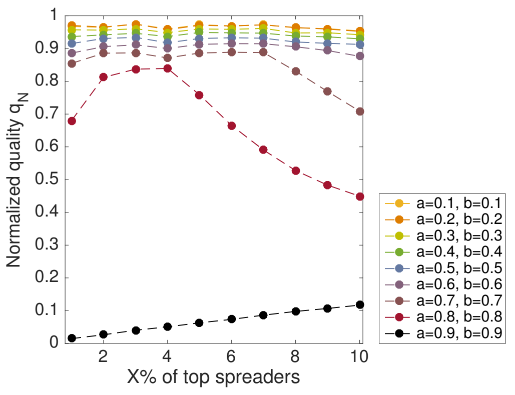
(c) WikiVote: Setting I



(d) WikiVote: Setting II



(c) EmailEnron: Setting I



(d) EmailEnron: Setting II

Figure 5.3: **Normalized Quality q_N of different percentages of top spreaders selected from anonymized versions of real datasets.**

information quality for nodes' spreading capabilities. Observe for example the behavior of the q_N values for both Settings for the WikiVote dataset up until there are 70% of edge modifications. Similarly for the NetHept dataset up until there are 50% of edge modifications. This behavior may be justified by the specific coreness distribution of the nodes in the original graph with which the comparison occurs in order to calculate the normalized quality values. Figure 5.2 depicts the number of nodes distributed in top coreness groups for the original forms of the datasets examined. The coreness groups depicted include the top 10% of nodes that are used to provide the quality results.

Finally, let us note that it would be possible to conduct experiments by controlling even more how edges are added and deleted, in order to increase the difficulty of de-anonymization techniques (e.g. by increasing the probability of removing edges for high degree nodes). We have not studied the impact of such optimizations since we believe that current results are already very convincing and show that our anonymization approach is feasible and provides good quality while achieving good security and privacy.

To conclude on the privacy aspect, one could reflect on the privacy risk of publishing the coreness value of a node. Coreness value is not a direct identifier, such as degree can be, where some nodes (or small groups of nodes) have unexpectedly high and/or rare degree values, and are consequently easy to deanonymize. On the contrary, coreness defines an equivalence class. As shown in column ICS of Tables 5.5 and 5.4, the cardinality of the smallest equivalence groups is usually hundreds of nodes, thus coreness benefits from an intrinsic form of k-anonymization.

5.7 CONCLUSIONS AND REMARKS

In this Chapter we have studied the privacy point of view of sharing metrics that are good indicators concerning the spreading capabilities of a node in a network. We have specifically focused on the k-core centrality of a node which has been proved to be an efficient metric to locate the nodes that succeed in disseminating information to a large part of the population.

We have designed an algorithm that computes in an efficient, correct, secure and privacy-preserving way this k-core metric and adopted a decentralization approach where the social network is considered as a Peer-to-peer (P2P) system. While a distributed algorithm that computes once and for all the nodes' coreness is already proposed, networks that evolve over time are not taken into account. Our main contribution is an incremental algorithm that succesfully computes the up-to-date k-core values that limits the number of messages and computations needed when a modification occurs to the network.

By performing a complexity analysis and experimenting on real graphs, we show that our algorithm is less costly than the existing baseline algorithm in terms of numbers of messages that will be sent between peers, number of local computations and temporal complexity.

We finally perform a security and privacy analysis of our system. After describing the possible attacks that may occur in a social network, we discuss the desired privacy and information quality that needs to be achieved in our scenario. After performing experiments on real datasets, we show that the information quality achieved is sufficiently high even when the original graph is subject to great perturbations that serve to provide anonymized versions of the graph for privacy purposes.

As future work we plan to extend our algorithm to support concurrent changes in the network. We also intend to propose a distributed algorithm that succeds in correctly computing other influence indicators (e.g., the K-truss centrality) for dynamic networks.

6 CONCLUDING REMARKS

Networks are ubiquitous and have introduced numerous challenging problems to the research community. This dissertation has focused on social networks and specially on social influence. We specifically use graph mining techniques to study influence propagation and influence maximization in social networks. We particularly:

- Develop tools that can efficiently rank the users based on their influential capabilities.
- Design algorithms that can locate a privileged group of nodes that – by acting all together – can maximize the spread of influence in a network at the end of a diffusion phenomenon.
- Develop models that can calculate metrics which measure the influence of an individual in a network in a secure and private way.

In the next Section, we provide an overview of the main contributions of the thesis and discuss future research directions.

