INVESTIGATING CORE PERIPHERY STRUCTURE FOR CENTRALITY RESILIENCE, PREDICTION AND REPRESENTATION LEARNING

Soumya Sarkar

INVESTIGATING CORE PERIPHERY STRUCTURE FOR CENTRALITY RESILIENCE, PREDICTION AND REPRESENTATION LEARNING

Thesis submitted to the
Indian Institute of Technology, Kharagpur
For award of the degree

of

Doctor of Philosophy

by

Soumya Sarkar

Under the supervision of

Prof. Animesh Mukherjee

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

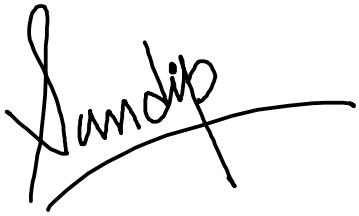
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

September 2020

APPROVAL OF THE VIVA-VOCE BOARD

Date: 17/ 09/ 20

Certified that the thesis entitled **"Investigating core periphery structure for centrality resilience, prediction and representation learning"** submitted by Soumya Sarkar to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.



(Member of DSC)

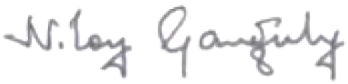
(Member of DSC)

(Member of DSC)

(Member of DSC)

(Supervisor)

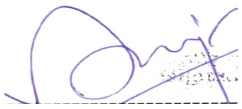
(External Examiner)



(Chairman)

CERTIFICATE

This is to certify that the thesis entitled **"Investigating core periphery structure for centrality resilience, prediction and representation learning"***, submitted by* **Soumya Sarkar** *to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy, is a record of bona fide research work carried out by him under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute. To the best of my knowledge, the results embodied in this thesis have not been submitted to any other University or Institute for the award of any other Degree or Diploma.*



Animesh Mukherjee

Associate Professor

CSE, IIT Kharagpur

Date: 17/09/2020

DECLARATION

I certify that

a. The work contained in this thesis is original and has been done by myself under the general supervision of my supervisors.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Soumya Sarkar

ACKNOWLEDGMENTS

I would like to take this opportunity here, to thank all those people who have directly or indirectly walked this path with me toward completion of this thesis. They say that *A journey of a thousand miles begins with a single step.* This is where I take the liberty to highlight some of those defining people, who have supported and guided me at every step of this journey.

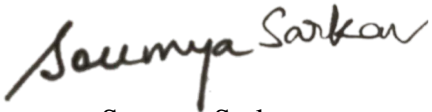
I would first thank Prof Joydeep Chandra, my mentor during final year project in the MTech programme. I would not have the courage to pursue career in research albeit the guidance of Prof Chandra, who explained to me how research can be fulfilling. He also introduced me to the Complex Network Research Group (CNeRG) and played a major role in finding me a suitable position in this group, which matched my interests. I would express my deep gratitude for my PhD supervisor Prof Animesh Mukherjee. The most unique thing about Prof Mukherjee is that he is equally invested in the research problems that his students pursue, even if in most publications, he is often credited as the last author! His enthusiasm is one of the primary reasons I did not loose hope on occasions when I was not making desired inroads, in my research endeavours. Besides technical inputs, Prof Mukherjee has always extended help in navigating bureaucratic maze of IIT Kharagpur which often ended in eating up his personal time. He has always made sure that I have grants for my PhD scholarship as well as conference travels which helped a lot in reducing stress, during my stay at IIT Kharagpur.

Besides my supervisor I would like to thank all my collaborators. I am indebted to

Prof Sanjukta Bhowmick of University of North Texas. I have had several discussions with Prof Bhowmick at various stages of this thesis and she has never shied away from having discussions even at odd hours. I would also thank Dr. Indrajit Bhattacharya, Principal Scientist at TCS Innovation Labs for taking time out of his busy schedule and answering my doubts. I would like to express gratitude to my seniors especially Sandipan Sikdar, Abhik Jana, Surjya Ghosh, Soumajit Pramanik, Suman Kalyan Maity and Mayank Singh for always taking time out, lending an ear to my research problems and giving me honest feedback.

I would like to thank some of my colleagues such as Ayan Kumar Bhowmick, Rohit Verma, Satadal Sengupta, Madhumita Mullick, Rijula Kar, Subhendu Khatuya, Sankarshan Mridha, Amrith Krishna, Sumitro Bhoumick, Binny Mathew, Gourab Patro, Abhishek Dash, Soumyajit Chatterjee, Rajdeep Mukherjee and Rajat Sadhukhan. The camaraderie I shared with some these guys will stay with me for a long time and I will remember the late night tea and philosophical discussion sessions.

Last but not the least, no words, sentences or even entire books would be sufficient to express my gratitude for my family. My parents have always supported my goals for higher education, sometime even at deep personal expense. This thesis would not have existed without their support. The contributions of my wife toward this thesis also cannot be measured by any means. She has sacrificed her time and energy in caring for my family which included single handedly raising my infant daughter Arvika in my absence, throughout the initial 3 years of our married life. I can only hope that I have been able to do justice to this sacrifice, through this thesis.



Soumya Sarkar

Kharagpur, India

Author's Biography

Soumya Sarkar has received his B.Tech degree in 2012 from Dept. of Information Technology, RCC Institute of Information Technology. He obtained M.Tech. degree in 2015 from Dept. of Computer Science and Engineering, Indian Institute of Technology, Patna. He has been pursuing PhD degree in Dept. of Computer Science and Engineering Indian Institute of Technology, Kharagpur from 2015

Publications from the Thesis

1. **Soumya Sarkar**, Sandipan Sikdar, Sanjukta Bhowmick, and Animesh Mukherjee. *Using core-periphery structure to predict high centrality nodes in time-varying networks.* Data Mining and Knowledge Discovery 32, no. 5 (2018): 1368-1396. accepted in ECML PKDD 2018 Journal Track.

2. **Soumya Sarkar**, Sanjukta Bhowmick, and Animesh Mukherjee. *On Rich Clubs of Path-Based Centralities in Networks.* In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 567-576. ACM, 2018.

3. **Soumya Sarkar**, Aditya Bhagwat , Animesh Mukherjee *Core2Vec: A Core-Preserving Feature Learning Framework for Networks.* In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 487-490. IEEE, 2018.

4. **Soumya Sarkar**, Aditya Bhagwat, and Animesh Mukherjee. *A core-periphery structure based network embedding approach.* (in submission at Social Network Analysis and Mining (SNAM) Journal .).

ABSTRACT

Complex systems modeled as networks, has emerged as an exciting field of study in the last decade. This is primarily due to the fact that our day to day lives are deeply intertwined with "hopelessly" complex systems. Modern communication infrastructure, which connects any two people in seconds materializes due to co-operation of billions of devices such as routers, cell phones and computers. Social media exists due to interaction of billions of entities such as people, organisations etc. which leads to spectacular phenomenon such as information cascade. Even our inherent capability to comprehend society around us requires seamless interactions between billions of neurons in our brain. Since networks lie at the center of social, technological and biological systems, it is important to come up with mathematical description of the structure, which may lead to optimal control of the underlying processes.

Resilience of a complex network relates to how well some of its properties are retained under attacks. Robustness has been historically studied from the perspective of node property of degree. This translates to how essential connectivity in the network can be disrupted by random and targeted node failures. Understanding robustness is crucial in tackling scenarios such as the catastrophic 2003 blackout in America, Canada as well as 2008 global financial meltdown. However existing works do not shed light on how other key properties are affected such as **centrality** due to random failures.

Centrality resilience is more difficult to detect than connectivity resilience. When one part of a network cannot communicate with the rest of the system, it is easy to infer that the cause is due to disconnectivity. Attack on centrality, however, may not disconnect the network, but result in longer distances when traversing the network.

The increased length of the distances, is due to the change in the ranking of the high centrality vertices which may not be immediately apparent until the centralities of the system are recomputed. This may potentially lead to delays in transport network or high latency in communication resulting in economic losses.

In this thesis we show that path-based centralities form dense clusters or "rich clubs" in certain networks, which manifest in the **inner cores** of a network. We demonstrate that stability of high central nodes in the network is related to these substructures. We empirically and theoretically show that "rich clubs" exists, if the **core-periphery structure** of the network is such that each shell is an expander graph, and their density decreases from inner to outer shells. We extend the concept of a single rich club to that of "scattered rich clubs" and explain how they connect to centrality resilience. We subsequently extend our analysis to time-varying networks and develop approaches to predict high central nodes based on the stability of the **core-periphery structures**. We finally show that **k-core structures** can be useful is developing novel network representation learning algorithms which is effective in various downstream prediction tasks.

Keywords: core periphery structure, centrality resilience, centrality rich club, scattered rich club, network representation learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preamble

Study of networks lies at the heart of understanding complex structures. These structures can be from various domains ranging from social, technological and biological systems. In spite of the diversity, studying these systems under the common framework of networks, has yielded fruitful insights. As an example, friendship or professional ties are abstracted as links, to form a network of social relations. Organizations such as *LinkedIn, Facebook, Twitter* etc. have invested considerable resources studying these networks, as well as underlying processes to discern individual or group behaviour. The communication infrastructure comprises billions of interconnected devices which are rigorously studied through projects such as *caida* or *dimes* [1, 157], to understand possible vulnerabilities. Mysteries of cognition and human reasoning are explored, by mapping connectivity patterns generated by billions of neurons in the human brain. Considering the universal applicability of networks in interdisciplinary domains, *network science* has emerged as an exciting field of exploration.

One of the key sub-problems in network science research is *resilience*. A resilient or robust system can carry out its essential functions in spite of a small fraction of component failures. For instance, a communication infrastructure such as the Internet functions reliably even when at any point in time, several routers may be malfunctioning. Similarly, metabolic

networks of living cells functions without hindrance, even when there are missed reactions. Understanding origins of this remarkable resilience in natural systems has been a topic of comprehensive exploration in the last decade. One of the seminal works in these lines by Barabasi et. al. [6] show that real world networks are immune to random node failures. However, they have low tolerance toward targeted attacks of high degree nodes. Hence real world networks are prone to be dissolved readily into disjoint components when an adversary launches targeted attacks. The origin of this property is rooted in the degree distribution of real world networks, which has been shown to follow a power law behaviour. Hence majority of nodes have small number of neighbors while a small proportion of nodes dominate the degree distribution. This inherently implies that random failure of components will not affect the network adversely, while targeted attacks can critically damage the system.

Another intrinsic vulnerability of networks lies in situations of cascading failures. In such cases, failure at one or few nodes distributes the load to their immediate neighbors. On exceeding capacity, this imbalance further escalates to the local neighborhood. Cascading failures have been observed in electrical transmission networks, with one of the most prominent examples being the 2003 blackout in Northeast United States and Canada which affected 55 million people[1]. The 2009-2011 financial crisis is also an instance of cascading failure which ended up paralyzing the world economy, leaving behind substantial financial turmoil[2]. Automatic detection of focal points of cascading failure is an ongoing research direction with diverse applications ranging from infrastructure, economic to healthcare systems. An interesting application lies in cancer research where inducing sequence of failures in our cells may eventually eliminate malignant tissue [11]. These strategies are also applicable in criminology toward crippling money laundering networks [11].

Vulnerability of networks has been mostly studied from the perspective of how network robustness changes when local neighborhood varies. Apart from degree, other key node properties such as 'centrality' have not been explored rigorously in the pursuit of understanding resilience. Centrality resilience relates to how much the rankings of the top-k central nodes in the network are perturbed under random or targeted removal of edges with the additional constraint that any such link removal does not disconnect the network. In case of traditional resilience, attack strategies will fundamentally alter the structure of the network. However

[1] en.wikipedia.org/wiki/Northeast_black out_of_2003

[2] en.wikipedia.org/wiki/Global_financial_crisis_in_September_2008

constrained edge removal will only increase the distance between multiple pairs of nodes in the graph. Hence efficient operational capability of a network can be disturbed. As an example in a transportation network if centrality ranking is not robust, small amount of edge removal will impact average distance between two nodes, which may lead to delays, leading to economic losses. Therefore, centrality resilience is a very powerful tool for insidiously disrupting the functioning of a system, without a drastic change to its structure. Since it is extremely expensive to periodically recompute centralities of the entire network and validate stability of rankings, a natural question that arises is that how can we utilise 'cheap-to-compute' structural properties of the network to ascertain the centrality resilience.

Resilience is often associated with a network substructure commonly known as *rich club*. Rich club emerges in a network when multiple hubs are interconnected among themselves. Most of the existing works have characterised richness of nodes from the perspective of node degree. However, other salient properties such as centrality have not been explored. Rich clubs are also associated with the core-periphery structure in networks. Core-periphery organization in networks can be extracted using *k-core decomposition* [12], which is a scalable approach of extracting dense subgraphs from the network. Since by definition of the core-periphery organization, innermost cores are deeply embedded in the network, we posit that a large fraction of shortest paths in the network will pass through the inner cores. Hence path based central nodes, i.e., nodes with high closeness and betweenness centrality will form a rich club within the inner cores of the network. We perform a detailed investigation regarding the relation between core- periphery structure and centrality resilience in this thesis. We further study stability of k-core structure in time varying networks and develop an approach for predicting high central nodes a priori. Finally, we also develop novel representation learning algorithms leveraging k-core structure which results in scalable node embeddings, useful for downstream prediction tasks.

1.2 Objectives

The three main objectives which govern the investigations carried out in this thesis are laid out below.

1.2.1 Rich centrality clubs in networks

In many real world networks the high degree nodes form a densely connected subgraph. This is known as the rich club phenomenon. Our objective here is to extend the definition of rich clubs from high degree vertices, to shortest path based centralities, particularly high closeness and betweenness centrality vertices. We also investigate how the k-core structure is connected to the notion of centrality based rich clubs. We attempt to group networks into two categories. In the first category we find networks in which the high central nodes are part of the innermost core and the second category where this property is not observed. We intend to study global topological properties for each category of networks and identify discriminating factors. Finally, we investigate how these properties can be leveraged into building novel applications.

1.2.2 Identifying influential nodes in time varying networks

Our objective here is to extend the categorization of networks based on the localisation of high central nodes in the inner cores, to dynamic time varying networks. An important problem in time varying networks is to know a priori, using minimal computation, whether the influential vertices of the current time step will retain their high centrality, in future time steps, as the network evolves. We aim to identify novel structural properties based on core-periphery organization, which help us identify classes of networks where high central nodes are part of inner cores and this property does not change significantly as the network evolves. We further aim to use time series based models to predict the overlap between the high centrality vertices in the current time step to the ones in the future time steps. A predictive approach allows us to avoid expensive shortest path computations in each step as the network changes.

1.2.3 Connecting deep neural approaches to network core-periphery

Our third and final objective revolves around developing novel network representations using k-core structures. State-of-the-art node embedding technique define a proxy of node

neighborhood using various random walk based approaches and learn representations such that neighboring vertices end up with similar embeddings. Core-periphery organization enforces an organic hierarchy in the network such that vertices which lie in similar levels can be deemed to have similar characteristics. This motivated us to seek novel node embedding approaches leveraging the k-core structure.

We further observe that nodes in the inner cores of the network have higher influence compared to vertices in the outer fringes or periphery. We sample a random set of nodes from the inner/outer cores of the network and label inner core nodes 1 (influential) and outer core nodes -1 (non-influential). We find that we can formulate a semi-supervised learning task using these labelled nodes utilizing recent advances in graph convolution network (GCN) framework. More precisely, we can classify unlabelled nodes into influential/not-influential category by learning from the representations of labelled counterparts. We recommend k nodes, which the classifier has highest confidence based on softmax probabilities of the final layer. Using this framework we can avoid explicit computation of highly influential nodes and obtain considerable speedup.

1.3 Contributions

The contributions that we make in this thesis to achieve the objectives outlined in the previous section are as follows.

1.3.1 Rich centrality clubs in networks

Many scale-free networks exhibit a 'rich club' structure, where high degree vertices form tightly interconnected subgraphs. In this thesis, we explore the emergence of rich clubs through shortest path based centrality metrics. We term subgraphs of connected high closeness or high betweenness vertices as rich centrality clubs (RCC). We explore the possibility of RCC's existing within the innermost core of the network. Networks often exhibit hierarchical nested subgraphs, such that each inner subgraphs are progressively denser. Consider an undirected graph G with V as the set of vertices and E as the set of edges. Let K be a

subset of vertices, i.e., $K \subseteq V$ and $G(K)$ be the graph induced on G by the vertices in K. $G(K)$ is considered to be k-core of the graph G only if

- For every $v \in K$, $d_{G(K)}(v) \geq k$ where $d_{G(K)}(v)$ denotes the degree of v in $G(K)$.
- For each $K \subset K' \subset V \; \exists \; u \in K' \backslash K$ such that $d_{G(K')}(u) < K$. All such u form the K' *shell* of the graph.

Since by definition, inner cores are embedded deep inside the network, we posit that large proportion of shortest paths in the network goes through the inner core. This implies that path based high central nodes are localised in the inner core. Our study is further motivated by the fact that over the last few years several papers [60, 104, 117, 138] have independently reported that vertices in the inner shells of the networks can be leveraged to identify high influential nodes or serve as seeds for community detection. However, each paper focused on only one type of hypothesis and there was rarely any overlap between the networks studied in these papers. We conducted an integrated study over a large set of real-world and synthetic networks and observed that the reported properties of the vertices in the inner shells hold only for a certain type of networks. This observation impelled us to investigate the topological property of networks where the inner shells contain high centrality nodes.

Structure and function of RCC

We observed that the inner shells of networks are typically dense, thus if they contain high centrality nodes, then by virtue of being dense, these cores would form a rich centrality club. However, unlike degree which is a local variable, closeness and betweenness centralities are based on shortest paths which are global variables. Building on this observation we demonstrate spectral properties of networks with/without RCC. Spectra of networks with/without RCC show distinctive signatures. We also show that presence of rich centrality clubs confers several favorable properties to the networks. In particular, due to the presence of path based central vertices within a small subgraph, the vertices in the RCC can be effective seed nodes in quickly spreading information across the network. Moreover, similar to the traditional rich club, the presence of RCC increases the resilience of the networks under edge perturbations. Given these favorable properties, we posit that,

in many cases, the presence of RCC is desirable. To this end, we propose a modification model that can implant a RCC in a network where it is absent. Our model is such that other properties of the original network including the power law exponent, the average degree, clustering co-efficient remain unchanged.

Scattered RCC

Our rich club definition is restrictive because it considers a single rich club, formed at the innermost cores of a network. This feature occurs only in specific types of networks. In order to understand centrality resilience of all types of networks, we extend the concept of a single rich club to that of *scattered rich clubs*. Scattered rich clubs are connected high centrality vertices, that may be spread across the network. We empirically demonstrate that rich clubs of centrality vertices can be spread across multiple cores of a network. We further demonstrate that analogous to the single rich club being formed in the innermost cores, the scattered rich clubs are present in the innermost cores of a meta network. This meta network can be constructed by extracting communities and cliques in the original network. This is indicative of a second order dependency between the substructures affecting centrality resilience.

To summarize, our **key contributions** are as follows.