6.1 SUMMARY OF CONTRIBUTIONS AND FUTURE WORK

IDENTIFICATION OF INDIVIDUAL INFLUENTIAL SPREADERS In Chapter 3 we focused on identifying individual influential spreaders in social networks. Even if degree centrality may seem an effective metric in locating such privileged entities, it appears to present some drawbacks. A node may have a great number of neighbors but if it is located in the periphery of the network its influential capabilities are reduced. It has been shown that those nodes that are centrally placed in the network are those that can efficiently spread information to the greatest possible number of individuals. We specifically proposed to use the K-truss decomposition to locate such individuals and we showed that indeed the specific nodes can influence a greater part of the network during the first steps of the spreading process but also the total fraction of influenced nodes at the end is higher.

Moreover, we explored the centralities of the entities that are involved in a spreading process which is triggered by different groups of influential spreaders of a network. While using models borrowed from the field of epidemics to simulate the process, we observed specific patterns during the spreading phenomenon. By comparing the simulated diffusion process with real influence, we observe that the aforementioned models cannot reproduce the real diffusion in terms of the evolution of the centralities of the infected nodes.

It would be interesting to experiment in the future with dynamic networks where node and/or edge modifications may occur. The question seems to be whether the K-truss decomposition is a robust enough metric to provide such entities that will still retain their spreading capabilities after changes in the network occur. Finally it would be interesting to check whether the already proposed metrics for locating privileged spreaders work equally well when different models are used to simulate the process. We would furthermore like to examine what is the behavior of the centralities of the entities when different spreading models are used and whether those can reproduce real diffusion.

identification of a group of influential spreaders

In Chapter 4 we studied the problem of locating a group of nodes in a social network that by acting together can maximize information diffusion. Nodes that are discovered using methods mentioned in Chapter 3 cannot be directly used in order to discover the set of nodes in question. That is justified by the fact that the influence of one can overlap with the influence of another top spreader. The problem of *Influence Maximization(IM)* – as it is usually called – constitutes an NP-hard problem. A simple greedy algorithm has been proved to provide good approximation guarantees. Nevertheless, there are obviously serious scalability concerns – the greedy algorithm cannot provide results as soon as needed for large-scale networks.

We proposed a Matrix Influence (MATI) algorithm, an efficient influence maximization algorithm designed for both the Linear Threshold (LT) and Independent Cascade (IC) diffusion models. MATI takes advantage of the possible paths that are created in each node's neighborhood and succeeds in locating the users that can maximize the influence in a social network while also being scalable for large datasets.

It would be interesting to experiment with the centralities used to identify individual spreaders in order to see whether they can describe the group of nodes that are discovered after applying influence maximization algorithms. If the respective group of nodes is well defined by the aforementioned centralities, it would be interesting to use them as heuristics to speed up the IM algorithms. It has been shown that the already existing models that simulate information diffusion cannot efficiently reproduce a real spreading process in networks. It would be interesting to design new models that can represent better a spreading phenomenon by incorporating memory of past events (succesful or unsuccesful efforts of entities to influence one another) or by taking into account also the fact that in real life influence is not always positive but might be negative as well.

SECURE AND PRIVATE COMPUTATION OF INFLUENTIAL METRICS In Chapter 5 we focused on the secure and private computation of metrics that reveal influential entities in social networks. Nevertheless such metrics require knowledge of the social network. Such practices raise serious concerns associated with the publishing of such sensitive information. We have adopted a decentralization approach to favor privacy, we specifically proposed a P2P algorithm that can efficiently calculate the k-core centrality of each node.

The k-core centrality has been proven to be a metric that succesfully ranks the nodes in a network based on their spreading capabilities. A distributed algorithm that calculates said metric already exists, though networks that evolve over time are not taken into account. Our main contribution is an incremental algorithm that succesfully computes the up-to-date k-core values that limits the number of messages and computations needed when a modification occurs to the network.

We performed a complexity analysis and made experiments on real graphs to show that our algorithm is less computationally expensive in terms of numbers of messages that are sent between peers, number of node computations and temporal complexity. Additionally, a security and privacy analysis has been performed along with experiments on real datasets. There is a need of good information quality in anonymized versions of the datasets in order to prevent attackers from reconstructing the original dataset. We showed that great perturbations do not result in severe loss of the quality of the spreaders that can be provided by the k-core decomposition thus our algorithm can indeed securely calculate such a measure.

Our algorithm treats each update individually but we are currently working on extending it in order to support concurrent changes in the network. It would be interesting to design a distributed algorithm that calculates other influence indicators such as the K-truss centrality which as presented in Chapter 3 has been proven to filter out the best spreaders of the k-core structure.

6.2 EPILOGUE

Social networks have presented numerous demanding but at the same time interesting problems to the research community. Throughout this dissertation we have presented our understanding of the area of influence maximization and reported findings that will hopefully help its comprehension and progress. Even though a lot of work has been done in the field, there are still unanswered questions and challenging problems that will further enlighten our knowledge about networks but also social behavior.

BIBLIOGRAPHY

[1] Hidayet Aksu, Mustafa Canim, Yuan-Chi Chang, Ibrahim Korpeoglu, and Özgür Ulusoy. "Distributed k-Core View Materialization and Maintenance for Large Dynamic Graphs." In: *IEEE Transactions on Knowledge and Data Engineering* 26.10 (2014), pp. 2439–2452.

[2] R. Albert, H. Jeong, and A.L. Barabási. "Error and attack tolerance of complex networks." In: *Nature* 406.6794 (2000).

[3] Réka Albert and Albert-László Barabási. "Statistical mechanics of complex networks." In: *Rev. Mod. Phys.* 74 (1 2002), pp. 47–97.

[4] Tristan Allard, Nicolas Anciaux, Luc Bouganim, Yanli Guo, Lionel Le Folgoc, Benjamin Nguyen, Philippe Pucheral, Indrajit Ray, Indrakshi Ray, and Shaoyi Yin. "Secure personal data servers: a vision paper." In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 25–35.

[5] Luıs A Nunes Amaral, Antonio Scala, Marc Barthelemy, and H Eugene Stanley. "Classes of small-world networks." In: *Proceedings of the national academy of sciences* 97.21 (2000), pp. 11149–11152.

[6] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. "Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography." In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 181–190.

[7] Joonhyun Bae and Sangwook Kim. "Identifying and ranking influential spreaders in complex networks by neighborhood coreness." In: *Physica A: Statistical Mechanics and its Applications* 395 (2014), pp. 549–559.

[8] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. "Everyone's an Influencer: Quantifying Influence on Twitter." In: *WSDM*. 2011.

[9] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. "Formal models for expert finding in enterprise corpora." In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2006, pp. 43–50.

[10] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. “Topic-Aware Social Influence Propagation Models.” In: *ICDM*. 2012.

[11] John Arundel Barnes. “Class and committees in a Norwegian island parish.” In: *Human relations* 7.1 (1954), pp. 39–58.

[12] Alain Barrat, Marc Barthlemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. 1st. New York, NY, USA: Cambridge University Press, 2008.

[13] Pavlos Basaras, Dimitrios Katsaros, and Leandros Tassiulas. “Detecting Influential Spreaders in Complex, Dynamic Networks.” In: *Computer* 46.4 (2013), pp. 24–29.

[14] Frank Bass. “A New Product Growth Model for Consumer Durable.” In: *Management Sciences* (1969), pp. 215–27.

[15] Vladimir Batagelj and Matjaz Zaversnik. “An O (m) algorithm for cores decomposition of networks.” In: *arXiv preprint cs/0310049* (2003).

[16] Vladimir Batagelj and Matjaz Zaversnik. “An $\mathcal{O}(m)$ Algorithm for Cores Decomposition of Networks.” In: *CoRR* (2003).

[17] Alex Bavelas. “A mathematical model for group structures.” In: *Human organization* 7.3 (1948), pp. 16–30.

[18] Peter S Bearman, James Moody, and Katherine Stovel. “Chains of affection: The structure of adolescent romantic and sexual networks.” In: *American journal of sociology* 110.1 (2004), pp. 44–91.

[19] Norman Biggs. *Algebraic graph theory*. Cambridge university press, 1993.

[20] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. “Complex networks: Structure and dynamics.” In: *Physics reports* 424.4 (2006), pp. 175–308.

[21] Phillip Bonacich. “Power and centrality: A family of measures.” In: *American journal of sociology* 92.5 (1987), pp. 1170–1182.

[22] Phillip Bonacich. “Some unique properties of eigenvector centrality.” In: *Social networks* 29.4 (2007), pp. 555–564.

[23] Francesco Bonchi and Elena Ferrari. *Privacy-aware knowledge discovery: novel applications and new techniques*. CRC Press, 2010.

[24] Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. "Core decomposition of uncertain graphs." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 1316–1325.

[25] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*. Vol. 290. Citeseer, 1976.

[26] Stephen P Borgatti. "Centrality and network flow." In: *Social networks* 27.1 (2005), pp. 55–71.

[27] Stephen P Borgatti and Martin G Everett. "A graph-theoretic perspective on centrality." In: *Social networks* 28.4 (2006), pp. 466–484.

[28] Javier Borge-Holthoefer, Alejandro Rivero, and Yamir Moreno. "Locating privileged spreaders on an online social network." In: *Phys. Rev. E* 85 (6 2012), p. 066123.

[29] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. "Maximizing Social Influence in Nearly Optimal Time." In: *SODA*. 2014.