- We study the formation of *rich clubs of shortest path based centralities* in complex networks and observe that their presence can lead to faster identification of high centrality nodes and communities. We demonstrate empirically and theoretically that *in networks containing RCC, the shells are expander-like and the density of the shells decreases from the inner to the outer shells*.
- Develop novel attack models and show that networks containing RCC are resilient to perturbations to their edges. Besides, *the vertices within the RCC are effective seed nodes* for information spreading.
- We propose a *modification model that can insert RCC* into a network, while maintaining other structural properties of the original network. Our model is reversible in

that when operations are applied in reverse (deletion instead of addition of edges), the RCC can be removed from a network, while also maintaining the other structural properties. Our model only requires the information of the degree of the vertices, which is a much faster operation than computing the betweenness and closeness centralities.

- We propose the concept of scattered rich clubs and show that rich club of central nodes can be scattered across the network as opposed to being concentrated at the core. They form a part of the innermost core of the second order meta network whose nodes are cliques in the corresponding original network.
- We provide theoretical justification supporting our empirical results.

1.3.2 Identifying influential nodes in time varying networks

One of the important problems in time-varying networks is predicting how their features change with time. If this information is known a priori using minimal computation, then users can take appropriate action in advance to utilize such features. The most significant among network properties are the centrality features, that are used to estimate the importance of a vertex in a network. Information can spread more quickly when high closeness centrality vertices are selected as the initial seeds. Similarly, vaccinating high betweenness centrality vertices, through which most of the shortest paths pass, can reduce the spread of disease. The central vertices also play an important role in spreading influence in a social network as has been observed in several works [5, 108]. In a dynamic setting (where the network changes over time) knowing these highly central vertices beforehand is of prime importance as it would facilitate in developing strategies for targeted advertising or setting up infrastructure for vaccination drives. However, this might result in expensive re-computation of shortest paths as the network varies over time. Our goal is to develop algorithms based on the network structure, so that such re-computations are avoided.

Drawbacks in current approaches

Current approaches focus on predicting the average centrality values of the network [89]. However, note that most applications, such as the ones discussed above, require the ids of *only the top-k centrality vertices*, not the values or the ranking of *all* the vertices in the network. Therefore, simply predicting the average centrality over the entire network may not be useful in a majority of the practical contexts.

Our approach

We **present a two-step algorithm for predicting the high centrality vertices of time-varying networks**. In the first step, we predict the overlap between the set of high centrality vertices of the current time step to the set of high centrality vertices of the future time step. In the next step, assuming that the network snapshot is already available in time, we analyze its innermost core to find the ids of the high centrality vertices. The key to our prediction method is based on the *hypothesis* that in many real world time-varying networks, *most of the highly central vertices reside in the innermost core*. In other words, a *large fraction of the shortest paths connecting pairs of vertices in such networks, pass through the innermost core*; the vertices in the periphery (and the outer cores) of the network are mostly connected via the vertices residing in the innermost core of the network. A key contribution of our work is that we develop a set of *novel heuristics to classify networks based on the extent to which the highly central vertices are in the innermost core*. We define the heuristics below:

- **Fraction of inter-edges connected to the top core (EF)**: We calculate the ratio of the number of inter core edges where one end point is in the top core to the total number of inter core edges in the network. Given this ratio is high, inner core nodes play major role towards connecting a random pair of nodes in the network.
- **Average density of the non-top cores (CFX)**: This metric computes the average density of all cores, except the top one. The lower the density, the sparser the core, and the higher the average intra-core distance.

- **Density of the top-core (ED)**: We compute the density of the top core which is the ratio of the number of intra-core edges in the top core to the total possible edges between the vertices in the core.

- **Top-core overlap (CV)**: This metric takes into account the changes in the top core structure over consecutive time steps. We measure the overlap as the Jaccard similarity between the vertices in the top cores of networks at two successive temporal snapshots.

We develop a classification framework based on the above metrics. Consider that we obtain $t-1$ temporal snapshots $G_1, G_2, ..., G_{t-1}$ for an initial set of networks. For each snapshot we calculate the heuristics hence each temporal graph G_t can be represented by 4 tuple, i.e., (EF, CFX, ED, CV). We calculate the cumulative distribution function for each parameter from the $t-1$ values obtained for that parameter. We perform hierarchical clustering on each parameter, using D-statistic as the pairwise similarity measure. We cut the dendrogram at cluster size two, hence having the clustering algorithm output two clusters (C_1, C_2) for each parameter. If the mean value of parameter in C_1 conforms more to the desirable property, i.e., have high values for EF, ED, CV or have low value for CFX, we consider C_1 as desirable cluster and C_2 as undesirable cluster. If an unseen network arrives, we compute the CDFs for each of the four parameters from the $t-1$ snapshots. Next we obtain the similarity (D-statistic) of the CDF of a particular parameter with the centroid of both C_1 and C_2 (i.e., the traditional Rocchio technique [10]). Finally, we classify the network to that class to which it is more similar based on the similarity with the centroid of the class. For a network which can be classified into a desirable group, time series models such as ARIMA can be applied successfully toward predicting high central nodes.

Validation of our framework

We further validate our results by comparing how the predicted and actual high centrality vertices perform in a practical context. For high closeness centrality vertices, we compare the time to spread a message when the high centrality vertices are taken as seeds, and for the high betweenness centrality vertices, we compare how the length of the diameter increases

as the high betweenness centrality vertices are deleted from the network. For these experiments we select a set of random vertices as control, and compare the performance of the actual high centrality vertices, predicted high centrality vertices and the randomly selected vertices. For networks where we could predict the results with high accuracy, the effect of the original and predicted vertices are very similar, and these results are markedly different from the effect on the randomly selected vertices. Interestingly, for networks, where our prediction accuracy is low, the effect is similar for closeness/betweenness centrality for all the three sets of vertices. This result indicates that networks with low prediction accuracy do not have significantly high closeness/betweenness centrality vertices and therefore the prediction itself does not serve any practical purpose.

To summarize, our **key contributions** are;

- We develop a feature set to calculate the extent to which high central nodes in the current temporal snapshot of a dynamic network will retain their importance in future time-steps. Based on these features we propose a *Rochio algorithm* based supervised classification technique for temporal networks into two predefined classes. In the first class of networks path based high central nodes are localized in the inner cores of the network, hence can be easily detected without explicit computation. In the second class of networks high central nodes cannot be extracted avoiding some form of explicit shortest path enumeration.
- In contrast to existing centrality prediction approaches which focus on predicting average centrality of the temporal snapshots, we focus on predicting *top k* high central nodes in the network. We argue that such an approach is more practically useful in downstream tasks like influence maximization. We show the efficacy of our approach through comparison against several existing baselines on centrality prediction task in dynamic networks.
- We devise useful downstream tasks, namely message spreading and increasing network diameter, to show that the predicted vertices indeed perform similar dynamic roles in the network, when compared against actual nodes.
- Finally, we present a theoretical rationale behind the workings of our method.

1.3.3 Connecting deep neural approaches to network core-periphery

The conventional paradigm of handcrafted feature engineering to generate node representations in networks has been largely overhauled due to advances in techniques which automatically discover and map a node's structural properties into a latent space. These techniques are useful because manual feature engineering requires extensive domain knowledge as well as tedious exploration of structural properties such as degree, centrality, clustering coefficient etc. Without loss of generality, representation learning encompasses the task of transforming a graph $G(V, E)$ from $V \rightarrow \mathbb{I}^{|V|\times|V|}$ to the mapping $V \rightarrow \mathbb{R}^{|V|\times d}$ with the constraint $d << |V|$.

Unsupervised approach for generating rich node representations relies on developing random walk based exploration strategies from each node. Two nodes which fall in the same exploration path multiple times are embedded close in space. This technique is an application of the word2vec framework [124], where the objective is to bring two words, which co-occur in the corpus, close in the vector space. Supervised approaches on the other hand generate a task specific representation. Given a target downstream task such as node classification, semi supervised approaches such as graph convolutional network [62, 91], generate representations for unlabeled nodes by learning from a small fraction of labeled nodes. These representations are useful in discriminating nodes into task specific classes. In this chapter we first develop a novel unsupervised network representation approach utilizing core-periphery structure. We further show that the information of the k-core structure is also useful in generating embeddings which help in locating influential nodes in the network.

Unsupervised network embedding techniques

Unsupervised network embedding problem is efficiently solved by applying a *skip-gram model with negative sampling* [124], which is a celebrated technique for learning meaningful vector representations for words. To represent a target word, nearby co-occurring words in the sentence are considered as context words. Adapting this framework for graphs, there have been several works such as [139, 167] which learn social representations of a graph's

vertices, by learning from neighbor nodes generated from short random walks. *These walk sequences act as proxy for context words in a sentence*. One of the key drawback in these works is the assumption that the context nodes can be always efficiently generated by walk sequences from a source node thus building a sample set appropriately representing the structural and the functional properties of the source node.

Our proposal

We propose a solution to the above problem by developing an algorithmic framework, *core2vec* which utilizes intermediate-scale structure of the network, i.e., the core-periphery structure, for learning the feature representation of a node. Nodes in similar cores have similar connectivity profiles and assume comparable roles in the network. We leverage this nested “onion like structure" in real world complex networks, to develop a flexible biased random walk which seeks similar core nodes as context nodes for a source vertex. More specifically we develop a strategy to guide a random walk sequence to identify similar core nodes both in close proximity as well as distant neighborhood. Our experiments show that our scheme brings nodes with similar core ids closer (*closeness*) as well as separates nodes with different core ids (*separability*) farther in the vector space compared to state-of-the-art methods like node2vec [63], DeepWalk [139] and LINE [167], thus establishing the necessity of our approach. Our method utilises the property that nodes in similar cores possess similar connectivity profiles and assume similar roles in the network.

Validation of core2vec

We validate the effectiveness of our scheme by estimating similarity of words (nodes) in word association networks. Networks built from linguistic units (e.g., word co-occurrence and word association networks) are known to have a well-defined core-periphery structure (see [33] and the references therein) and hence the motivation to choose word association networks for our validation purpose. We learn representations of each node in two large word association networks using core2vec as well as using similar baselines. We next estimate the (cosine) similarity of the vector representations of word pairs, rank these word pairs based on the similarity obtained and compare the same with the ranking of the pairs

drawn from different ground-truth datasets on word similarity. We compare the rankings using the Spearman's rank correlation co-efficient and show that we always outperform the baselines and by at most 46% in certain cases. We further project the representations obtained by our method and comparable method on 2D space using PCA and show *core2vec* performs better in bringing semantically related words closer.

Semi supervised network embedding techniques

Semi supervised learning (SSL) approaches are well known in graph based learning tasks [190, 191]. However these techniques have become particularly popular over the last couple of years due to introduction of graph convolutional architecture (GCN). GCNs are generalizations of convolutional neural networks, and work efficiently on general grid like geometric data akin to graphs. In most real world scenarios graphs are arbitrarily large in size hence it is expensive to obtain ground truth labels for all nodes in the network. SSL approaches are useful in cases where limited label information is available in structured data. Given $\boldsymbol{A}$ as the adjacency matrix, $\tilde{\boldsymbol{D}}_{ii} = \sum_j (\boldsymbol{A} + \boldsymbol{I})_{ij}$ and $\mathcal{X}$ denotes known node representation, the GCN framework updates $\mathcal{X}$ in each iteration using the Equation 5.3 below.

$$\boldsymbol{H} = f\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}})\mathcal{X}\boldsymbol{W}\right) \tag{1.1}$$

Here $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ are the model parameters and f denotes non-linear activation function. $\boldsymbol{H}$ is subsequently used to predict the class label of the downstream classification task. The error in case of labeled nodes are backpropagated using gradient descent approaches.

Influential node prediction

We propose a task of ranking nodes based on network influence using GCNs. Influence of nodes in network can be computed using centrality metrics such as closeness, betweenness which is expensive. We propose to alleviate this problem by learning discriminating representation of nodes which aides in node ranking. In our investigations, we have observed that nodes with high coreness have higher influence compared to nodes in the periphery.

This information is obtained using k-core decomposition [12] in *O(E)*. Once the core information is available, we label a random sample of nodes with high coreness as label 1 and equal number of random nodes in the outer cores as -1. For each unlabeled node, positive and negative label distribution is predicted by the model by learning from the local structure of the training labels. Once the predicted labels are obtained we recommend top-n positive nodes with respect to class probabilities as possible influential nodes. In our experiments we find that the update method proposed by Kipf et al. [91] akin to Equation 5.3 does not work well in generating useful representations. We modify the update setup by introducing a novel masking scheme (Equation 1.2). Here $\mathcal{C}$ is the adjacency matrix such that distant core edges are removed. This method eliminates edges which span distant cores and increases network homophily. Our technique reduces morphing of useful features in the network by reducing the neighborhood set.

$$\boldsymbol{H} = f\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\left(\tilde{\boldsymbol{A}} \odot \mathcal{C}\right)\tilde{\boldsymbol{D}}^{-\frac{1}{2}})\mathcal{X}\boldsymbol{W}\right) \tag{1.2}$$

To summarize, our **key contributions** are as follows:

- We design a novel network embedding model *core2vec*, with a unique exploration strategy for context nodes guided by global information. Our technique does not require any compute intensive meta information like community label or domain specific node level attributes.
- We demonstrate that our model can favorably map similar core nodes closer in space and distant core nodes farther in space.
- We illustrate an application of our embedding in word analogy task, where we compare similar words scored by our method against manually curated gold standard scores. Our method can predict similar words better from the baseline techniques.
- A novel SSL approach for ranking influential nodes based on core-periphery structure. We introduce a novel masking technique which works well compared to vanilla aggregation setup.

1.4 Organization of the Thesis

The thesis is organized into six chapters.

Chapter 2 presents a detailed literature survey of the metrics, analysis and methods pertinent to this thesis.

In **Chapter 3**, we demonstrate empirical evidence that for a class of networks, path based high central nodes reside in the innermost core. This naturally result in the emergence of a second class of networks where high central nodes do not lie in the innermost core. We coin the former dense subgraph as rich centrality club (RCC). We show discriminating structural properties in both classes of networks. We further establish useful properties of RCC in terms of information propagation and resilience. Since RCC is a desirable property in networks we develop a novel method for implanting RCC in networks without altering general network property like degree distribution, clustering coefficient etc. Leveraging communities and cliques inside communities, we construct a meta network from the original network and show that in the second class of networks, it is possible to locate high central nodes in the inner core of this meta network. Finally we present theoretical justification of our empirical results.

In **Chapter 4** we deep dive into our second objective, i.e., predicting high central nodes in temporal networks utilizing a k-core structure. We first propose a novel set of heuristics which help us classify networks into two categories. Given these two categories we show application of time series models such as ARIMA, ARMA, MA to identify ids of the top central nodes in the network. We further validate our prediction framework in details with extensive experiments. Finally we provide a brief theoretical justification of our prediction framework.

In **Chapter 5** we first propose a novel network representation learning framework (core2vec) utilizing hierarchical structure induced by k-core decomposition. We explain our methodology in details and further validate on real world word association data. We show that our framework can map semantically similar words close in vector space better compared to similar baselines.

Besides random walk based approach, we also develop a novel semi supervised method to identify influential nodes in the network utilising graph convolution networks (GCN). We develop a novel masking approach on top of the vanilla GCN and further illustrate its utility in predicting influential nodes in the network.

Chapter 6 concludes the thesis by summarizing the contributions and pointing to a few topics of future research that have opened up from this work.

Chapter 2

Related Work

In this chapter, we review the relevant literature related to the objectives laid out in this thesis. We organise the related work as follows. We discuss definitions of network measures pertinent in this thesis in the first section. This is followed by an elaborate discussion on applications developed based on these measures and potential drawbacks which we address here. In the second section we cover relevant topics in this thesis connected to temporal networks as well as time series models, we leverage in our work. Finally in the last section we make a detailed survey of network representation approaches covering both unsupervised methods as well as state-of-the-art graph convolutional networks.

2.1 Centrality metrics

One of the key questions which drives considerable research effort is "which nodes in a network are influential?". There exist several measures inspired by social science, which quantifies influence of a node based on his/her local or global position in the network. We first enumerate some of the metrics we study in this thesis. Throughout this thesis we denote a graph by G, where V, E are the set of vertices and edges respectively.

Degree centrality: This is the simplest form of centrality in the network which calculates total number of edges connected to the node. It is intuitive in social science literature to

assume that nodes which have high degree have dominating social prestige [132]. However in real world web scale graphs it is possible for malicious nodes to garner high degree centrality using link farming techniques [58].

Closeness centrality $(C_lC(v))$**:** This metric calculates the average of the shortest distance between a vertex v and all other vertices in the network. It is calculated as $C_lC(v) = \frac{1}{\sum_{s \neq v \in V} dis(v,s)}$, where $dis(v, s)$ is the length of the shortest path between v and s. The shortest path computation is performed using Dijkstra's algorithm [41]. The inverse of the shortest distance is considered because in case node s is unreachable, $dis(v, s)$ is infinite. Unreachable nodes do not contribute to overall computation of C_lC.

Betweenness centrality $(B_wC(v))$**:** Betweenness centrality proposed by Freeman et al. [55] for a vertex v is the ratio of the number of shortest paths between a vertex pair that passes through v and all the shortest paths possible between that pair. It is given by $B_wC(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the total number of shortest paths between s and t, and $\sigma_{st}(v)$ is the total number of shortest paths between s and t that pass through v. In our experiments we use the computation algorithm proposed by Brandes et al. [21] which has a complexity of $O(|E|*|V|)$ for undirected, unweighted graphs.

k**-core (**$G_{k_{max}}$**):** A *k-core* of a graph G is a maximal subgraph $G_{k_{max}}$, such that each of its nodes has degree at least k_{max}. If graph G is connected we can consider it as G_{k_1}, i.e., a *1-core*. Recursively removing all nodes of degree one leads to a subgraph where all nodes having degree at least which is a *2-core*. This process of peeling the network to discover *k-core* subgraphs is called *k-core* decomposition [12]. This process has complexity $O(|E|)$ and assigns a unique core number $(\mathcal{C}_v)$ to every node $v \in V$. If $\mathcal{C}_v = k_m$ it implies that $v \in G_{k_m}$ and $\forall G_{k_i}$ possible, G_{k_m} is the maximum subgraph which v can be part of. The innermost *k-core* subgraph is also referred to as the *graph degeneracy*.

k**-shell (**S_k**):** *k-shell* is a set of vertices such that each node in the *k-shell* has at least k connections to nodes in the k-shell or higher shells in the network. We show a toy example in Figure 2.1. Core-periphery structure has been generally defined in existing literature as either *hub and spoke* formulation of *onion layered* or *series of shells* formulation [56]. In this thesis we have followed the later formulation however for detailed exposition of the former typology, we refer to the seminal work of Borgatti et. al. [17]

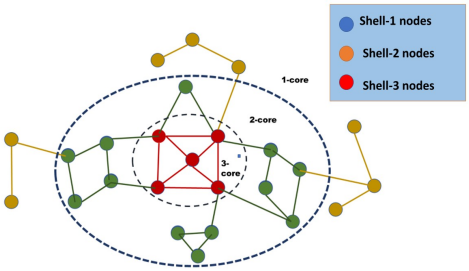


Figure 2.1: Illustrative example of a network decomposed into hierarchical subgraphs indicating different k-cores. Nodes of different shells are colored.