[30] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine." In: *Computer networks and ISDN systems* 30.1 (1998), pp. 107–117.

[31] Sergey Brin and Lawrence Page. "Reprint of: The anatomy of a large-scale hypertextual web search engine." In: *Computer networks* 56.18 (2012), pp. 3825–3833.

[32] Jacqueline Johnson Brown and Peter H Reingen. "Social ties and word-of-mouth referral behavior." In: *Journal of Consumer research* 14.3 (1987), pp. 350–362.

[33] Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. "A Model of Internet Topology using k-shell Decomposition." In: *PNAS* 104.27 (2007), pp. 11150–11154.

[34] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. "k-Degree anonymity and edge selection: improving data utility in large networks." In: *Knowledge and Information Systems* 50.2 (2017), pp. 447–474.

[35] Damon Centola. "The spread of behavior in an online social network experiment." In: *science* 329.5996 (2010), pp. 1194–1197.

[36] Deepayan Chakrabarti and Christos Faloutsos. "Graph mining: laws, tools, and case studies." In: *Synthesis Lectures on Data Mining and Knowledge Discovery* 7.1 (2012), pp. 1–207.

[37] Duan-Bing Chen, Hui Gao, Linyuan Lü, and Tao Zhou. "Identifying Influential Nodes in Large-Scale Directed Networks: The Role of Clustering." In: *PLoS ONE* 8.10 (2013), e77455.

[38] Duan-Bing Chen, Rui Xiao, An Zeng, and Yi-Cheng Zhang. "Path diversity improves the identification of influential spreaders." In: *EPL (Europhysics Letters)* 104.6 (2013), p. 68006.

[39] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. "Identifying influential nodes in complex networks." In: *Physica A: Statistical Mechanics and its Applications* 391.4 (2012), pp. 1777–1787.

[40] Wei Chen, Chi Wang, and Yajun Wang. "Scalable Influence Maximization for Prevalent Viral Marketing in Large-scale Social Networks." In: *KDD*. 2010.

[41] Wei Chen, Yifei Yuan, and Li Zhang. "Scalable Influence Maximization in Social Networks Under the Linear Threshold Model." In: *ICDM*. 2010.

[42] James Cheng, Yiping Ke, Shumo Chu, and M Tamer Özsu. "Efficient core decomposition in massive networks." In: *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE. 2011, pp. 51–62.

[43] Nicholas A Christakis and James H Fowler. "The spread of obesity in a large social network over 32 years." In: *n engl j med* 2007.357 (2007), pp. 370–379.

[44] Nicholas A Christakis and James H Fowler. "The collective dynamics of smoking in a large social network." In: *New England journal of medicine* 358.21 (2008), pp. 2249–2258.

[45] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. "Power-Law Distributions in Empirical Data." In: *SIAM Review* 51.4 (2009), pp. 661–703.

[46] Edith Cohen. "Size-Estimation Framework with Applications to Transitive Closure and Reachability." In: *Journal of Computer and System Sciences* 55.3 (1997).

[47] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. "Sketch-based Influence Maximization and Computation: Scaling Up with Guarantees." In: *CIKM '14*. 2014.

[48] Jonathan Cohen. "Trusses: Cohesive subgraphs for social network analysis." In: *National Security Agency Technical Report* 16 (2008).

[49] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. "Efficient immunization strategies for computer networks and populations." In: *Physical review letters* 91.24 (2003), p. 247901.

[50] Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. "Breakdown of the Internet under Intentional Attack." In: *Phys. Rev. Lett.* 86 (16 2001), pp. 3682–3685.

[51] Gerard Cornuejols, Marshall L Fisher, and George L Nemhauser. "Exceptional paper—Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms." In: *Management science* 23.8 (1977), pp. 789–810.

[52] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. "Safebook: A privacy-preserving online social network leveraging on real-life trust." In: *IEEE Communications Magazine* 47.12 (2009).

[53] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. "The anatomy of a scientific rumor." In: *Scientific reports* 3 (2013), p. 2980.

[54] Klaus Dietz. "Epidemics and rumours: A survey." In: *Journal of the Royal Statistical Society. Series A (General)* (1967), pp. 505–528.

[55] Ying Ding, Erjia Yan, Arthur Frazho, and James Caverlee. "PageRank for ranking authors in co-citation networks." In: *Journal of the Association for Information Science and Technology* 60.11 (2009), pp. 2229–2243.

[56] Peter Sheridan Dodds and Duncan J Watts. "Universal behavior in a generalized model of contagion." In: *Physical review letters* 92.21 (2004), p. 218701.

[57] Pedro Domingos and Matt Richardson. "Mining the Network Value of Customers." In: *KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001, pp. 57–66.

[58] Pedro Domingos and Matt Richardson. "Mining the network value of customers." In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2001, pp. 57–66.

[59] Leo Egghe and Ronald Rousseau. *Introduction to informetrics: Quantitative methods in library, documentation and information science*. Elsevier Science Publishers, 1990.

[60] Marius Eidsaa and Eivind Almaas. "S-core network decomposition: A generalization of k-core analysis to weighted networks." In: *Physical Review E* 88.6 (2013), p. 062819.

[61] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. "On Power-law Relationships of the Internet Topology." In: *SIGCOMM '99: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 1999, pp. 251–262.

[62] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. "On power-law relationships of the internet topology." In: *ACM SIGCOMM computer communication review*. Vol. 29. 4. ACM. 1999, pp. 251–262.

[63] Uriel Feige. "A threshold of ln n for approximating set cover." In: *Journal of the ACM (JACM)* 45.4 (1998), pp. 634–652.

[64] Linton C Freeman. "A set of measures of centrality based on betweenness." In: *Sociometry* (1977), pp. 35–41.

[65] Linton C Freeman. "Centrality in social networks conceptual clarification." In: *Social networks* 1.3 (1978), pp. 215–239.

[66] Joseph Galaskiewicz. *Social organization of an urban grants economy: A study of business philanthropy and nonprofit organizations*. Elsevier, 2016.

[67] Joseph Galaskiewicz and Peter V Marsden. "Interorganizational resource networks: Formal patterns of overlap." In: *Social science research* 7.2 (1978), pp. 89–107.

[68] Antonios Garas, Frank Schweitzer, and Shlomo Havlin. "A k-shell decomposition method for weighted networks." In: *New Journal of Physics* 14.8 (2012), p. 083030.

[69] Craig Gentry et al. "Fully homomorphic encryption using ideal lattices." In: *STOC*. Vol. 9. 2009. 2009, pp. 169–178.

[70] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. "D-cores: Measuring collaboration of directed graphs based on degeneracy." In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE. 2011, pp. 201–210.

[71] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. "Evaluating cooperation in communities with the k-core structure." In: *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE. 2011, pp. 87–93.

[72] Jacob Goldenberg, Barak Libai, and Eitan Muller. "Talk of the network: A complex systems look at the underlying process of word-of-mouth." In: *Marketing letters* 12.3 (2001), pp. 211–223.

[73] Jacob Goldenberg, Barak Libai, and Eitan Muller. "Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata." In: *Academy of Marketing Science Review* 2001 (2001), p. 1.

[74] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption & how to play mental poker keeping secret all partial information." In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM. 1982, pp. 365–377.

[75] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. "Learning influence probabilities in social networks." In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pp. 241–250.

[76] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. "A data-based approach to social influence maximization." In: *Proceedings of the VLDB Endowment* 5.1 (2011), pp. 73–84.

[77] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. "SIMPATH: An Efficient Algorithm for Influence Maximization Under the Linear Threshold Model." In: *ICDM*. 2011.

[78] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. "CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks." In: *WWW*. 2011.

[79] Nabil Guelzim, Samuele Bottani, Paul Bourgine, and François Képes. "Topological and causal structure of the yeast transcriptional regulatory network." In: *Nature genetics* 31.1 (2002), p. 60.

[80] Michelle Gumbrecht. "Blogs as "protected space"." In: *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*. Vol. 2004. 2004.

[81] Frank Hoppenstaedt. *Mathematical theories of populations: demographics, genetics and epidemics*. SIAM, 1975.

[82] Harold Hotelling. "Simplified calculation of principal components." In: *Psychometrika* 1.1 (1936), pp. 27–35.

[83] Bonan Hou, Yiping Yao, and Dongsheng Liao. "Identifying all-around nodes for spreading dynamics in complex networks." In: *Physica A: Statistical Mechanics and its Applications* 391.15 (2012), pp. 4012–4017.

[84] William H Hsu, Andrew L King, Martin SR Paradesi, Tejaswi Pydimarri, and Tim Weninger. "Collaborative and Structural Recommendation of Friends using Weblog-based Social Network Analysis." In: *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. Vol. 6. 2006, pp. 55–60.

[85] Bernardo A Huberman. *The laws of the Web: Patterns in the ecology of information*. mit Press, 2001.

[86] O Hugo and E Garnsey. "Hotmail: Delivering E-mail to the World." In: *European Case Clearing House (ECCH)* (2002).

[87] Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. "Locating data in (small-world?) peer-to-peer scientific collaborations." In: *Peer-to-Peer Systems* (2002), pp. 232–241.