Rich club: *Rich club* is a key structural organisation where group of nodes with high degree centrality, also known as hubs are connected among themselves [40]. A quantitative definition is provided by Zhou et al. [188,189], where authors calculate rich club coefficient as $\phi(k) = \frac{2*E_{>k}}{N_{>k}*(N_{>k}-1)}$. Here $N_{>k}$ denotes nodes with degree at least k and $E_{>k}$ denotes their corresponding links.

Spectral gap ($h(G)$): Spectral gap is an approximate measure for the *expansion factor* in a graph. The expansion of a set of nodes $S \subset V$ where $|S| \leq \frac{|V|}{2}$ is a function of the number of edges connecting $V - S$ to S. That is, if $N(S)$ is the set of nodes to which S is connected, then the expansion of S is $\frac{|N(S)|}{|S|}$. *Expansion factor* is defined for a graph such that it is the minimum expansion for all possible subsets, i.e., $\min_{\forall S \in V : |S| \leq \frac{|V|}{2}} \frac{|N(S)|}{|S|}$.

Expansion factor offers help in deciphering structural cues of a graph, in particular they can inform us about the presence (or absence) of edges which can act as bottlenecks inside the network. This practically means that measuring the expansibility of a graph, enables one to know to what extent the graph can be broken into disjoint modules. Large expansion factor implies a large fraction of edges need to be removed from the graph to arrive at disjoint subsets of comparable sizes [116].

Accurate calculation of the expansion factor is NP hard [126]. For d-regular graphs it can be approximated by the second smallest eigenvalue of the spectrum of the normalised graph Laplacian given by $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. The second smallest eigenvalue (λ_2), also known as the spectral gap, is related to Cheeger constant by $\frac{\lambda_2}{2} \leq h(G) \leq \sqrt{2\lambda_2}$ which is also know as Cheeger's inequality [80]. It was shown in [34] that the lower bound also holds

for general graphs.

2.2 Network applications

In this section we conduct a literature survey on some of the notable works in crucial network applications. We focus on information diffusion and robustness, which are the key applications we investigate in this thesis.

Information diffusion: Spread of information in networks is often considered in the same light as spread of epidemics in social contact networks. Hence several models relevant in the field of epidemiology has been applied in information diffusion for networks. This includes classical models like susceptible-infected-susceptible (SIS) model [46] where each individual in the population is either infected or susceptible at any point in time. Satoras et al. [136] discovered absence of an epidemic threshold in scale free networks [23], which imply real world networks are inclined to the spreading and the persistence of infections. Several variants of these models exist and have been validated. Eguiluz et al. [49] showed that variation of network topology such as high clustering coefficient and degree correlations dictate epidemic threshold in networks and further conjecture that connection between hubs play a crucial role in determining epidemic threshold. Similar results were reported by Moore et al. [128] where authors connected epidemic threshold with general percolation theory. Chakrabarti et al. [28] further showed that epidemic threshold is related to the largest eigenvalue of the graph adjacency matrix and developed counter strategies for virus propagation using non linear dynamic models. Several models exist to predict information propagation since they are applicable in viral marketing, planning interventions etc. Kempe et al. [86] proposed approximation algorithms with performance guarantees for basic diffusion models such as linear threshold model and independent cascade model in their seminal work. We use simpler variants of these models in some our works in this thesis.

Robustness: Robustness in network is generally concerned with understanding how network measures such as centrality metrics, core number etc. change under random or targeted perturbation of the network. In these lines one of first work was proposed by Borgatti et al. [16] Here the authors tested Erdős Rényi random graphs with four possible errors,

that can occur in the graph. These are node removal (random removal of a proportion of the existing nodes in the graph), node addition (insertion of a proportion of extra nodes into the graph along with new edges randomly added from each of the added nodes to existing nodes), edge removal (random removal of a proportion of existing nodes), edge addition (insertion of new edges not present in the original graph). Networks were perturbed with the above random errors and different centrality measures such as degree, betweenness, closeness, and eigenvector centralities were examined using Jaccard index or Pearson's correlation coefficient. The main result of their simulations was that the accuracy of the centrality measures in the perturbed graphs decreases with increasing error predictably and monotonically. Similar experiments were performed in [152, 153, 170] on random as well as real world networks and stability of centrality metrics were probed and a stable variant of betweenness centrality was proposed. The stability of rankings for PageRank metric in complex networks was examined under degree preserving edge rewiring based perturbation in [59]. Results show the emergence of "super-stable" nodes in scale-free networks. Adiga et al. [4] developed several useful noise models and show how inner core subgraph is affected following perturbation of network using these models. Laishram et al. [96] proposed strategies to make k-core robust under edge based perturbation. Ufimtsev et al. [171] used addition based noise models proposed by [4] and showed using empirical as well as analytical means that ranking of centrality metrics due to stochastic perturbation depend on local connections. In our experiments in this thesis we follow the perturbation strategy inspired by [96, 171].

2.3 Application of k-core decomposition

Correlation of coreness and centrality measures: Coreness has been shown to be correlated with several centrality metrics. A strong Spearman's rank correlation between degree and coreness has been presented in [158, 159] and the authors developed an anomaly detection system based on this correlation. In contrast, in [102] the authors showed that core number has low Pearson's correlation with centrality metrics such as degree, closeness and betweenness. An explanation could be that while many nodes in the network could potentially have the same core number, they would typically tend to be different in terms of the

centrality measures and thus Pearson's correlation would be low. A more accurate comparison would be to consider the overlap of the top ranked nodes based on core numbers and other centrality metrics, which is what we do here.

Coreness for community detection: *k-core* decomposition outputs an ordered partition of the graph after processing it hierarchically. In [60], the authors proposed that this hierarchical information can be utilized by any graph clustering algorithm to obtain more meaningful partitions. In [138], the authors proposed a framework to accelerate label computation for nodes by modularity maximization utilizing the *k-core* information. They estimate a maximum speedup of 80% through rigorous experiments. Wang et al. [176] utilized *k-core* information to speedup classical community detection method such as CNM [38] by at most 25%. We show in this thesis that such methods are not useful for all real world networks; however, there is a special class of networks for which these methods work very well.

Coreness for spreading: The core number, though derived from degree, is a better indicator of the capacity for information dissemination. Strategically placed nodes, as detected by the *k-core* decomposition are able to spread information to a larger portion of the graph. This result has been shown by several works such as [9, 92, 137]. In [39, 145, 175], the authors applied *k-truss* decomposition which is a triangle based extension of *k-core* decomposition, with the objective of finding a refined set of influential nodes from all the potential high core nodes. The authors show that k-truss decomposition extracts influential spreaders which can infect a large portion of target nodes within first few steps of the SIR epidemic model. We show in this thesis that in some class of networks, coreness based spreading method does not work well and it is comparable to naïve random seed based method. On the other hand, for a special class of networks, the coreness based spreading is very effective. There exist other significant applications of core periphery structure for which we refer to the comprehensive tutorial [115].

2.4 Temporal networks

Centrality in time-varying networks: Time varying networks have been a topic of huge interest largely because in contrast to static networks, they take into account the time frame

of pairwise interactions, which explains the dynamics of roles undertaken by nodes in the network. We refer to comprehensive reviews [78, 79] for detailed coverage on the topic. One of the key directions of research in temporal networks has been identifying influential agents in a dynamic setting. In this lines, there have been two types of research – the first type have attempted to adapt definitions of traditional centrality metrics to the dynamic setting and the second type have attempted to predict centrality of nodes from previous versions of the network, without explicitly calculating the metrics for the current version.

The inherent dependence of centrality metrics on the network structure has been further studied by Braha et al. [19, 20, 75]. Their seminal work throws insights on the dynamical behavior, including a dramatic time dependence of the role of nodes apparent in the different snapshots of the network. These characteristics are not apparent in static (time aggregated) analysis of node connectivity and network topology thus motivating the necessity to have separate definitions of centrality in temporal networks. In [88, 133], the authors utilize a powerful framework of time ordered graphs which transforms dynamic networks to static networks with directed flows. The authors adapt definitions of centrality such as closeness, betweenness and degree for temporal networks, by defining temporal geodesic distances on time ordered graphs. However, the authors assume the knowledge of all the nodes in the combined time steps beforehand which is not generally possible in dynamic networks, where both the edge and the node may appear or disappear in subsequent time steps. Further the time complexity for calculating temporal paths is comparable to static case, hence no real gain in terms of efficiency is achieved. Lerman et al. [98] propose a dynamic centrality, where they show that ranking of nodes obtained by their method varies considerably from traditional approach of detecting important nodes by aggregating temporal networks as static graph. There are also separate lines of work [112, 147, 168, 169] that adapt the definition of eigenvector centrality and PageRank for dynamic networks; however, they require iterative parameter estimation and are not suitable for streaming setting. In contrast, to existing methods, in this thesis we propose to predict central nodes in future temporal time point by virtue of studying activity in past timesteps.

Social contact patterns: In [89] the authors showed that temporal human activity networks have periodic patterns and further devise prediction functions to calculate centrality of nodes at future time steps based on past history. Yang et al. [181] developed a method, that combines concepts of preferential attachment and triadic closure to capture a

node's prominence profile. They further showed that the proposed node prominence profile method is an effective predictor of degree centrality. In [186, 187] the authors observed that a node's temporal social contact patterns show strong correlation with its centrality. Based on this property they predicted closeness centrality and applied their prediction to improve data forwarding in opportunistic mobile networks. One of the prime drawback of these previous works is in the datasets, which have been primarily restricted to human contact networks. To the best of our knowledge ours is the first work which applies centrality prediction on diverse real world and synthetic networks. We also use time series prediction models, which have not been employed for this task previously. These key nodes play important role in information diffusion processes and epidemiology, due to their strategic position in the network. In case of static networks, influential nodes are estimated in terms of centrality metrics. The authors in [66] proposed a generalization of betweenness centrality in dynamic frameworks using the idea of temporal shortest path. The authors showed difference between traditional betweenness centrality and temporal betweenness centrality. On similar lines [88, 166, 168] proposed generalization of other node level centrality metrics in dynamic setting and motivated the need behind this objective.

Time series forecasting: Time series modeling have found a lot of application in econometric analysis and financial forecasting [47, 52, 68]. State of the art time series models such as ARIMA model and its variants [85, 120, 121, 184] have been applied in several real world problems yeilding fruitful results. Real world problems such as traffic forcasting [64, 106] and weather forcasting [163] have been tackled using statistical time series models. Hybrid methods have also been devised to learn models in online settings [107, 183]. In the context of temporal networks major works incorporating time series formulation includes [72, 81, 149]. In [161], the authors leveraged time series forecasting models to predict global properties of the network at a future time step. Although our prediction scheme is based on a similar foundation, to the best of our knowledge ours is the first attempt toward identifying the top central nodes in a network.

2.5 Network representation learning

In this section we provide a brief overview of network representation learning. We first cover algorithmic techniques for unsupervised network representation learning. We then describe supervised approaches, i.e., graph convolution neural networks as well as some of its major variants.

2.5.1 Unsupervised network representation learning

Recently there has been a surge of research toward developing node embedding methods, where the goal is to encode nodes as low-dimensional vectors. These vectors summarize information regarding node position in the graph and local neighborhood. Such representations can also be considered projecting nodes into a latent space, where geometric relations between vectors in this latent space, generally measured using dot product, correspond to proximity based similarity in the original graph [25].

The basic framework on which several algorithms works on, learns an encoder function $f(u) \rightarrow \mathbb{R}^d$ which projects a node u into low d dimensional vector. Similarly a decoder function $g(f(u), f(v)) \rightarrow \mathbb{R}^+$ is trained that maps pairs of node embeddings to a real-valued node similarity measure. The similarity which the decoder represents is user defined graph statistics such as existence of edges between nodes, presence of common neighbors etc. Given that we have a precomputed structural similarity between two nodes given by $\mathcal{G}(u, v)$, the encoder-decoder architecture is optimized such that $g(f(u), f(v)) \sim \mathcal{G}(u, v)$. Most approaches learn parameters of the encoder-decoder framework by minimising the reconstruction loss function $l(g(f(u), f(v)), \mathcal{G}(u, v))$ over pair of nodes in the training dataset. Once we have optimized the encoder-decoder system, we can use the trained encoder to generate embeddings for nodes, which can then be used as a feature inputs for downstream machine learning tasks. For example, one could feed the learned embeddings to a logistic regression classifier to predict the community that a node belongs to [139], or one could use distances between the embeddings to recommend friendship links in a social network [63].

Many recent successful methods that follow the above framework learn the node embeddings based on random walk statistics. The key idea is optimizing the node embeddings so that nodes end up having similar embeddings if they tend to co-occur on short random walks over the graph. Thus, instead of using a deterministic measure of node similarity, these random walk methods employ a flexible, stochastic measure of node similarity, which has led to superior performance in a number of settings. One of the pioneering works in this direction was proposed by Perozzi et al. [139] as DeepWalk. Here author extended a neural language model approach, i.e., skipgram for constructing the node embedding. Sikpgram [124] aim to maximize the co-occurrence probability among the words that appear within a predefined window size. DeepWalk [139] first samples a set of walks from each node of the input graph using truncated random walk. The basic approach is to uniformly sample a neighbour of the last visited node until the maximum length is reached. Each path sampled from the graph corresponds to a sentence from the corpus, where a node corresponds to a word. Following the exploration step skipgram is applied on the walks to maximize the probability of observing a node's neighbourhood conditioned on its embedding. In this way, nodes with similar neighbourhoods, i.e., having large second-order proximity values end up sharing similar embedding. DeepWalk optimizes the following objective function

$$\mathcal{L} = \sum_{u,v \in \mathcal{D}} -\log g(f(u), f(v))$$

where $g(.)$ is represented as softmax function

$$g(f(u), f(v)) = \frac{\exp^{f(u)^T * f(v)}}{\sum_{\forall k \neq v \in V} f(k)^T * f(v)}$$

The softmax function mimics the conditional probability $p(u|v)$, i.e., node u co-occurring on the random walk path starting from v.

Similar to DeepWalk [139], node2vec [63] preserves higher order proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks. The crucial difference from DeepWalk is that node2vec employs biased-random walks, that provide a trade-off between breadth-first (BFS) and depth-first (DFS) graph

searches, and hence produces higher-quality and more informative embeddings than DeepWalk. Choosing the right balance enables node2vec to preserve community structure as well as structural equivalence between nodes.

DeepWalk and node2vec initialize the node embeddings randomly for training the models. As their objective function is non-convex, such initializations can be stuck in local optima. HARP [31] introduces a strategy to improve the solution and avoid local optima by better weight initialization. To this purpose, HARP creates hierarchy of nodes by aggregating nodes in the previous layer of hierarchy using graph coarsening. It then generates embedding of the coarsest graph and initializes the node embeddings of the refined graph (one up in the hierarchy) with the learned embedding. It propagates such embeddings through the hierarchy to obtain the embeddings of the original graph. Thus HARP can be used in conjunction with random walk based methods like DeepWalk and node2vec to obtain better solutions to the optimization function.

One of the limitations of the above approaches is that they donot scale to very large networks. Tang et al. proposed an efficient approach LINE [167] which scales to arbitarily large network. More precisely LINE explicitly defines two functions; one for first order proximity and another for second order proximity. In the experiments conducted, second order proximity performed significantly better than first, and it was implied that including higher orders may level off the improvements in accuracy. LINE defines two joint probability distributions for each pair of nodes then minimizes the KL divergence of the distributions. The two distributions are the adjacency matrix and the dot product of node embedding. A departure from the former approach has been taken in [26, 179], where authors learn low dimensional global representation for nodes using matrix factorization based techniques.

2.5.2 Semi supervised network representation learning

Deep learning techniques on graphs has seen remarkable progress in the last few years. This success is attributed to similar success in image recognition tasks through application of convolutional neural networks (CNN) [87]. However CNN's cannot be applied directly to graph data since they do not have regular grid like structure. Graph convolutional neural

network (GCN) solves this problem by learning spatially invariant, parameterized filter on graph data.

The key mathematical techniques essential to solve these objectives is borrowed from the rich literature of graph spectral analysis [36]. Spectral graph theory proposes graph Laplacian matrix which is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}_{ij} = \sum_j \mathbf{A}_{ij}$. Graph Laplacian is further normalised as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. One of the amenable properties of $\mathbf{L}$ is that it is positive definite, hence can be diagonalised using singular value decomposition such that $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{\mathbf{T}}$. Here $\mathbf{U}$ denotes a space of orthonormal eigenvectors corresponding to Λ, i.e., eigenvalues of the graph Laplacian. Given $\mathbf{x} \rightarrow \mathbb{R}^n$, where x_i indicates a scalar signal for the i^{th} node in the network, graph Fourier transform [160], translates signal $\mathbf{x}$ into frequency domain using matrix multiplication with $\mathbf{U}^T\mathbf{x}$. $\mathbf{U}$ is interpreted as the Fourier basis, where Λ is interpreted as frequencies of the graph signal. Bresnon et al. [44] proposed parameterised local filters denoted by g_θ such that $\hat{\mathbf{x}} = g_\theta(L)\mathbf{x}$. This convolution operation can be further expressed as

$$\hat{\mathbf{x}} = g_\theta(\mathbf{U}\Lambda\mathbf{U}^{\mathbf{T}})\mathbf{x} = \mathbf{U}g_\theta(\Lambda)\mathbf{U}^{\mathbf{T}}\mathbf{x}$$

The authors suggested that explicit computation of the former operation is computationally expensive. This is primarily because eigenvalue extraction has $O(N^3)$ complexity [54]. The authors proposed a solution to the computational burden by expressing $g_\theta(L)$ as recursively computed Chebyshev polynomial inspired by [70, 73].

Kipf et al. [91] showed that *K localised* spectral graph convolution approach proposed by [44] suffers from overfitting in cases of networks with large neighborhood sizes. This is prevalent in real world networks emerging from social, information, biological domains [100] which have wide degree distributions. They proposed an efficient version of GCN by constraining parameters as well as the Chebychev expansion. The final convolved signal matrix is represented as $\mathbf{Z} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\mathbf{X}\boldsymbol{\theta}$ where $\tilde{A} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}_{ij}} = \sum_j \tilde{\mathbf{A}}_{ij}$. This is also called *renormalization trick*, applied to alleviate vanishing/exploding gradient in case of deep networks.

Wu et al. [177] showed that non-linear activation function GCN [91] borrowed from traditional deep learning machinery [71] adds unnecessary complexity, which can be removed

by removing non linearity and collapsing stacked layers. They empirically validated that in some datasets superior results can be obtained compared to other complex architectures. The authors in [177] further theoretically justified that their convolution operation manifests as low-pass-type-filter which captures low-frequency signals and corresponds to smoothing features across a graph.

Attention mechanism has become a cornerstone in state-of-the-art performance of sequence based tasks [105, 148, 173]. One of the benefits of attention mechanism is that they add interpretability to the final results. Velickovic et al. [174] proposed GAT which can be considered as the first attention based architecture for node classification task on graph data. This architecture is a special case of MoNet proposed by Monti et al [127].