[88] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. "A comprehensive two-hybrid analysis to explore the yeast protein interactome." In: *Proceedings of the National Academy of Sciences* 98.8 (2001), pp. 4569–4574.

[89] Swami Iyer, Timothy Killingback, Bala Sundaram, and Zhen Wang. "Attack robustness and centrality of complex networks." In: *PloS one* 8.4 (2013), e59613.

[90] James Holland Jones and Mark S Handcock. "An assessment of preferential attachment as a mechanism for human sexual network formation." In: *Proceedings of the Royal Society of London B: Biological Sciences* 270.1520 (2003), pp. 1123–1128.

[91] VK Kalapala, V Sanwalani, and C Moore. "The structure of the United States road network." In: *Preprint, University of New Mexico* (2003).

[92] Matt J Keeling and Pejman Rohani. *Modeling infectious diseases in humans and animals*. Princeton University Press, 2008.

[93] David Kempe, Jon M. Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network." In: *KDD*. 2003.

[94] David Kempe, Jon M Kleinberg, and Éva Tardos. "Influential Nodes in a Diffusion Model for Social Networks." In: *ICALP*. Vol. 5. Springer. 2005, pp. 1127–1138.

[95] W. O. Kermack and Ag McKendrick. "A Contribution to the Mathematical Theory of Epidemics." In: *Proceedings of the Royal Society of London* 115.772 (1927), pp. 700–721.

[96] William O Kermack and Anderson G McKendrick. "A contribution to the mathematical theory of epidemics." In: *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*. Vol. 115. 772. The Royal Society. 1927, pp. 700–721.

[97] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. Stanley, and H. Makse. "Identification of Influential Spreaders in Complex Networks." In: *Nature Physics* 6.11 (2010), pp. 888–893.

[98] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. "Identification of influential spreaders in complex networks." In: *arXiv preprint arXiv:1001.5285* (2010).

[99] Konstantin Klemm, M Ángeles Serrano, Víctor M Eguíluz, and Maxi San Miguel. "A measure of individual role in collective dynamics." In: *Scientific reports* 2 (2012).

[100] Bryan Klimt and Yiming Yang. "The enron corpus: A new dataset for email classification research." In: *Machine learning: ECML 2004* (2004), pp. 217–226.

[101] Paul L Krapivsky, Sidney Redner, and D Volovik. "Reinforcement-driven spread of innovations and fads." In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.12 (2011), P12003.

[102] Matthieu Latapy. "Main-memory triangle computations for very large (sparse (power-law)) graphs." In: *Theoretical Computer Science* 407.1-3 (2008), pp. 458–473.

[103] Damien Leprovost, Lylia Abrouk, Nadine Cullot, and David Gross-Amblard. "Temporal semantic centrality for the analysis of communication networks." In: *Web Engineering* (2012), pp. 177–184.

[104] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. "The dynamics of viral marketing." In: *ACM Transactions on the Web (TWEB)* 1.1 (2007), p. 5.

[105] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. "Predicting positive and negative links in online social networks." In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 641–650.

[106] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 2.

[107] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. "Cost-effective Outbreak Detection in Networks." In: *KDD*. 2007.

[108] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters." In: *Internet Mathematics* 6.1 (2009), pp. 29–123.

[109] Qian Li, Tao Zhou, Linyuan Lü, and Duanbing Chen. "Identifying influential spreaders by weighted LeaderRank." In: *Physica A: Statistical Mechanics and its Applications* 404 (2014), pp. 47–55.

[110] Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao. "Efficient core maintenance in large dynamic graphs." In: *IEEE Transactions on Knowledge and Data Engineering* 26.10 (2014), pp. 2453–2465.

[111] David Liben-Nowell and Jon Kleinberg. "The link-prediction problem for social networks." In: *journal of the Association for Information Science and Technology* 58.7 (2007), pp. 1019–1031.

[112] Jian-Guo Liu, Jian-Hong Lin, Qiang Guo, and Tao Zhou. "Locating influential nodes via dynamics-sensitive centrality." In: *Scientific reports* 6 (2016).

[113] Linyuan Lü, Duan-Bing Chen, and Tao Zhou. "The small world yields the most effective information spreading." In: *New Journal of Physics* 13.12 (2011), p. 123005.

[114] Linyuan Lü, Yi-Cheng Zhang, Chi Ho Yeung, and Tao Zhou. "Leaders in social networks, the delicious case." In: *PloS one* 6.6 (2011), e21202.

[115] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. "Recommender systems." In: *Physics Reports* 519.1 (2012), pp. 1–49.

[116] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. "Vital nodes identification in complex networks." In: *Physics Reports* 650 (2016), pp. 1–63.

[117] Tomasz Łuczak. "Size and connectivity of the k-core of a random graph." In: *Discrete Mathematics* 91.1 (1991), pp. 61–68.

[118] Vijay Mahajan, Eitan Muller, and Frank M Bass. "New product diffusion models in marketing: A review and directions for research." In: *Diffusion of technologies and social behavior*. Springer, 1991, pp. 125–177.

[119] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. "Pregel: a system for large-scale graph processing." In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, pp. 135–146.

[120] Fragkiskos D Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. "Locating influential nodes in complex networks." In: *Scientific reports* 6 (2016).

[121] David W Matula and Leland L Beck. "Smallest-last ordering and clustering and graph coloring algorithms." In: *Journal of the ACM (JACM)* 30.3 (1983), pp. 417–427.

[122] Ericka Menchen. "Blogger motivations: Power, pull, and positive feedback." In: *6th International and Interdisciplinary Association of Internet Researchers* (2005).

[123] Daniele Miorandi and Francesco De Pellegrini. "K-shell decomposition for dynamic complex networks." In: *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*. IEEE. 2010, pp. 488–496.

[124] Michael Molloy. "Cores in random hypergraphs and Boolean formulas." In: *Random Structures & Algorithms* 27.1 (2005), pp. 124–135.

[125] Alberto Montresor, Francesco De Pellegrini, and Daniele Miorandi. "Distributed k-core decomposition." In: *IEEE Transactions on parallel and distributed systems* 24.2 (2013), pp. 288–300.

[126] Jacob L Moreno. *Who shall survive*. Vol. 58. JSTOR, 1934.

[127] Yamir Moreno, Maziar Nekovee, and Amalio F Pacheco. "Dynamics of rumor spreading in complex networks." In: *Physical Review E* 69.6 (2004), p. 066130.

[128] Arvind Narayanan and Vitaly Shmatikov. "De-anonymizing social networks." In: *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE. 2009, pp. 173–187.

[129] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I." In: *Mathematical Programming* 14.1 (1978), pp. 265–294.

[130] M. E. J. Newman. "Spread of epidemic disease on networks." In: *Physical Review E* 66.1 (2002), p. 016128.

[131] M.E.J. Newman. "The Structure and Function of Complex Networks." In: *SIAM Review* 45.2 (2003), pp. 167–256.

[132] Mark EJ Newman. "The structure and function of complex networks." In: *SIAM review* 45.2 (2003), pp. 167–256.

[133] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[134] Romualdo Pastor-Satorras and Alessandro Vespignani. "Epidemic Spreading in Scale-Free Networks." In: *Phys. Rev. Lett.* 86 (14 2001), pp. 3200–3203.

[135] Romualdo Pastor-Satorras and Alessandro Vespignani. "Immunization of complex networks." In: *Physical Review E* 65.3 (2002), p. 036104.

[136] Sen Pei and Hernán A Makse. "Spreading dynamics in complex networks." In: *Journal of Statistical Mechanics: Theory and Experiment* 2013.12 (2013), P12002.

[137] Sen Pei, Lev Muchnik, José S Andrade Jr, Zhiming Zheng, and Hernán A Makse. "Searching for superspreaders of information in real-world social media." In: *Scientific reports* 4 (2014), p. 5547.

[138] Sen Pei, Lev Muchnik, Shaoting Tang, Zhiming Zheng, and Hernán A Makse. "Exploring the complex pattern of information spreading in online blog communities." In: *PloS one* 10.5 (2015), e0126894.

[139] James Edward Van der Plank. *Plant diseases: epidemics and control*. Elsevier, 2013.

[140] Filippo Radicchi, Santo Fortunato, Benjamin Markines, and Alessandro Vespignani. "Diffusion of scientific credits and the ranking of scientists." In: *Physical Review E* 80.5 (2009), p. 056103.

[141] John Rice. *Mathematical statistics and data analysis*. Nelson Education, 2006.

[142] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. "Trust management for the semantic web." In: *The Semantic Web-ISWC 2003* (2003), pp. 351–368.

[143] Maria-Evgenia G Rossi, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "Spread it good, spread it fast: Identification of influential nodes in social networks." In: *Proceedings of the 24th International Conference on World Wide Web*. ACM. 2015, pp. 101–102.

[144] Maria-Evgenia G Rossi and Michalis Vazirgiannis. "Exploring Network Centralities in Spreading Processes." In: *International Symposium on Web AlGorithms (iSWAG)*. 2016.

[145] Maria-Evgenia G Rossi, Bowen Shi, Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Christos Giatsidis, and Michalis Vazirgiannis. "MATI: An Efficient Algorithm for Influence Maximization in Social Networks." In: *Manuscript*. 2017.