One of the common themes in most of the previously discussed methods is that they apply spectral approach for extending CNN architecture over graphs. Hamilton et al. [69] introduced GraphSAGE, which forsakes this direction of exploration and proposed an inductive approach. This technique operates by sampling a fixed-size neighborhood of each node, and then performing a specific aggregator over it (such as the mean over all the sampled neighbor's feature vectors, or the result of feeding them through a recurrent neural network). Chen et al. [32] proposes a similar approach but with more principled sampling methodology which saves gradient computation time. [142,185] are other notable non spectral architectures which borrow from statistical relational learning and Bayesian methods respectively.

Emergence of GCN have opened up several potential applications in various domains ranging from natural language processing, knowledge graphs etc. Yao et al. [182] formulated a heterogeneous graph from a corpus such that nodes are documents and words; edges among nodes are word occurrence in documents (document-word edges) and word co-occurrence in the whole corpus (word-word edges). They showed that existing architecture [91] of GCN yields superior document representation even with naïve one-hot initialisation. In similar lines Vashisth et al. [172] applied GCN towards improving word representation by using novel syntactic dependency graphs from sentences. [24, 135, 150] showed utility of GCN architecture in knowledge graphs and solved pertitent problems such as link prediction, important node prediction, community detection. In this thesis we modify GCN architecure [91] to leverage graph topology and develop a method for detecting central

nodes similar to existing works such as [103, 135, 141].

Chapter 3

Rich centrality clubs in networks

3.1 Introduction

Network resilience measures how well certain properties of a network are maintained under attacks to the system. To date, most research has focused on the robustness of a network, that is how attacks can disconnect the network [6, 29, 30]. In this thesis, we study network resilience with respect to path-based centralities, specifically betweenness and closeness centralities. We term this type of resilience as *centrality resilience*, as opposed to the connectivity resilience in earlier studies. Centrality resilience is more difficult to detect than connectivity resilience. When one part of a network cannot communicate with the rest of the system, it is easy to infer that the cause is due to disconnectivity. Attack on centrality, however, may not disconnect the network, but result in longer distances when traversing the network. The increased length of the distances, is due to the change in the ranking of the high centrality vertices which may not be immediately apparent until the centralities of the system are re-computed. Therefore, perturbing centrality resilience is a very powerful approach for insidiously disrupting the functioning of a system, without a drastic change to its structure. Such techniques can be applied for attacks that are typically malicious (stealth attacks in cybersecurity), where the location of attack cannot be immediately known. They can also be benign such as vaccination under limited resources, for systems where complete disconnection is not possible.

3.1.1 Structural properties that affect centrality resilience

In this thesis we show that core periphery structure of networks play significant role in governing centrality resilience. The core number of a node is the highest value k such that the node is a part of a subgraph (*k-core*), with every node having atleast k neighbors. The assignment of core numbers to nodes can be computed in $O(|E|)$ ($|E|$= # edges) complexity using *k-core* decomposition [12]. To find a *k-core*, Batagelj et al. [12] recursively removes nodes with degree less than k iteratively starting with pendant vertices. We posit that since the inner cores of the network are dense and have a hierarchical structure, path based high central nodes, i.e., high closeness and betweenness nodes should be localised within the inner cores. These high central nodes form a rich club within the inner cores which we term as *rich centrality clubs* (RCC). In this chapter of the thesis, we empirically show this property in several real world and synthetic scale free networks. We further demonstrate certain desirable properties in networks with RCC. We show these rich clubs can be used as potential seed nodes for community detection as well as effective propagators of information in the network. We develop novel attack models and show that networks with RCC are robust against random and targeted edge perturbation. Our results also demonstrate that not all networks have this property, hence we develop novel techniques to extract rich clubs using second order dependencies. Given these favorable properties, we posit that, in many cases, the presence of RCC is desirable. To this end, we propose a modification model that can form a RCC in a network where it is absent. Our model is such that other key properties of the original network including the power law exponent, the average degree, shortest path based centralities and clustering coefficient remain unchanged. Finally we give theoretical justification of our empirical results.

Network suite

The real-world networks that we use for our experiments are available publicly at the following resources [95,99]. Given below is a brief description of these networks. A summary of their properties is noted in Table 3.1 and 3.2. The α is the scale free exponent obtained after fitting the empirical degree distribution to a power law distribution which is given by $p(k) \sim k^{-\alpha}$ where $p(k)$ is the fraction of nodes having degree k. All the networks are

considered to be undirected.

- **Autonomous systems networks (AS and Caida)**: The autonomous system is a network of highly connected routers. Each AS exchanges traffic with neighbors (peers) using BGP (Border Gateway Protocol). Here we use two example networks; both were created using BGP table snapshots. The first dataset (AS) was collected from University of Oregon Route Views Project and it contains 733 daily data traces between autonomous systems which span an interval of 785 days from November 8, 1997 to January 2, 2000. The second dataset was collected by Center for Applied Internet Data Analysis (Caida) from January 2004 to November 2007.

- **Bible**: This data is derived from text of King James Version of the Bible and captures the relation between nouns (places and names). Each time two names occurred in the same verse, a connection was created between them to create a co-occurrence network of people and places.

- **Software**: This graph is formed by analyzing the Apache Ant library which is a tool for developers to build Java projects from the source code to the executables. The DependencyFinder tool was used to extract the class hierarchy and inter class dependency. Each vertex represented a class and two classes were connected by an edge if one class depended on another. Although the relation is directed, for our experiments we consider the network to be undirected.

- **Power**: This network contains information about the power grid of the Western States of the United States of America. An edge represents a power supply line. A node is either a generator, a transformer or a substation.

- **Facebook**: This network is composed of Facebook users and the post they have made to each other's walls. Here nodes represent a user and an edge between u and v indicates a post has been made by at least one of the users (u or v) to the other's wall.

- **Protein**: This is a network of protein-protein interactions in yeast. A node represents a protein and an edge represents a metabolic interaction between two proteins.

Network	Nodes	Edges	α	$\mu(d_v)$	$\mu(C_lC)$	$\mu(BC)$	*LCN*
AS [99]	6474	13895	1.235	4.29	0.27	0.004	12
Caida [99]	16493	33372	1.17	4.04	0.27	0.001	20
Bible [95]	1707	9059	1.523	10.61	0.31	0.001	15
Software [95]	994	4645	1.168	9.32	0.34	0.002	11
Protein [95]	1458	1993	2.106	2.73	0.15	0.004	5
Facebook [99]	7178	10298	2.896	2.86	0.11	0.001	5
Hepth [99]	2694	4255	1.487	3.15	0.18	0.001	7
Power [95]	4941	6594	2.845	2.66	0.05	0.003	5

Table 3.1: Test suite of real world networks and their properties. α: power-law exponent, $\mu(d_v)$: average degree, $\mu(C_lC)$: average clustering co-efficient, $\mu(BC)$: average betweenness centrality. (LCN): largest core number in the network.

- **Hepth**: This is a collaboration network from the e-print arXiv and covers scientific collaborations between authors and papers in High Energy Physics. If an author i co-authored a paper with author j, the graph contains an edge connecting i to j. If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes. The data covers papers in the period from January 2001 to April 2003.

- **Synthetic network**: We also generate as many as 16 synthetic networks of varying sizes and varying network core structures using the MUSKETEER synthetic network generation tool discussed in [65]. We refer to these networks as N1 through N16 in the rest of this chapter (see Table 3.2).

Network	Nodes	Edges	α	$\mu(d_v)$	$\mu(C_lC)$	$\mu(BC)$	*LCN*
N1	14212	34901	1.215	16.3	0.25	0.0007	16
N2	10162	25154	1.28	4.95	0.27	0.0008	14
N3	36469	96990	1.237	2.66	0.24	0.003	27
N4	65630	170061	2.29	2.66	0.13	0.003	12
N5	4091	33352	1.438	2.66	0.33	0.003	21
N6	6785	44381	1.421	2.66	0.31	0.003	19
N7	6009	13585	2.016	2.66	0.18	0.003	6
N8	3863	22356	1.268	2.66	0.34	0.003	23
N9	11278	19616	2.737	3.47	0.03	0.003	5
N10	19623	33711	2.832	3.43	0.05	0.00049	5
N11	5980	9501	3.027	3.17	0.05	0.05	6
N12	6045	13592	2.373	4.49	0.11	0.0035	7
N13	7783	35185	2.517	3.61	0.06	0.0033	6
N14	15988	28373	3.029	3.54	0.09	0.004	5
N15	28651	51159	3.073	3.57	0.04	0.0028	6

Table 3.2: Test suite of synthetic networks and their properties generated using [65]. α: power-law exponent, $\mu(d_v)$: average degree, $\mu(C_lC)$: average clustering co-efficient, $\mu(BC)$: average betweenness centrality. (LCN): largest core number in the network.

3.2 Motivating experiments

We present the experiments that motivated our research. We test whether vertices with high core numbers **(i)** have high centralities and **(ii)** can be used as seed nodes for community detection.

3.2.1 Correlation with other centrality metrics

Several papers [37, 77, 104, 122, 156, 162], claim that the vertices with high core numbers should also have high centrality values. To test this claim, we compute the Jaccard coefficient (J_c) given by $\frac{S_1 \cap S_2}{S_1 \cup S_2}$ between the set of vertices with highest core numbers (S_1) and an equal number of high ranked nodes for each of the centrality metrics (S_2).

The results in Table 3.3 show a clear separation of the networks. In the first group, all the networks (blue) have high J_c implying significant number of high central nodes also have highest core numbers. In the second group, (brown) the high core numbered nodes do not have high centrality as per the low J_c scores.

Network	degree	closeness	betweenness
AS	0.6	0.75	0.5
Caida	0.56	0.69	0.49
Bible	0.49	0.64	0.43
Software	0.40	0.5	0.23
Protein	0	0	0
Facebook	0.10	0	0
Power	0	0	0
Hepth	0.05	0.05	0.05
N1	0.63	0.86	0.68
N2	0.722	0.530	0.653
N3	0.74	0.78	0.80
N4	0.68	0.81	0.68
N5	0.80	0.82	0.78
N6	0.57	0.71	0.6
N7	0.39	0.48	0.37
N8	0.79	0.79	0.79
N9	0	0	0
N10	0	0	0
N11	0	0	0
N12	0.02	0	0
N13	0.06	0	0
N14	0	0	0
N15	0.002	0	0

Table 3.3: Jaccard index between nodes with highest coreness and equal number of high centrality nodes. Results clearly separate the two categories of networks into ones that have an RCC (blue) and ones that do not have an RCC (brown).

3.2.2 High core numbers to detect communities

In existing techniques of community detection utilizing *k-core* structure [138, 176], the network is reduced to its *k-core* subgraph, for a predetermined value of k, and the communities in the subgraph are computed. The vertices in these communities are used as seed nodes to propagate the community information to other vertices.

We note that this process will succeed only if the communities are well represented in the reduced network. Therefore to test the applicability of such algorithms, we tabulate how the vertices in the innermost core are distributed across the communities. Figure 3.1, 3.2 plot the community ids of the networks in the x-axis and the number of nodes from the innermost core that are members of a particular community in the y-axis. The communities are ordered in decreasing order of size and we use the Louvain method [15], to find communities. We include all communities whose at least one vertex is in the innermost core. We again observe a separation between two class of networks. In one group (see Figure3.1), the vertices from the innermost core are spread over multiple communities, whereas in the other group (see Figure 3.2), the vertices from the innermost core are concentrated in one or two, communities. Clearly, the first group is more suitable for the community detection algorithm described earlier. Figure 3.3 shows the community distribution in the innermost core of two networks from our test suite. These results demonstrate that *vertices with high core numbers are not always distributed across multiple communities, and in some cases can be concentrated in only one community.*

3.2.3 Evidence of rich centrality club

These motivating experiments demonstrate the existence of two groups of networks. One group consists of networks where the vertices from the inner cores have high correlation with vertices of different high centrality metrics and can be used as seed nodes for community detection. The other group consists of networks where the vertices from the inner cores have no correlation with other high centrality metrics, and are concentrated in one or two communities.

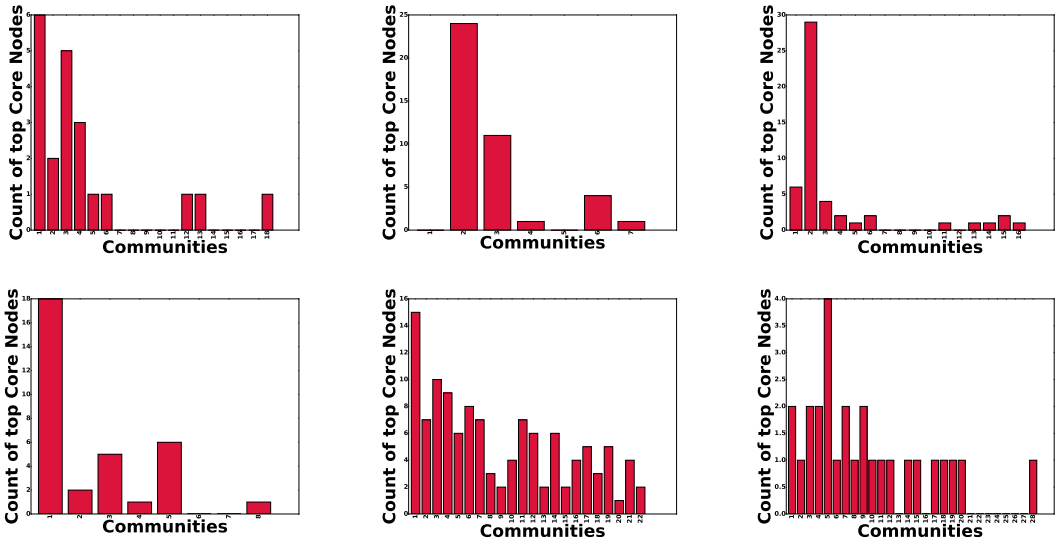


Figure 3.1: Distribution of innermost core nodes in the different communities of the networks with RCC. The X-axis indicates the community ids of a network ordered in terms of number of nodes present in that community. Y-axis indicates the number of innermost core nodes in a particular community with the id on the x-axis. The X-axis stretches includes all communities that contain the nodes from the innermost core.

To visualize how rich centrality clubs are formed in the innermost cores, we divide the vertices in the network into two sets of nodes, based on whether they are part of the innermost core or not. We partition the nodes outside the innermost core into their respective communities and combine each community into a single supervertex. This process of graph summarization is similar to previous work of Koutra et al and others [109, 110, 164, 180]. The set of nodes of the innermost core are also combined into a supervertex. Two supervertices are connected if there is at least one edge between the nodes comprising them.

Figure 3.4 shows a visualization of the reduced network for two benchmark networks. Each node is labeled as c_x for communities and k for the innermost core. The supervertices are ordered by size with respect to average centrality (closeness and betweenness) of constituent vertices. For the network Caida, which is in the first group, the supervertex corresponding to the innermost core has significantly high centrality and is in the centre. For the network, Power, which was in the second group, there are no distinctively high centrality supervertex.

Since in the first group, the high centrality nodes are in the innermost cores and since by

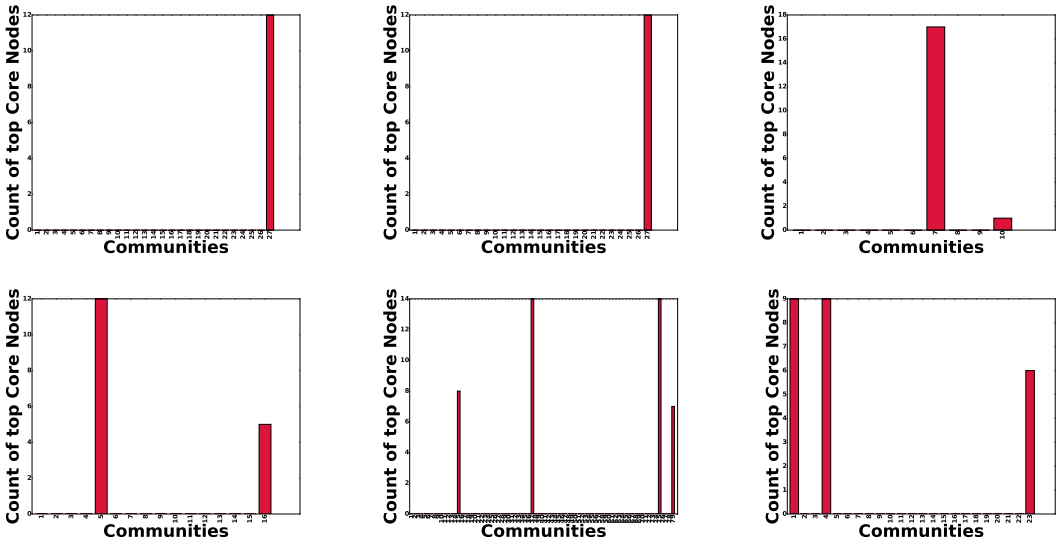


Figure 3.2: Distribution of innermost core nodes in the different communities of the network without RCC. The X-axis indicates the community ids of a network ordered in terms of number of nodes present in that community. Y-axis indicates the number of innermost core nodes in a particular community with the id on the x-axis. The X-axis stretches includes all communities that contain the nodes from the innermost core.

their definition these vertices are connected to each other, our experiments demonstrate that *rich club of high centrality vertices* is formed in these networks. In the next section, we present the topological property of these networks that lead to the formation of RCC.

To qualitatively see that the innermost core contains the high centrality vertices, we rank each supervertex based on its centrality score. If the supervertex corresponding to the innermost core is within the top 10 ranked supervertices, we report its reciprocal rank (1/rank) in Table 3.4; otherwise, if it is not within the top ten, we report zero. We observe a distinct grouping of our test suite of networks.

Clearly, the networks where the reciprocal rank of the supevertex is central, i.e. the reciprocal rank is almost always one, comprises of the set of nodes forming the rich centrality club.

We observe that in one group of networks in our test suite, the reciprocal rank of the supevertex corresponding to the innermost core is almost always one, indicating that this supervertex is indeed the most central.

Network	Closeness	Betweenness
AS	1	0.5
Caida	1	1
Bible	1	1
Software	1	1
Protein	0	0
Hepth	0.5	0.33
Power	0	0
Facebook	0.5	0.5
N1	1	1
N2	1	1
N3	1	1
N4	1	1
N5	1	1
N6	1	1
N7	0.5	1
N8	1	0.5
N9	0	0
N10	0	0
N11	0.5	0.33
N11	0.33	0.33
N12	0.33	0.5
N13	0.5	0.5
N14	0	0
N15	0	0

Table 3.4: Reciprocal rank of the supervertex corresponding to the innermost core of the network. The ranking in done both based on closeness (second column) as well as betweenness (third column) centrality.