[146] Maria-Evgenia G Rossi, Cédric Eichler, Pascal Berthomé, and Benjamin Nguyen. "Private, Secure and Distributed Computation of k-cores." In: *Manuscript, presented in APVP*. 2017.

[147] Gert Sabidussi. "The centrality index of a graph." In: *Psychometrika* 31.4 (1966), pp. 581–603.

[148] Ahmet Erdem Saríyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V Çatalyürek. "Streaming algorithms for k-core decomposition." In: *Proceedings of the VLDB Endowment* 6.6 (2013), pp. 433–444.

[149] Thomas Schank. "Algorithmic aspects of triangle-based network analysis." In: (2007).

[150] Stephen B. Seidman. "Network Structure and Minimum Degree." In: *Social Networks* 5 (1983), pp. 269–287.

[151] Stephen B Seidman. "Network structure and minimum degree." In: *Social networks* 5.3 (1983), pp. 269–287.

[152] Marvin E Shaw. "Group structure and the behavior of individuals in small groups." In: *The Journal of psychology* 38.1 (1954), pp. 139–149.

[153] Alfonso Shimbel. "Structural parameters of communication networks." In: *The bulletin of mathematical biophysics* 15.4 (1953), pp. 501–507.

[154] Konstantinos Skianis, Maria-Evgenia G Rossi, Fragkiskos D Malliaros, and Michalis Vazirgiannis. "SpreadViz: Analytics and Visualization of Spreading Processes in Social Networks." In: *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. IEEE. 2016, pp. 1324–1327.

[155] Xiaodan Song, Belle L. Tseng, Ching-Yung Lin, and Ming-Ting Sun. "Personalized Recommendation Driven by Information Flow." In: *SIGIR Conference on Research and Development in Information Retrieval*. 2006.

[156] Jorg Stelling, Steffen Klamt, Katja Bettenbrock, Stefan Schuster, and Ernst Dieter Gilles. "Metabolic network structure determines key aspects of functionality and regulation." In: *Nature* 420.6912 (2002), p. 190.

[157] Youze Tang, Xiaokui Xiao, and Yanchen Shi. "Influence Maximization: Near-optimal Time Complexity Meets Practical Efficiency." In: *SIGMOD*. 2014.

[158] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. "Link prediction in relational data." In: *Advances in neural information processing systems*. 2004, pp. 659–666.

[159] Michael Trusov, Randolph E. Bucklin, and Koen Pauwels. "Effects of Word-of-Mouth Versus Traditional Marketing: Findings from an Internet Social Networking Site." In: *Journal of Marketing* 73.5 (2009), pp. 90–102.

[160] Jia Wang and James Cheng. "Truss decomposition in massive networks." In: *Proc. VLDB Endow.* 5.9 (2012), pp. 812–823.

[161] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press, 1994.

[162] Duncan J Watts and Steven H Strogatz. "Collective dynamics of'small-world'networks." In: *nature* 393.6684 (1998), p. 440.

[163] Dong Wen, Lu Qin, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. "I/o efficient core graph decomposition at web scale." In: *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE. 2016, pp. 133–144.

[164] Xintao Wu, Xiaowei Ying, Kun Liu, and Lei Chen. "A survey of privacy-preservation of graphs and social networks." In: *Managing and mining graph data* (2010), pp. 421–453.

[165] An Zeng and Cheng-Jun Zhang. "Ranking spreaders by decomposing complex networks." In: *Physics Letters A* 377.14 (2013), pp. 1031–1035.

[166] Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. "Social media analytics and intelligence." In: *IEEE Intelligent Systems* 25.6 (2010), pp. 13–16.

[167] Xiaohang Zhang, Ji Zhu, Qi Wang, and Han Zhao. "Identifying influential nodes in complex networks with community structure." In: *Knowledge-Based Systems* 42 (2013), pp. 74–84.

[168] Yang Zhang and Srinivasan Parthasarathy. "Extracting Analyzing and Visualizing Triangle K-Core Motifs within Networks." In: *ICDE '12: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. 2012, pp. 1049–1060.

[169] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. "A fast order-based approach for core maintenance." In: *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE. 2017, pp. 337–348.

[170] Yan-Bo Zhou, Linyuan Lü, and Menghui Li. "Quantifying the influence of scientists and their publications: distinguishing between prestige and popularity." In: *New Journal of Physics* 14.3 (2012), p. 033033.

COLOPHON

This document was typeset in LaTeX using the typographical look-and-feel `classicthesis`. The graphics in this dissertation are generated using the `Matlab` numerical computing environment, the `R` language, the `Ipe` extensible drawing editor and `pgf/tikz`. The bibliography is typeset using `biblatex`.