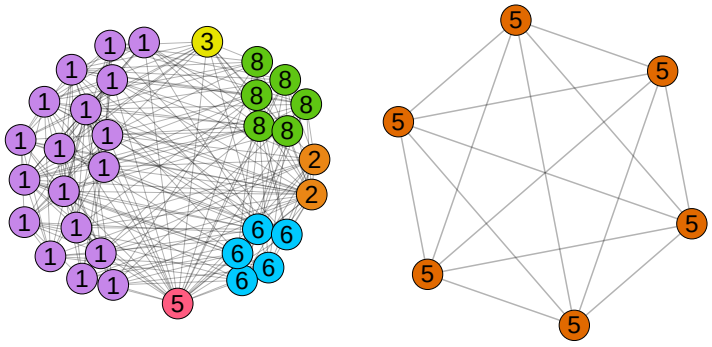


Figure 3.3: Visualization of the subgraph formed using the innermost core nodes. Here nodes belonging to the same community are annotated with the same color and id. In the software network the innermost core clearly has nodes from several communities. In the protein network all nodes in the innermost core belong to the same community.

3.2.4 Formal definition and rationale

Let the subgraph induced by the vertices in shell k and their neighbors be S_k. Let d_k be the average degree and n_k, the number of nodes in S_k. Let λ_k be the second smallest eigenvalue of the normalized Laplacian matrix of S_k. Let the average distance of a vertex in shell S_a to a vertex in inner shell C_b, $a < b$, be r_{ab}.

Given these parameters, we state that a network will contain a RCC if the following properties hold.

1. If for two shells $k1$ and $k2$, $k1 < k2$, then $d_{k1} < d_{k2}$ and $n_{k1} > n_{k2}$.
2. For all shells S_k, $\lambda_k > \alpha$
3. For all shells S_k, $r_{kx} < \beta$, where C_x is a high numbered core, with density close to 1.

The first property requires that the shells have progressively smaller number of vertices, and become more dense from outer to inner shells. The second property provides the upper bound of the second smallest eigenvalue, and in turn, to the Cheeger constant at each shell.

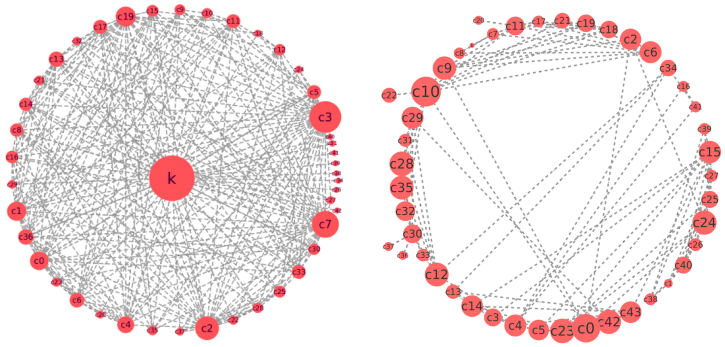


Figure 3.4: Network formed by two category of supervertices, i.e, communities (denoted by c_1, c_2, ...) and the innermost core (denoted by k). Two supervertices are connected if the corresponding nodes from which they are formed are connected by at least an edge. Higher size of supervertex imply higher average centrality of constituent vertices.

The higher λ_k, the more expander-like the associated shell. If the shell has multiple components, then each of them should maintain this property. The third property states that the hops to travel from the outer shells to inner cores should be small.

The values of the parameters α and β are determined based on size and density of the whole network. As per our experiments, setting $\alpha > 0.5$ and $\beta < 4$ can clearly distinguish between networks that contain RCC and those that do not.

3.2.5 Density of shells

The first condition is a feature of the core-periphery structure of almost all scale-free networks, whether they contain RCC or not. To demonstrate this, we first subdivide the vertices into subgraphs S_k. For each, we compute the average degree and the number of nodes. Since the number of shells varies across networks, for uniform presentation of the results we divide our results into three buckets. Starting from innermost we place the first 25% shells in the first bucket (k_4). The next 25% falls in the second bucket (k_2) and the final 50% falls in the last bucket (k_n). For each bucket we calculate the mean and the standard deviation of the average degrees and number of nodes classified in that bucket. These values are plotted in Figure 3.5(a) (average degree) and Figure 3.5(b) (number of nodes). As seen from the figure with the exception of slight deviations, the first property is maintained

in both sets of networks.

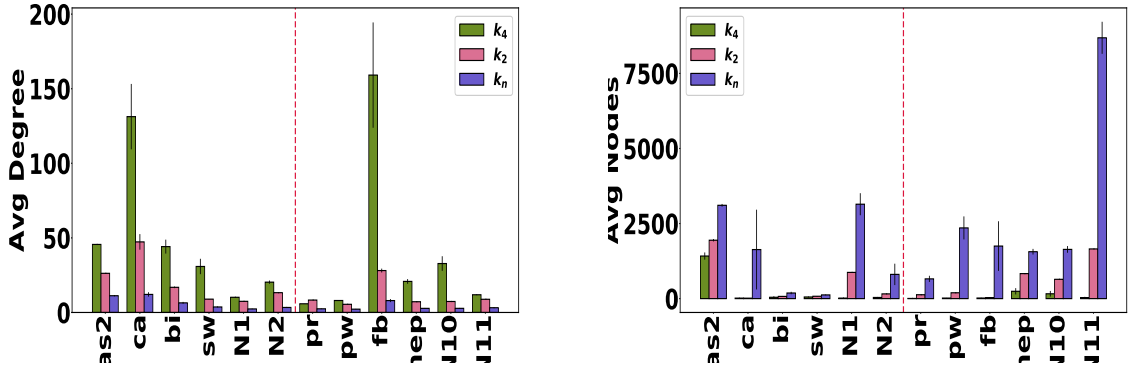


Figure 3.5: (a) The average degree of, and (b) the number of nodes in, the shell based subgraphs for different buckets of shells for each network.

3.2.6 Eigenvalue of shells

For each S_k, we compute the normalized Laplacian, extract its spectrum using eigenvalue decomposition, and compute the eigengap. Since the number of shells varies across networks, for uniform presentation of the results we divide our results into three buckets. Starting from innermost we place the first 25% shells in the first bucket. The next 25% falls in the second bucket and the final 50% falls in the last bucket. For each bucket as defined, we calculate the average eigengap and the standard deviation. These values are plotted in Figure 3.6.

We observe that in graphs where we assume that RCC exists, there is a slow decline of the average eigengap. The Cheeger constant is high in the inner shells and gradually decreases from the inner to the outer shells. In the other group of networks, there is an abrupt fall in the average eigengap after the first bucket of inner shells. The first group can be bound by a large α than the second group, thus corroborating the second property.

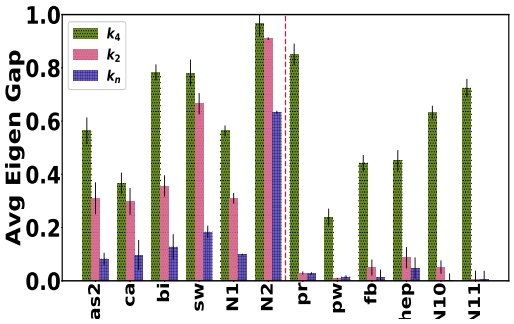


Figure 3.6: Average eigengap of the shell based subgraphs for different buckets of shells for each network. Results show graceful degradation for the networks with an RCC while an abrupt fall for the networks with no RCC.

3.2.7 Distance between shells

The third property enforces that on average two vertices in outer shells are more likely to be connected through inner dense shells. We show this in Figure 3.7, where in networks with an RCC, the average shortest distance of the nodes in the outer shells to the innermost (k_{max}) and the second innermost shells (k_{max-1}) is low compared to networks without an RCC. Considering that a large fraction of nodes lie in outer shells and nodes residing in outer shells are closer to nodes in the inner dense shells than the other nodes in the nearby outer shells, most of the shortest paths should pass through inner dense shells.

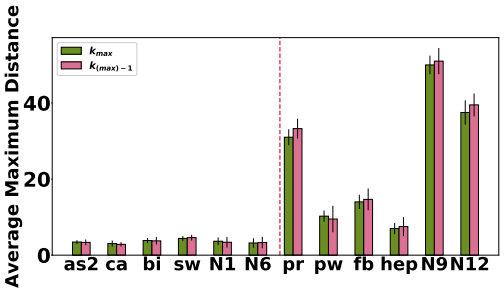


Figure 3.7: The average shortest distance of a node in the outer shells to a node in the innermost (k_{max}) and the second innermost shell (k_{max-1}).

3.2.8 Experimental observations

We see from the experiments that networks with RCC have dense innermost core. Furthermore, several communities emanate from the innermost core. These communities are spread across multiple shells and as per the definition of a community, vertices within a community would be tightly connected. Thus every shell in a network with RCC has several tightly connected regions. We also observe, that due to their definition, boundary of the shells increase as we go from inner to outer shells. Taking these observations together we hypothesize that *the networks with RCC are expander-like across their shells.*

Finally, a majority of the shortest paths pass through the innermost core. However, the shortest paths between two vertices in the same community are likely to be within the community. Therefore, in *the networks with RCC most of the paths in between the communities pass through the innermost core.* To test our hypothesis we conduct two experiments. First, we compute the expansion property as approximated by the eigengap of the networks. Networks with RCC should have high spectral gap to indicate that the Cheeger constants for the subgraphs are also high.

Second, to test the overall 'centrality' of the innermost core, we create a reduced network by combining vertices in the innermost core into a supervertex, and the vertices of each community into other supervertices. For networks with RCC the supervertex consisting of the innermost core should have the most central (betweenness and closeness) node in the reduced network.

3.3 Centrality based rich club coefficient

Degree based rich club coefficient is defined as $\phi(k) = \frac{2*E_{>k}}{N_{>k}*(N_{>k}-1)}$. Here $N_{>k}$ denotes nodes with degree at least k and $E_{>k}$ denotes their corresponding links. Let $E_G(V_1, V_2) = \frac{Cut(V_1,V_2)}{min(Vol(V_1),Vol(V_2))}$ be the minimum number of edges required to disconnect graph G such that $V_1, V_2 \in G$. We argue rich club coefficient covering centrality can be defined as $\Phi(k) = \sum_{i>=k} |E_{g_i} - E_{g_{i+1}}|$ where g_i represent subgraphs formed using the i^{th} shell volume. Higher E_{g_i} for the inner shells indicate that high central nodes have formed a rich

centrality club and low value of $\Phi(k)$ indicates that the rank order is stable against random edge based perturbations. We have empirically validated this in our real and synthetic dataset. Our experiments have shown that networks such as AS, Caida with higher E_{g_i}, lower $\Phi(k)$ have resilient centrality clubs compared to power, Facebook where this property does not hold. However all these networks are structurally almost indistinguishable as far as degree related properties are concerned including a very similar range of scale free exponent α (see Table 3.1, 3.2).

3.4 Beyond single rich clubs

Single rich clubs have been established to be present in the innermost core of the network. We now show that a similar relation also holds for the second class of networks where inner *k-core* subgraph do not contain the major high central nodes. The work of Estrada et al. [50, 51] shows that scale-free real world networks, such as the ones used in our experimental setup, fall in two categories. Either they have a dense core with sparsely connected periphery or the network is composed of modular units which are individually densely connected but have sparse inter module connection. Few related works [42, 93, 94, 178] have shown that dense substructures or multi-cores emerge within these modules. Based on these results, we explore how rich clubs are located within modular units of the network. We term such sub-structures *scattered rich clubs* and extract them using through the following steps. (Figure 3.8 shows an illustrative example).

Step (i) *Finding dense clusters in the network.* We first find dense clusters in the network. Each dense cluster should be sparsely connected to other clusters. To do so, we first partition the network into disjoint communities to obtain modular substructures[1] and then use a clique propagation method [45] to identify dense subgraphs (size ≥ 4) within these communities.

[1] We use the Louvain method [15]. Note that, regardless of the non-determinism of community detection any standard algorithm can be used since our primary goal is not to find exact communities, but to partition the graph into modular substructures.

Step (ii) *Forming the meta network.* Once the set of significant cliques (size ≥ 4) has been found, we analyze their hierarchies using graph summarization methods inspired by [97, 110]. We construct a second order meta network formed by the cliques, where each clique is turned into a supervertex. There are edges between supervertices if the representative cliques are linked to each other.

Step (iii) *Exploring the core periphery structure.* We now perform *k-core* decomposition on this second order meta network to uncover its innermost core, and then observe where the high centrality vertices are located, with respect to the *k-core*s on this meta network.

Results in Table 3.5 show that the networks can be clearly classified into two groups. In the top group (top panel), most of the high centrality vertices are located in the innermost core of the *original network*. These are the networks that have a single rich club. For the bottom group (bottom panel) very few of the high centrality nodes are present in the cores. However, for *all* the networks, if we consider the innermost cores of the *meta network*, most of the high centrality vertices are part of them.

This result demonstrates two important properties, *(i)* most high centrality vertices are indeed present in dense clusters, forming the scattered rich clubs and *(ii)* the scattered rich clubs are connected to each other and form the innermost core of the second order meta network, where the clusters are represented as vertices. In a practical context, the single rich club are formed in networks with a single important centre, such as a hierarchical organization structure with one set of important people at the top. In contrast, scattered rich clubs represent networks with multiple centres of importance. An example, would be a flat organization structure, where each department has its own administrative officers. The second order connections between the scattered rich clubs represent that the administrative officers are connected among themselves, such as through executive committees.

In the previous sections, we observed that in certain real world networks, *inner most k-core* hosts a large fraction of high central nodes. This intrinsic connection between *k-core subgraph* and *high central nodes*, can be leveraged toward building useful applications. In the next section we showcase some of these downstream tasks which can be envisaged, taking advantage of our novel findings.

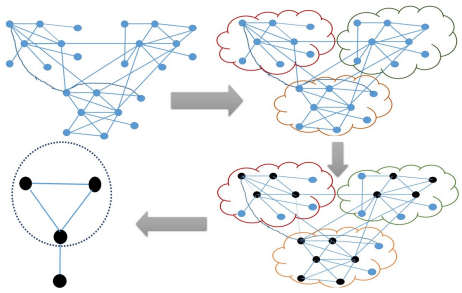


Figure 3.8: Schematic diagram of the meta network generation process. Top left panel is the original network. Community detection method is applied to extract the key modules [15] (top right panel). Clique percolation method is applied on each module to find existing cliques [45] (bottom right panel). Finally meta network is formed such that cliques act as supervertices. Influential nodes exists in the innermost core of the meta network (bottom left panel).

Network	$G(C_{max})$ (%)		$G'(C_{max})$ (%)	
	CC	BC	CC	BC
as2	65	60	85	80
caida	65	60	85	80
bible	65	60	85	80
sw	65	65	90	85
N1	75	70	80	85
N2	70	75	80	85
protein	10	10	95	90
facebook	0	0	95	100
hepth	5	5	85	90
pow	0	0	45	25
N9	0	0	100	85
N10	5	0	80	85

Table 3.5: Percentage of top-20 high central closeness (CC) and betweenness (BC) nodes in the inner most cores of the original network $(G(C_{max}))$ and the second order network $(G'(C_{max}))$. The red rows correspond to networks that have a single rich club while the green rows correspond to networks that have scattered rich clubs. Note that for both types of networks a large percentage of high centrality vertices are in the innermost cores of the second order networks.

3.5 Application

In this section, we demonstrate how the presence of RCC can be leveraged in some important applications.

3.5.1 Attack models

We develop attack models that leverage our knowledge of the single and scattered rich clubs. We empirically demonstrate that attack models based on networks with scattered rich clubs are the most effective. Our attack models are as follows.

1. **Uniform perturbation**: Our first attack model is based on random selection of edges and does not take into account any information about the network structure. Many attack models in literature are based on random attacks, and hence we select it as a baseline. We remove edges from the network uniformly at random. Probability that edge (u, v) is removed is given by $P(u, v) \propto \frac{1}{|E|}$ where $|E|$ is the number of edges in the network.

2. **Core assortative perturbation**: We remove edges if the core number of both the end points is high. This attack is based on the single rich club model, where the high centrality vertices are located in the innermost core. Given an edge (u, v), $P(u, v) \propto c_u * c_v$, where c_u, c_v are the core numbers of the nodes. Thus edges belonging to high core nodes are more likely to be removed.

3. **Rich club assortative perturbation**: This model is based on our observation that high centrality nodes in certain class of networks are located within scattered rich clubs. We remove an edge (u, v) if it is part of a (scattered) rich club obtained in the previous section. The probability that (u, v) is removed is given by $P(u, v) = 0$ if (u, v) are *not* part of a (scattered) rich club. All edges within a scattered rich club have equal probability of getting removed.

Description of experiment: For each network, we first calculate the rank order of the top-20 high central closeness and betweenness nodes. We then apply three different attack

models. For each attack model, we remove from 2% to 8% of the existing edges. We calculate the rank order of the original top-20 nodes in the perturbed version of the network. We calculate the difference between the two rank orders using the Kendall τ metric (τ) which gives a score of 1 if the orders are perfectly aligned and -1 if none of the original orders are retained. For each network we obtain five instances of the perturbed network and report the average τ along with the standard deviation (σ) in Figure 3.9, 3.10, 3.11[2]. Our results are representative and hold for all other networks in the dataset.

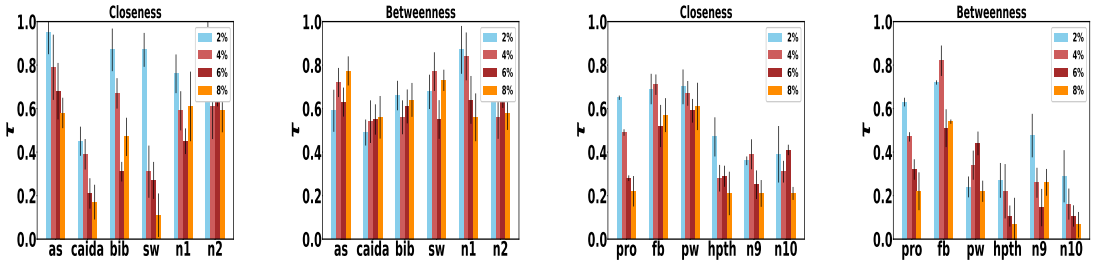


Figure 3.9: Results for application of uniform perturbation based attack on different networks.

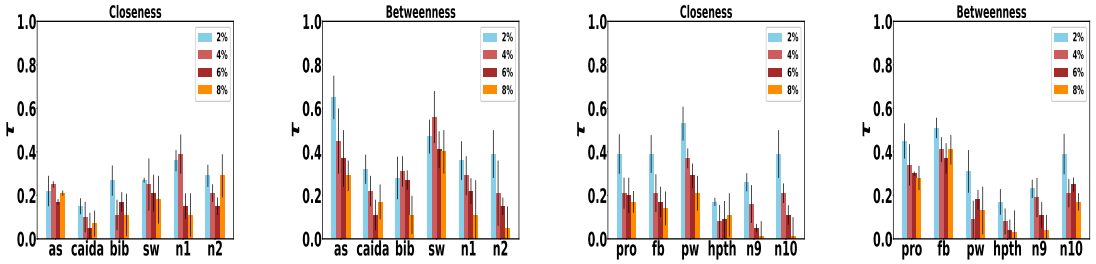


Figure 3.10: Results for application of core assortative perturbation based attack on different networks.

Results: Figure 3.9 shows the results of the random attack. Majority of the networks with single rich club show high τ, i.e., they are less affected by the attack. The networks with scattered rich clubs, are more affected as evident by the low τ score. This is because a random attack will remove edges across the network, and some of them will be part of the scattered rich clubs which are also spread across the network. Figure 3.10 shows the results of the core associative model. Both the networks with single rich club and with scattered

[2]We report here top-20 central nodes. Our experiments with different values of the top m central nodes where $m = 10, 15, 25, 50$ have yielded similar results.

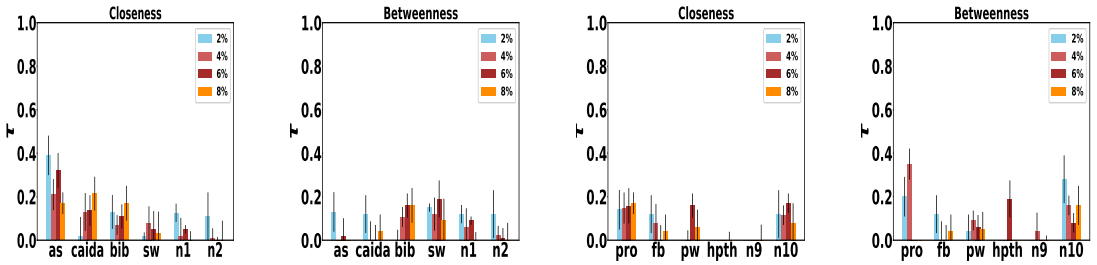


Figure 3.11: Results for application of rich club assortative perturbation based attack on different networks.

rich clubs are affected. However, the networks with scattered rich clubs only show a minor drop in τ when compared to the uniform perturbation. This is because not all vertices of this class of networks are in the innermost core. Figure 3.11 shows the results of the rich club associative model. This attack is the most effective of the three for both classes of networks. This is because the attack is based on the structure of the scattered rich club and so can locate dense cliques with *all* the top-k high centrality vertices. Thus, *the rich club assortative model is the most effective attack model.*

3.5.2 RCC as influential nodes

Vertices, which when selected as seed nodes can accelerate the diffusion process in networks are known as influential nodes. We hypothesize that the RCC, if present in a network, is a natural choice for the seed nodes for spreading. In order to test this hypothesis we execute a diffusion process adapted from [86, 114] on two groups networks in our dataset, i.e., those with an RCC and those without.

We choose a seed set size of S (here we show results for $S = 20$) and populate this set preferentially on the basis of highly connected nodes which includes high centrality nodes (degree, closeness, betweenness), innermost shell nodes (these are nodes from within the RCC in networks that demonstrate its presence) and a random set of nodes. These set of seed nodes initially have an information and they pass it to all their neighbors using a flooding based broadcast approach (all neighbors of an informed vertex get informed).

[2]Our experiments with different value of S yield similar results.

This approach spreads the message very quickly and hence modified versions have been used in peer-to-peer networks for sending queries and searching [84]. Although in real world systems this method is difficult to implement due to scalability issues, our goal here is to study how effective the vertices in the RCC are as seed nodes.

The x-axis in Figure 3.12, 3.13 shows the fraction of vertices that have received the message and the y-axis the steps to reach these fraction of vertices. The networks form two groups. In one group that demonstrate the presence of RCC, the vertices from the innermost core are effective seed nodes for broadcasting and the time is comparable to the time when high centrality vertices are selected as seeds. The other group is when the vertices from the innermost core perform very poorly as seed nodes. The time to spread the information is equal to or worse than a random selection. These results show that *only vertices in the innermost core in networks that demonstrate the presence of RCC are effective as seed nodes for spreading information*.

We observe that networks with RCC have certain desirable properties. In these networks nodes in the *innermost k-core subgraph*, control major information pathways in the network. Therefore they can be used as seed nodes for rapid information dissemination in the network. This determination of seed nodes is computationally inexpensive, compared to traditional methods of explicit central node computation. *Top-k central nodes* in these networks also have stable rankings i.e. random perturbation of the network in terms of edge removal donot significantly alter rankings. This property is useful in infrastructure networks such as router network, where shortest paths are preserved even with minor edge based perturbation so that key nodes retain their ranks and router tables need not be updated frequently. Therefore it would be important to have a mechanism to be able to impose a RCC in a network. In the next section we outline a method which injects rich centrality club in a network, without altering key network properties.

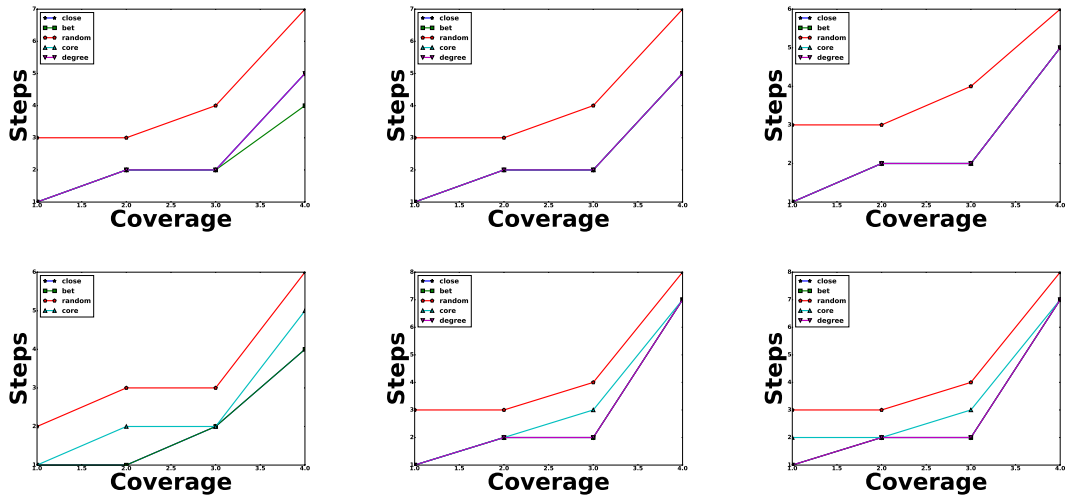


Figure 3.12: Time required in terms of the number of steps for the information to disseminate to $\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n$ nodes in the network. Results are plotted for the networks that demonstrate the presence of RCC, coreness based seed nodes consistently appear to be good choices as message initiators.

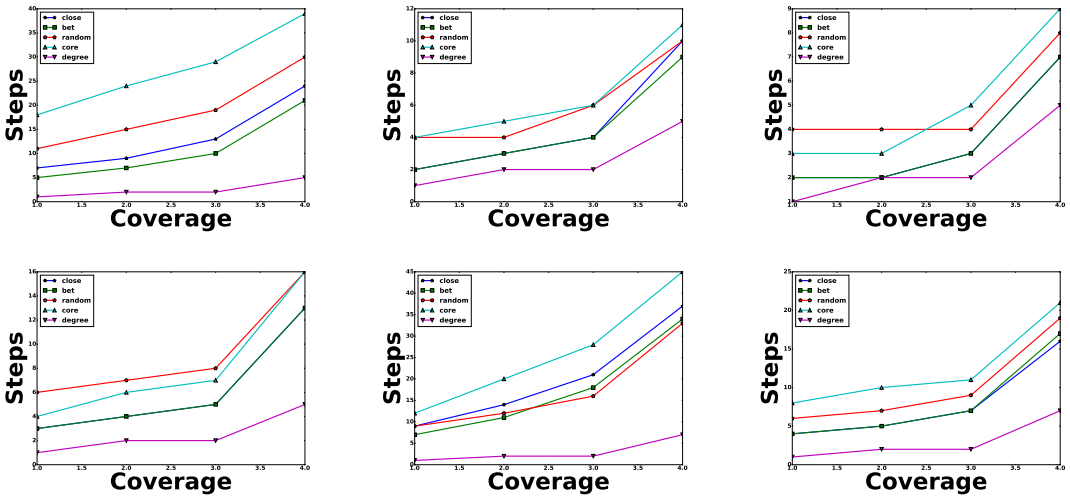


Figure 3.13: Time required in terms of the number of steps for the information to disseminate to $\frac{n}{4}, \frac{n}{2}, \frac{3n}{4}, n$ nodes in the network. Results are plotted for the networks that donot demonstrate the presence of RCC. In such the networks our experiments show that coreness based initiators perform equal to or worse compared to random initiators

3.6 Algorithm to construct RCC

We now present a simple yet effective modification algorithm for inserting RCC into a network and conversely removing RCC from a network containing it.

Rationale for the algorithm: To explain the rationale for our algorithm, we present two simplified models of a network with a RCC and without a RCC (Figure 3.14). If the network contains RCC, then the inner shells are expander like, and communities meet at the RCC. An example model conforming to this structure would be a large clique in the center surrounded by smaller cliques.

In a network without a RCC, the majority of the communities do not meet through the inner core. This indicates that the inner core is not at any special position with respect to the paths connecting the communities. One example model of such a network would be a ring of cliques of different sizes. The smaller cliques can have connections between them. Here the highest core is at the side of the network rather than the center[3].

As per the example figure, to introduce a RCC, we can simply connect the high degree vertices across communities. The high degree of the vertices ensures that the clique (or near clique) formed by them will have higher core numbers. Joining communities ensures that the communities connect within this subgraph. In the network without an RCC in Figure 3.14, we would connect all the vertices in the ring.

Conversely, to destroy the RCC property of the network, we will simply delete the edges in the inner core, such that the connections between the communities are destroyed, and the highest numbered core moves away from the center. Our approach is inspired by existing works in tuning network topology for attack mitigation [111, 151].

Algorithm for forming RCC: Our proposed approach for connecting the communities via high degree vertices is however very expensive. This is because finding communities itself is a computationally intensive operation. A faster alternative is to simply connect (or

[3]We emphasize that these models are only idealized representations of the two types of networks, and more complicated connections occur for real-world networks. Nevertheless the principal idea is maintained, i.e., for networks with a RCC, the innermost core is at the center of the network.

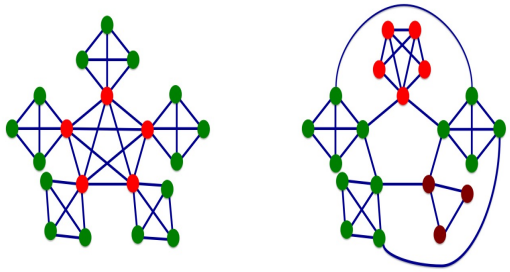


Figure 3.14: Simplified models of a network with (left) and without (right) an RCC. Red vertices have core number 4, green vertices have core number 3 and brown vertices have core number 2. Note that the RCC is formed in the innermost core.

disconnect) connections between the high degree vertices. This method works, because in networks without a RCC, high degree vertices within the same community are likely to be already connected. Therefore, any vertex pair connected as part of the modification algorithm will be in different communities.

Moreover, increasing the connections among the high degree nodes also brings all those (usually low degree) nodes that are neighbors of these high degree nodes closer in the network. Thus, the nodes in the innermost cores will have high centrality, as is a characteristic of networks with a RCC. On the converse side, for a network with a RCC, the high degree vertices will be in the inner core, so removing edges between them disconnects the cores.

It might seem from our approach that rich clubs of high degree vertices are also the rich centrality clubs. Figure 3.14 shows a counter example. The right hand graph has a rich club of high degree vertices but not a dense subgraph of high centrality vertices.

Experiments. The pseudocode of the algorithm is given in Algorithm 1. Figure 3.15, plots the eigengaps of the networks before (blue lines) and after (green lines) the modification. To clearly compare between the original and modified networks, we plot the eigengaps for each shell, rather than over an aggregate of shells as done in Figure 3.6(c). Note that for the networks that demonstrate the presence of RCC (AS, Bible and Software), the eigengap of the modified network is smaller, i.e., the green line is lower and has a steeper slope than the original network. For the networks that do not demonstrate the presence of RCC (Power, Protein and Facebook), the green line is higher, showing that the value of the eigengap

N/W	|V|	|E|	α	$\mu(d_v)$	$\mu(C_lC)$	$\mu(BC)$	*LCN*
As	6474	12439	1.245	4.07	0.26	0.0012	9
Bible	1773	8600	1.557	10.07	0.298	0.006	11
Software	1003	4400	1.236	8.85	0.339	0.004	9
Protein	1870	2052	1.756	2.89	0.17	0.016	7
Facebook	7178	10349	2.311	2.82	0.125	0.0123	6
Power	4941	6698	2.344	2.71	0.0715	0.017	7

Table 3.6: Network statistics for the modified graphs. Note that the parameter values are comparable to that of the original networks in Table 4.1.

increased and has a more gradual slope. We report the statistics of the modified network in Table 3.6. The table clearly shows that our model also preserves the crucial structural properties of the original network, for e.g., the scale-free exponent α and the average degree.[4]

Algorithm 1: Algorithm for increasing $(flag == 1)$/decreasing $(flag == 0)$ expansion property.

Input: $G(V, E)$

Output: E

Parameters: $h, \gamma, flag$

Sort vertices in G based on decreasing degree;
Select the top h nodes based on degree;
if $flag == 1$ **then**
    find E_1 possible edges that could be formed among the h nodes
else
    find E_1 actual edges that are present among the h nodes
Select E_f edges randomly from E_1 where $|E_f| = \gamma * |E_1|$;
if *flag == 1* **then** $E \leftarrow E \cup E_f$ **else** $E \leftarrow E - E_f$ **return** E;

[4]We set the model parameter $h = 30$. The results are similar for $h = 20$ and $h = 15$. γ is set to 0.2

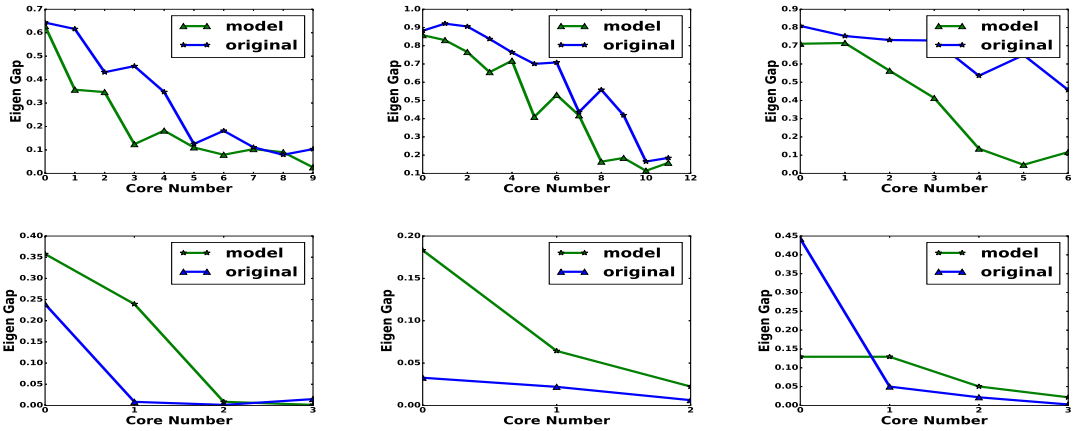


Figure 3.15: The outcome of the modification model. The first three networks (top panel) that originally demonstrate presence of RCC, i.e., AS, Bible and Software get transformed to networks with no RCC. The last three networks (bottom panel) that originally do not demonstrate the presence of RCC, i.e., Power, Protein and Facebook get converted to networks with RCC. These plots are similar to the eigengap chart of Figure 3.6(c), except we show the eigengap over all the shells rather than in groups. The blue (green) plot shows the eigengap for the original (modified) network.

3.7 Theoretical insights

We now theoretically demonstrate how the three properties described in section 3.2.4 lead to the formation of rich centrality clubs. We consider an ideal network, where the vertices of each shell form a connected component and as per property 2, the values of α are large enough such that each shell is an expander graph. Since an expander graph has no bottleneck, or clear partition, random graphs fulfill these criteria. We therefore assume that each subgraph, S_k induced by a shell k and its neighbors, is an Erdős-Réyni random graph, with n_k vertices and d_k average degree, and the probability of connection among a pair of vertices p_k; thus, $d_k \approx n_k p_k$.

In [35], the authors prove several bounds on the average path length in a random graph $G(n, p)$. In particular, they state that if $np \geq c > 1$, then the average distance is constant times $\frac{\log n}{\log np}$. Using this result, we assume that the average distance between two vertices in subgraph S_k is $\frac{\log n_k}{\log n_k p_k} \approx \frac{\log n_k}{\log d_k}$.

Now consider a path between the two vertices v_1 and v_n, that is the sequence of vertices

$v_1, v_2, \ldots, v_x, \ldots, v_{n-1}, v_n$. Let core numbers of these vertices be $l_1, l_2, \ldots l_x, \ldots l_{n-1}, l_n$. Let l_x be the highest numbered shell in this sequence. We assume that for all shortest paths between v_1 and v_n, $l_1 \leq l_2 \leq \ldots l_x \geq \ldots l_{n-1} \geq l_n$.

This means that the shortest paths travel monotonically from a source low shell to the highest shell required, and then back from the high shell to the destination low shell. Note that this assumption allows the path to remain in the same shell throughout as well. However, paths that zig-zag from a high shell to a low shell and back to a high shell are not allowed. This rationale is based on the fact that since lower valued shells are sparser, it is more likely that the paths will connect through higher shells than lower shells.

With these assumptions in place we can state the following

Lemma 1. *If there exists a shell k_x, such that, $\frac{\log n_{k_y}}{\log d_{k_y}} > r_{yz} + \frac{\log n_{k_z}}{\log d_{k_z}}$, for all $k_y < k_x$ and for at least one $k_z \geq k_x$, then high centrality vertices are located in core C_x.*

Proof. Let k_x be the lowest numbered shell that satisfies the equation in the lemma. Consider the path between two vertices v and u. If either v or u is in C_x, then the path between them has to pass through C_x. If neither of the vertices are in C_x, we have to consider two cases.

First case, the two vertices in the same shell k_a, $x > a$, then on average, the distance between them will be $\frac{\log n_{k_a}}{\log d_{k_a}}$.

Second case, the vertices from two different shells k_a and k_b, $x > a > b$. On average the length of the shortest path will be $r_{ba} + \frac{\log n_{k_a}}{\log d_{k_a}}$. This value is greater than the path simply going through k_a.

Thus for both cases, if $\frac{\log n_{k_a}}{\log d_{k_a}} > r_{ax} + \frac{\log n_{k_x}}{\log d_{k_x}}$, the shortest path between any two vertices in the graph is on average going to pass though C_x. Thus the core C_x will contain high closeness and betweeness centrality vertices.

□

As per property 1, $n_y > n_x$ and $d_y < d_x$, where $k_y < k_x$. Therefore the condition

$\frac{\log n_{k_y}}{\log d_{k_y}} > \frac{\log n_{k_x}}{\log d_{k_x}}$ will hold for any two shells. To maintain the condition of the lemma, we have to ensure that the distance from the steps to go from one shell to another, r_{yz}, is small enough. In other words, the steps to go from shell y to core z is smaller than the difference of their average distance. We have observed that for networks with RCC, r_{yz}, where z is the innermost or second innermost core, the value is between 2-4. The number of nodes in the outer shells can go upto thousands, thus easily satisfying the condition.

It might seem that because finding the eigenvalue is an expensive operation, identifying networks with RCC would also be more expensive than simply finding the high centrality vertices. However, note that our lemma is based on the average path per shell. This metric can be computed in parallel for each shell, and is faster than computing the centralities over the whole network.

3.7.1 Relation between the attack models

We now discuss how the three models are related to each other. It is easy to see that the rich club associative model is a generalization of the core associative model. If the network has just one single rich club, then that will be located at the core, and the two models become equivalent. We also observe in Figure 3.9, that the uniform perturbation model is more effective for networks with scattered rich clubs. This indicates a relationship between the rich club assortative model and the uniform perturbation model as well.

To discover this relation, we first analytically demonstrate that if the uniform perturbation attack is used, then *the larger the size of the clique, the greater the resilience.*

We have observed that with more successful attacks, the variability of the top ranking high centrality vertices decrease (see Figure 3.16). Based on this observation we construct a baseline graph where the variability of the values of degree, betweenness and closeness centralities is zero. For example, the class of distance regular graphs fulfills this criteria [143], but note that other graphs, not necessarily distance regular can also be constructed with zero variance of centralities. We then add cliques to alter the variability in the centrality values, and analytically prove that **(i)** the high centrality vertices are located in the cliques, and **(ii)** it is easier to disrupt a smaller size clique as compared to a larger one.

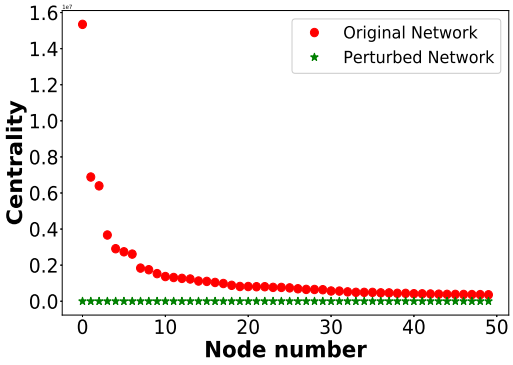


Figure 3.16: Change of betweenness centrality value due to perturbation for top-50 high central nodes (Network: as2).

Consider baseline graph, $G(V, E)$ where each vertex has degree α, and the diameter of the graph is d. We then select a set of vertices, C, and add edges such that the vertices in C form a clique. Let this be the new graph G_C.

Note that by adding a clique, the distances will only decrease from G to G_C and all new paths formed in G_C will have to include at least one vertex of C. Let the distance between two vertices a and b in graph G be given as $\delta_G(a, b)$. Since G is undirected $\delta_G(a, b) = \delta_G(b, a)$. For any vertex a, μ_a is the vertex in the clique C that is closest to x.

Lemma 2. *In the graph G_C, any vertex $x \in C$ will have higher closeness centrality than any vertex $y \in V - C$.*

Proof. We compare the distance of $y \in V - C$ and $x \in C$ to another vertex $z \in V; z \neq y, z \neq x$.

$$\delta_{G_C}(y, z) = \delta_{G_C}(y, \mu_y) + \delta_{G_C}(\mu_y, \mu_z) + \delta_{G_C}(\mu_z, z)$$
$$\delta_{G_C}(x, z) = \delta_{G_C}(x, \mu_x) + \delta_{G_C}(\mu_x, \mu_z) + \delta_{G_C}(\mu_z, z)$$

Since $x \in C$, $\delta_G(x, \mu_x) = 0$. Therefore the difference between the distances is;

$$\delta_G(y, \mu_y) + \delta_{G_C}(\mu_y, \mu_z) - \delta_{G_C}(\mu_x, \mu_z)$$

$\delta_G(y, \mu_y) \geq 1$, since y is not part of the clique. The distance between any two vertices in clique C can be either 0 or 1. If $\mu_z = \mu_y$ then the difference is at least 0, otherwise the difference is at least 1. Thus the distance of y to any vertex z is greater, or in some rare cases equal to the distance of x to z. Thus, the closeness centrality of y is less than that of x.

□

Lemma 3. *In graph G_C, given a vertex $y \in V - C$ and $\mu_y \in C$, μ_y will have higher betweenness centrality than y.*

Proof. Recall that betweenness centrality of a vertex v, is given as $BC_G(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the total number of shortest paths between s and t, and $\sigma_{st}(v)$ is the total number of shortest paths between s and t that pass through v. In the baseline graph, $BC_G(y) = BC_G(\mu_y)$.

We consider the following three cases.

Case 1: The distance between s and t remains unchanged in G_C and thus the relative value of $BC_{G_C}(y)$ and $BC_{G_C}(\mu_y)$is not affected for the pair (s, t).

Case 2: A path between s and t that used to go through y in G no longer goes through y in G_C. Thus $BC_{G_C}(y)$ is increased. Since $\mu_y \in C$, Case 2 will never occur for μ_y. Thus $BC_{G_C}(\mu_y) \geq BC_G(\mu_y)$ for the pair (s, t).

Case 3: A path between s and t that did not go through y in G now goes through y in G_C. Thus the value of $BC_{G_C}(y)$ is increased. However, all the new shortest paths have to include at least one vertex from the clique C. Since μ_y is the clique vertex closest to y, this path will also pass through μ_y. Therefore if the betweenness centrality of y increases, there will be an equal increase in the betweenness centrality of μ_y.

Thus for all cases, $BC_{G_C}(\mu_y) \geq BC_{G_C}(y)$.

□

Using the same rationale as the above proofs, we can show that if multiple cliques, not

necessarily of the same size, are added to G, the centralities of a vertex not part of a clique will be lower than the centralities of a vertex within a clique.

To disrupt the ranking of the high centrality vertices, we have to break the cliques. We show for the uniform perturbation attack, the probability of breaking a smaller clique is higher than the probability of breaking a larger clique.

Lemma 4. *Consider a graph $G_{C1,C2}$ that is built by adding two non-overlapping cliques to the baseline graph G. Under the uniform perturbation model, the probability of removing a vertex x in any one of the cliques is $P(x)$. In sufficiently large graphs, if the size of clique $C1$ is less than the size of clique $C2$, then $P(a) > P(b)$ for $a \in C1, b \in C2$.*

Proof. Let the degree of each vertex in the baseline graph G be α, and the total number of edges be E. Let the size of the clique $C1$ be $r1$. The degree of the vertices in $C1$ is $p = \alpha + r1 - 1$. Let the size of the clique $C2$ be $r2$, and the degree of the vertices in $C2$ $q = \alpha + r2 - 1$. Also $r2 > \ r1$.

If we use the uniform perturbation model to delete the edges, then the probability of deleting the $r1$ edges from a vertex in the clique $C1$, which has degree p is

$$D_{C1} = \frac{p}{E}\frac{p-1}{E-1}\cdots\frac{p-(r1-1)}{E-(r1-1)}$$

.

$$= \frac{p!\,(E-r1)\,!}{E!\,(p-r1)\,!} = \frac{p!\,(E-r1)\,!}{E!\,(\alpha-1)\,!}$$

.

Similarly the probability of selecting $r2$ edges from a vertex in clique $C2$, which has degree q is

$$D_{C2} == \frac{q!\,(E-r2)\,!}{E!\,(\alpha-1)\,!}$$

.

Given that $q - p = r2 - r1$ and $(E - r1) - (E - r2) = r2 - r1$, the ratio $\frac{D_{C1}}{D_{C2}}$ is,

$$\frac{D_{C1}}{D_{C2}} = \frac{p!\,(E-r1)\,!}{q!\,(E-r2)\,!}$$

$$= \frac{(E - r2 + 1)}{(p + 1)} \frac{(E - r2 + 2)}{(p + 2)} \cdots \frac{(E - r1)}{q}$$

If the graph has a large number of edges such that $E > p + r2$ or $E > \alpha + r1 + r2$, then each of the fractions of the form $\frac{(E-r2+x)}{(p+x)}$ will be greater than 1. Thus $D_{C1} > D_{C2}$, and the probability of disconnecting a vertex from $C1$ is greater than the probability of disconnecting a vertex from $C2$.

□

Thus smaller cliques are more affected by the uniform perturbation model. Given a fixed value of k top ranked centrality vertices, the higher the number of scattered rich clubs, the more likely that they are to be smaller in size. Smaller the size of the rich clubs, the more likely uniform perturbation will be effective in disrupting the network.

3.7.2 Trade off in cost and effectiveness of the attack models

The three attack models demonstrate a sliding scale of cost versus effectiveness. The most effective model, rich club assortative perturbation, is also the most expensive to implement. In contrast, the uniform perturbation model is the cheapest and the least effective. Moreover, applying the uniform perturbation model requires no knowledge of the network structure, whereas to apply the rich club model one should at least have information about the edges whose vertices are in nearby cores. We also observe that the uniform perturbation and the core associative model represent the two extremes of the scattered rich clubs. Uniform perturbation is most effective when the scattered rich clubs are very small, and the core associative model is most effective when the scattered rich clubs form just one dense club. An interesting future research direction would be to explore when to use which model or combination of models, given our knowledge of network structures and limits on computation costs.

3.8 Summary of the chapter

To summarize, our **key contributions** are as follows:

- We show a novel network data categorization, where in one group of network high central nodes are located within the inner most core and these inner most core nodes expand out in several dense modular units in the network. We further show evidence of *rich centrality club* using supervertex based network summarization and empirically validate that supervertex composed of inner most core doesnot always play pivotal role in governing connectivity in some networks.
- Through comprehensive set of experiments, we show that distinctive spectral signature emerges from networks with *rich centrality clubs*. Specifically the normalised cut gradually decreases for subgraphs formed using shell volumes, as we traverse from the inner to the outer shells. This also implies that random pairs of nodes in a shell is dependent on the inner shell nodes for shortest paths.
- We extend our proposal of single rich clubs with the novel concept of scattered rich clubs. We posit that high central nodes instead of organising in the inner most layers of the network, can be scattered in dense modular units of the network. We further develop a clique percolation based approach to discover such scattered rich clubs in the network.
- We show that networks with rich centrality clubs are resilient against random edge perturbation based attacks. However we propose novel targeted edge perturbation based attacks which are effective in disturbing original rank order significantly in the case of *RCC* based networks
- We also provide an edge rewiring strategy to insert RCC into a non-RCC network. Our strategy is simple, reversible and does not alter the key global network properties drastically. We show that this selective rewiring of connections helps in stability of high central nodes, by the changing spectral properties.
- We provide justification of our empirical results through theoretical arguments.

Chapter 4

Centrality of time-varying networks

4.1 Introduction

One of the important problems in time-varying networks is predicting how their features change with time. If this information is known a-priori using minimal computation, then users can take appropriate action in advance to utilize such features. The most significant among network properties are the centrality features, that are used to estimate the importance of a vertex in a network.

Information can spread more quickly when high closeness centrality vertices are selected as the initial seeds. Similarly, vaccinating high betweenness centrality vertices, through which most of the shortest paths pass, can reduce the spread of disease. The central vertices also play an important role in spreading influence in a social network as has been observed in several works [5, 27, 108, 129]. In a dynamic setting (where the network changes over time) knowing these highly central vertices beforehand is of prime importance as it would facilitate in developing strategies for targeted advertising or setting up infrastructure for vaccination drives. However, this might result in expensive re-computation of shortest paths as the network varies over time. Our goal is to develop algorithms based on the network structure, so that such re-computations are avoided.

Current approaches focus on predicting the average centrality values of the network [89].

However, note that most applications, such as the ones discussed above, require the ids of *only the top-k centrality vertices*, not the values or the ranking of *all* the vertices in the network. Therefore, simply predicting the average centrality over the entire network may not be useful in a majority of the practical contexts.

In this chapter, we **present a two-step algorithm for predicting the high centrality vertices of time-varying networks**. In the first step, we predict the overlap between the set of high centrality vertices[1] of the current time step to the set of high centrality vertices of the future time step. In the next step, assuming that the network snapshot is already available in time, we analyze its innermost core to find the ids of the high centrality vertices.

The rationale for our prediction method: The key to our prediction method is based on a unique *hypothesis* that in many real world time-varying networks, *most of the highly central vertices reside in the innermost core*. In other words, a *large fraction of the shortest paths connecting pairs of vertices in such networks pass through the innermost core*; the vertices in the periphery (and the outer shells) of the network are mostly connected via the vertices residing in the innermost core of the network. A key contribution of our work is that we develop a set of *novel heuristics to classify networks based on the extent to which the highly central vertices are in the innermost core*. Our heuristics do not require the explicit computation of the centrality values. We also separately report our predictions for each class of networks. We empirically demonstrate that the higher the number of high centrality vertices in the inner core, the higher is the accuracy with which we can predict these vertices for future time steps. For real networks that maintain this property to the largest extent, our $F1$-score for prediction is **0.81** for closeness and **0.72** for betweenness. similarly, for synthetic networks that maintain this property to the best extent, the $F1$-score for prediction is **0.94** for closeness and **0.92** for betweenness.

Validation: In addition to $F1$-scores, we further validate our results by comparing how the predicted and actual high centrality vertices perform in a practical context. For high closeness centrality vertices, we compare the time to spread a message when the high centrality vertices are taken as seeds, and for the high betweenness centrality vertices, we compare how the length of the diameter increases as the high betweenness centrality vertices are

[1] In this chapter, when we mention highly central vertices, we specifically refer to high closeness or betweenness centrality and not other types of centralities.

deleted from the network. For these experiments we select a set of random vertices as control, and compare the performance of the actual high centrality vertices, predicted high centrality vertices and the randomly selected vertices. For networks where we could predict the results with high accuracy, the effect of the original and predicted vertices are very similar, and these results are markedly different from the effect of the randomly selected vertices. Interestingly, for networks, where our prediction accuracy is low, the effect is similar for closeness/betweenness centrality for all the three sets of vertices. This result indicates that networks with low prediction accuracy do not have significantly high closeness/betweenness centrality vertices and therefore the prediction itself does not serve any practical purpose. To summarize, our **key contributions** are;

- Develop a set of heuristics to classify temporal networks based on the extent to which the highly central vertices are part of the innermost core (section 4.2). To perform the classification, these heuristics do not require the explicit computation of the high centrality vertices.
- Develop an efficient algorithm to predict the high betweenness and closeness centrality vertices. To the best of our knowledge, this is the first algorithm to predict the vertices that does not require explicit computation of the centralities (section 4.3).
- Validate our results in practical application scenarios, namely message spreading and increasing the diameter, to show that the effect of our predicted vertices is similar to the actual ones. For networks, where our prediction accuracy is low, we show that even a random selection of vertices can produce same results as the actual high closeness/ betweenness centrality vertices. This indicates that for such networks, there are no significantly high closeness/betweenness centrality vertices. Thus, in these cases, prediction of high closeness/betweenness centrality vertices is of no practical use.
- Present a theoretical rationale for our algorithm.

Our condition for accurate prediction of high centrality vertices is that they are part of the innermost core of the network. As we shall see, in some real-world time-varying networks in our dataset, this condition holds. This condition is also true for a large number of synthetically generated networks. Even for the networks where the condition only holds to a

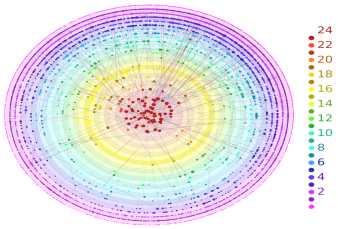
moderate extent, our predictions are reasonably well. We also observe that for those networks where the condition fails to hold, it is not really worth finding high centrality vertices (by any algorithm) as there is no explicit ranking present in such networks. The functional effect of the high centrality vertices is equivalent to those of a randomly selected set of vertices.

4.1.1 Our hypothesis

We hypothesize that the high centrality vertices in many real world time-varying networks are more likely to be located in the innermost core. As a first step, we note that if most of the shortest paths pass through the innermost core, then the high centrality vertices would also be part of the innermost core. Moreover, if these vertices are tightly connected to each other, they can enhance each others' centrality values [171]. Thus, a dense innermost core through which most of the shortest paths pass provides us a smaller subset in which to search for high centrality vertices.

However, not all networks have such dense innermost cores. Figure 4.3 visually contrasts a network that has a small dense inner core (CA) to a network where the inner core is sparse and forms the bulk of the network (FW). From visual inspection, it is clear that it is easier to locate the high centrality nodes in the Caida network than in the network of Facebook users.

Nevertheless, explicitly computing whether a network conforms to our hypothesis is expensive since that would lead to calculating all pairs of shortest paths between the nodes repetitively as the network changes over time. We therefore provide a set of novel heuristics in the next section to classify networks based on the extent to which they conform to our hypothesis.



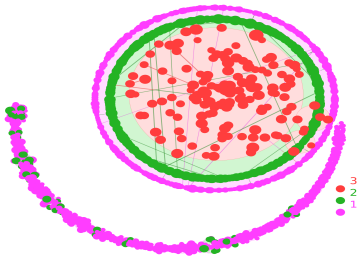


Figure 4.1: Caida network **Figure 4.2:** Facebook network

Figure 4.3: Visualization of the core-periphery structure with the corresponding shell index created using Lanetvi [7]; sizes of nodes are ordered based on the degree. Note that the Caida network has several layers of cores and and a small dense innermost core. In contrast, the Facebook network has only three cores of which the innermost core is sparse and predominant. Best viewed in color.

4.2 Classification of the networks

In this section we propose the following four parameters to classify the temporal networks according to whether the high centrality vertices are within the innermost cores. Note that each of these parameters are less computationally expensive than computing the centrality of the vertices. Our heuristics are as follows:

- **Fraction of inter-edges connected to the top core (EF)**: This metric is the ratio of the number of inter edges with one end point in top core to the total number of inter edges. *The higher the fraction, the more the network will have high centrality vertices in the top core.*
- **Average density of the non-top cores (CFX)**: This metric computes the average density of all cores, except the top one. The lower the density, the sparser the core, and the higher the average intra-core distance. The density of each core is computed as ratio of the number of intra-core edges in each core by the total possible edges

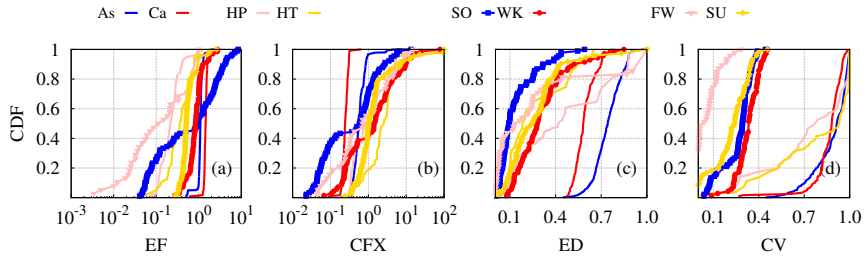


Figure 4.4: Classification of the networks according the distribution of the parameters. From left to right the parameters are, (a) fraction of inter-edges connected to the top core (EF), (b) average density of the non-top cores (CFX), (c) the density of the top core (ED) and (d) the overlap in the top-core at consecutive time steps (CV). Here AS, CA, HP, HT, SO, WK, FW and SU, given by the lines in different colors, represent the datasets. The X-axis represents the time points and the Y-axis plots the Cumulative Distribution Function (CDF) for each of the parameters.(Please refer to Table 4.1 for detailed description.)

between the vertices in the core. To find the average density we divide this value by the number of cores. *The lower this value the more the network will have high centrality vertices in the top core.*

- **Density of the top-core (ED)**: We compute the density of the top-core which is the ratio of the number of intra-core edges in the top core to the total possible edges between the vertices in the core. *The higher this value, the more the network will have high centrality vertices in the top core.*

- **Top-core overlap (CV)**: This metric takes into account the changes in the top core structure over consecutive time steps. We measure the overlap as the Jaccard similarity between the vertices in the top cores of networks at time step $t-1$ and t. *The higher the value, the greater the overlap.*

We explain the classification framework based on the above parameters in the following steps.

(A) For a set (D) of initial networks:

1. We consider the $t-1$ temporal snapshots $G_1, G_2, ..., G_{t-1} \in D$. We calculate the 4 tuple heuristic for each snapshot. Hence each snapshot can now be represented as (EF, CFX, ED, CV)

2. We calculate the cumulative distribution function (CDF) for each parameter from the $t-1$ values obtained for that parameter. For a particular parameter (say EF) we now have CDF for each of the initial datasets considered (Figure 4.4 shows the CDFs for the eight real world networks). Note that we compute the CDF over a range accumulated from all the parameter values obtained from all the previous time steps. In particular, it is not possible to compute the CDF if multiple values of the variable are not available, thus highlighting the use of multiple time points and consequently the dynamical properties of the network.
3. We perform hierarchical clustering on each parameter, using D-statistic as the pairwise similarity measure between the CDFs of a particular parameter from two different networks. We cut the dendrogram at cluster size 2, hence having the clustering algorithm output 2 clusters (C_1, C_2) for each parameter.
4. If the mean value of parameter in C_1 conforms more to the desirable property, i.e., have high values for EF, ED, CV or have low value for CFX, we label C_1 as good (G) and C_2 as bad (B) or vice versa. We have outlined our framework in Figure 4.5.

(B) For each new (unseen) network:

1. Once again we compute the CDFs for each of the four parameters from the $t-1$ snapshots.
2. Next we obtain the similarity (D-statistic) of the CDF of a particular parameter with the centroid of both C_1 and C_2 (i.e., the traditional Rocchio technique [10]).
3. Finally, we classify the network to that class to which it is more similar (based on the similarity with the centroid of the class.)

We use eight real world networks as the initial set and 20 synthetic networks as unseen and classify them based on the Rocchio scheme. The category for each network are presented in Table 4.2 and Table 4.8 for real and synthetic networks respectively. For all the parameters, except CFX, higher value is good. For CFX, lower value is desirable. This classification matches with the results in Table 4.2 and Table 4.8, i.e., if EF, ED and CV are high and

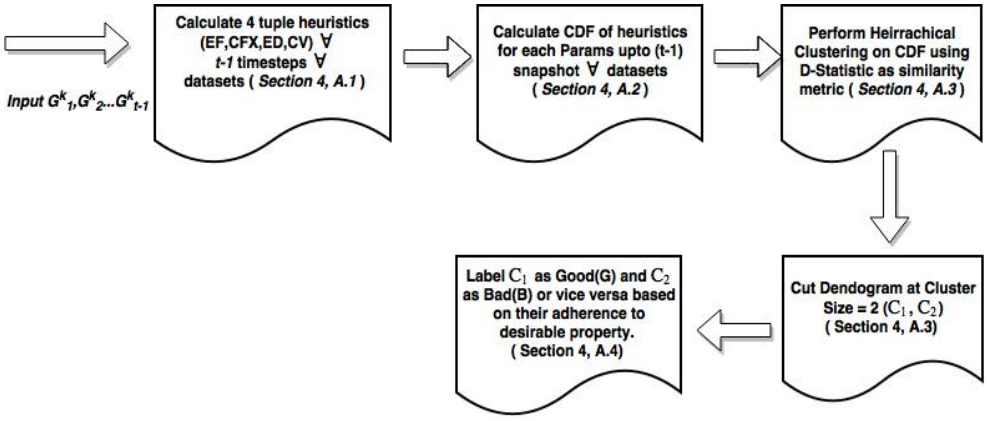


Figure 4.5: Classification framework.

CFX is low then the high centrality vertices in the networks can be predicted with higher accuracy using our scheme.

Figure 4.4 shows the distribution of the different parameters for each of the real world networks. The parameters for top core overlap and top core density form the most distinct clusters. The clusters for fraction of edges to the top core, and intra-core density are not as distinct, indicating that almost all the networks conform to our hypothesis at least to some extent and that the size and density of the top core are the critical factors in determining predictability. The groupings for each parameter and for each network are noted in Table 4.2 (real networks) and Table 4.8 (synthetic networks).

4.3 Algorithm for predicting high centrality vertices

Our prediction framework is composed of three steps. In the first step we classify the network based on the four parameters introduced in the previous section. In the second step we use the already computed overlap time series to predict the future overlap values using ARIMA models. In the final step we identify high centrality vertices in the network at the next time step.

- **Step 1: Classifying the networks based on the parameter values across time steps**:

Algorithm 2: Calculate the classification parameters.

Data: $G_{t-1}(V, E)$, graph at time step $t-1$; l_k set of top core vertices at time $(t-2)$

Result: EF, CFX, ED, CV

$c[v] \leftarrow$ `FindCore(`G`) // Returns core number of each vertex` $v \in G$;

$C_m \leftarrow$ `MaxCore(`c`) // Returns max core number from c` ;

$L_k \leftarrow \{\}$;

$N(v) \leftarrow$ neighbors of v ;

$tempV[C_m], tempE[C_m], tempD[C_m] \leftarrow [0]$;

for $\forall v \in V$ **do**

  for $u \in N(v)$ **do**

    if $c[u] \neq c[v]$ **then**

      if $(c[u] == C_m \vee c[v] == C_m)$ **then**

        $e_m \leftarrow e_m + 1$;

      $e_i \leftarrow e_i + 1$;

$EF \leftarrow e_m / e_i$;

for $\forall v \in V$ **do**

  for $u \in N(v)$ **do**

    if $c[u] == c[v]$ **then**

      $tempE[c[u]] \leftarrow tempE[c[u]] + 1$;

      $tempV[c[u]] \leftarrow tempV[c[u]] + 2$;

for $u < C_m$ **do**

  $tempD[u] = 2 * tempE[u] / (tempV[u] * (tempV[u] - 1))$;

for $u < C_m - 1$ **do**

  $d \leftarrow d + tempD[u]$;

$CFX \leftarrow d/(C_m - 1)$; $ED \leftarrow tempD[C_m]$;

for $\forall v \in V$ **do**

  if $c[v] == C_m$ **then**

    $L_k \leftarrow L_k \cup v$;

$CV \leftarrow$ `Jaccard(`l_k, L_k`)` ;

return *EF, CFX, ED, CV*;

Algorithm 3: Predict top central vertices

Input: $G_t(V, E)$, graph at time step t, $N_v = |V|$

Output: T[k]

```
1  temp[N_v] ← [0];
2  m, j ← 0;
3  c[v] ← FindCore(G);
4  C_m ← MaxCore(C) // Returns max core number from c;
5  for i ← 1 to N_v do
6      if c[i] == C_m then
7          temp[j] ← i;
8          j ← j + 1;
9  for i ← 1 to k do
10     n ← MDVid(temp) // Max Degree vertex id;
11     T[m++] ← temp[n] ;
12     temp[n] ← 0;
13 return T;
```

Our first step is to classify the networks to see whether the vertices with high centrality are consistently located in the top core over all the t time steps. To obtain this classification, we compute the parameters, defined in section 4.2, for networks at every time step from 1 to t. The complexity for computing the parameters are as follows:

To compute these parameters, we first need to compute the core numbers of the vertices. We do this using the function $FindCore$ (see line 1, Algorithm 2), which implements the algorithm presented in [12] and has a complexity of $O(|E|)$, where $|E|$ is the number of edges.

For computing fraction of inter core edges connected to the top core, i.e., *EF*, (lines 6–12, Algorithm 2), we need to iterate over all edges once and keep count of the number of edges where the end points belong to different cores and at least one of them is part of the top core. Thus the complexity is $O(|E|)$.

To compute the density of each core, we sum the number of the edges whose endpoints are both in that core and then divide this sum by the total number of possible

edges in the core. Using these values we can compute the average density of all the non-top cores, i.e., *CFX*, as well as the density of top core, i.e., *ED*, (lines 13-22). Once again computing these parameters requires us to go through all the edges, and thus has complexity $O(|E|)$.

The overlap between the top core vertices, *CV*, in the networks over two consecutive time steps is computed using Jaccard similarity (see Algorithm 2). Finding the vertices in the top core, requires us to go over all the vertices and identify their core numbers (complexity $O|E|$). The complexity of computing the Jaccard similarity is linear to the number of vertices in the top core.

Once we obtain these parameters for all the networks over the time steps 1 to t, we classify each network by the technique described in the previous section. Based on the classification, the network might fall in the good class G (i.e., majority or at least equal number of parameters fall in the good class) or in the bad class B. *Note that this classification, requires information from not just one time step, but from multiple time steps thus establishing how our framework is dependent on the dynamical properties of the network.* If the network falls in class G we proceed to Steps 2 and 3.

- **Step 2: Estimating overlap among the top central vertices**: After analyzing the networks from time steps 1 to t, consider the network G_{t+1} at time step $t+1$. Even if the network is not available, we leverage information about the Jaccard overlap between the top core vertices for consecutive time steps from 1 to t, and use the autoregressive-integrated-moving-average (ARIMA) algorithm [18] to estimate the overlap in top cores of G_t and G_{t+1}.

- **Step 3: Identifying the top central vertices**: If the network is classifed into class G (good), then we identify the top central vertices in the new network G_{t+1}, using Algorithm 3.

  We identify the vertices in the top core of the new graph using the $FindCore$ function (line 3, Algorithm 3). Then we implement a m-search algorithm to extract the m highest degree vertices (lines 5-12, Algorithm 3) of the graph, that are in the top core. These vertices are marked to be the high centrality vertices.

If the network falls in the class *B*, a random selection of nodes from the network and the

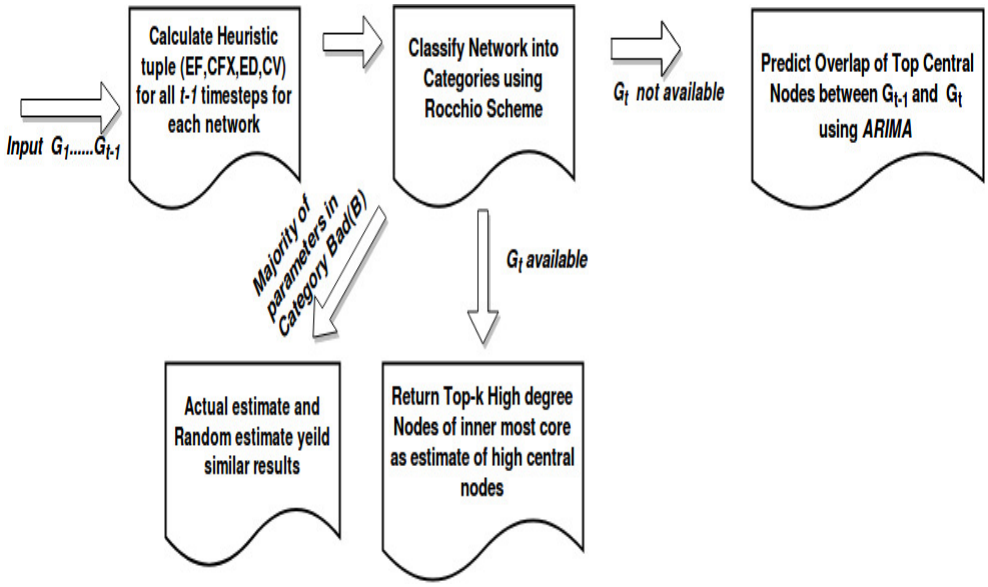


Figure 4.6: Prediction framework.

actual high central nodes perform equivalently with respect to path based centrality (substantiated by the validation results in section 4.6.)

We now provide further details on the most crucial steps 2 and 3 in the rest of this section.

4.3.1 Estimating the extent of overlap

For a given temporal network (G) with network snapshots $G_1, G_2, \ldots$ observed at times $1, 2, \ldots$, we calculate for initial few time steps the top (5) 10 central vertices based on betweenness and closeness. The overlap l_c^t (closeness) and l_b^t (betweenness) are calculated as the Jaccard overlap between the top (5) 10 central vertices between network snapshots G_t and G_{t+1}. Note that this is calculated for the initial few time steps to eliminate the cold start problem. We next represent l_c^t (l_b^t) as a set of points ordered in time or equivalently a time series. This representation allows us to leverage time series forecast models to predict the values of l_c and l_b at a future time step. Specifically, we use the ARIMA model to fit the resulting time series. On fitting an ARIMA(p,d,q) model to a time series we obtain an auto-regressive equation of the form -

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \beta_1 e_{t-1} + \cdots + \beta_q e_{t-q} + c$$

where y_t represents the value of the time series at time t, $e_t, e_{t-1} \ldots$ are the white noise terms and α_i and β_i are parameters of the model. To summarize, given the centrality overlap values till $t-1$, we are able to predict it at time t. If the top central vertices are known for G_{t-1} and we know the extent of overlap among top central vertices between G_{t-1} and G_t (which we predict using our proposed method), we can roughly estimate the top central vertices in G_t given the predicted overlap value is high and the prediction error is low. Note that *for the above technique to work, the network G_t itself is not required.* We show later (section 4.5) that our method is indeed able to predict the overlap values for certain networks with very low error.

4.3.2 Identifying the top central vertex

The first step of the proposed prediction scheme allows for estimating the extent of overlap among the top central vertices between the two consecutive snapshots of the network. In this step we further refine the prediction when the network G_t is available. Specifically, instead of explicitly calculating the centrality values, we use the algorithm described in [12] to obtain the core-periphery structure and, thereby, identify the vertices in the top core. Our analysis in section 4.7, suggests that these vertices are the most likely candidates for being the high central ones in the network. To obtain the top m central vertices in the system, we rank the vertices in the top core based on their degree and filter out the top m vertices (see Algorithm 3). Note that this step has complexity $O(|E|)$ (for computing the core and ranking the top-core vertices by degree), which is significantly less than $O(|V|*|E|)$, the complexity for computing all the closeness and betweenness centrality vertices and then ranking them.

The innermost core hence acts as a 'container' for the high central nodes in certain class of networks which we explain in Section 4.7. We also empirically show this by performing the following experiment: instead of predicting the high degree nodes from within the top core as the high central nodes we directly predict the globally highest degree nodes agnostic of which core they belong to. This results in drastically poorer $F1$-score values as noted

in Table 4.5.

We also compare the results obtained from our prediction algorithm against existing algorithms for forecasting high centrality nodes in temporal networks [89]. Note that all these schemes are compute intensive as they require explicit computation of the centrality values in all the earlier time steps for prediction in the current time step unlike our approach. For both closeness and betweenness centrality our results are superior compared to existing state-of-the-art in almost all cases of the cases.

We illustrate the complete flowchart of our prediction framework in Figure 4.6. Note that all the subsequent steps of prediction are dependent on the initial classification step which is completely regulated by the temporal dynamics of the network under consideration.

In section 4.5, we show that our proposed method is indeed able to identify a large fraction of central vertices in the network without explicitly calculating the centrality values (betweenness and closeness) for each vertex.

4.4 Experimental setup

In this section we describe the networks in our test suite. We consider 8 diverse type of real world networks. We also consider 20 synthetic networks of varying numbers of nodes, edges and temporal snapshots and generated using two different tools – Musketeer [65] and Dancer [14].

4.4.1 Test suite of real-world networks

We consider a diverse set of benchmark networks of various sizes and over discrete timescales. The networks are collected from public repositories made available at [95, 99]. Our networks can be grouped into three different categories according to the application domain. Given below is a brief description of the categories, the networks in them, and how we obtained the time series network from the data. The sizes of the networks are given in

Table 4.1.

- **Autonomous systems network:** The Internet is sub-structured as interconnected subgraphs of highly connected routers. These subgraphs are known as autonomous systems (AS). Each AS exchanges traffic with neighbors (peers) using BGP (Border Gateway Protocol). Here we use two example networks; both were created using BGP table snapshots and made publicly available.

  AS: The first dataset (AS) was collected from University of Oregon Route Views Project and it contains 733 daily data traces between autonomous systems which span an interval of 785 days from November 8 1997 to January 2 2000.

  CA: The second dataset was collected by Center for Applied Internet Data Analysis (CA) from January 2004 to November 2007 and it comprises anonymized interaction of ISP's.

- **Citation network:** This type of network connects two papers if one paper cited the other. Although the links are directed, for purpose of our experiments we consider them to be undirected.

  We use citation data from two different research topics in high energy physics. For both these networks, every paper is timestamped by the submission time to the archive. We also have a list of papers which are cited by a submitted publication. Based on this information, we created an aggregated growing network in terms of months and considered each network as a distinct snapshot. For citation network a node once added is not deleted. This is not the case for the other classes of networks.

  HepPh (HP): (High Energy Physics Phenomenology) is a citation graph from the e-print arXiv and covers all citations within a dataset of 34,546 papers with 421,578 edges. Citation between paper i and j, is represented as an edge. The data covers an almost complete set of papers from January 1993 to April 2003.

  HepTh (HT): (High Energy Physics Theory) is a citation graph similar to HP, with 27,770 papers with 352,807 edges. The data focuses on papers from January 1993 to April 2003.

- **Social communication networks:** These networks are of interactions via different types of social media. We study four different networks with each individual edge ac-

companied by unique timestamp. For all these networks we aggregated all the edges appearing in the same month and created a single temporal snapshot of the network per month.

StackOverflow (SO): On stack exchange web sites, users post questions and receive answers from other users, and users may comment on both questions and answers. A temporal network is derived by creating an edge (u, v, t) if, at time t, user u: (1) posts an answer to user v's question, (2) comments on user v's question, or (3) comments on user v's answer.

Facebook Wall (FW): The edges of this dataset are wall posts between users on Facebook located in the New Orleans region. Two users are connected if they post on the same wall.

Wiki Talk (WT): This dataset represents edits on user talk pages on Wikipedia. An edge (u, v, t) signifies that user u edited user v's talk page at time t.

Superuser (SU): This dataset is derived from question answer site Superuser which exists for computer enthusiasts. As in the case of StackOverflow, an edge (u, v, t) exists if, at time t, user u: (1) posts an answer to user v's question, (2) comments on user v's question, or (3) comments on user v's answer.

4.4.2 Test suite of synthetic networks

We also consider 20 synthetic networks generated using the multi-scale network generation tool Musketeer [65] and the dynamic attributed networks generation tool Dancer [14]. We generate several temporal snapshots of each network with various core structures to present additional prediction results and, thereby, further strengthen our hypothesis.

Network	Nodes	Unique Edges	Temporal Edges	Time Span
AS	7716	27183	57,05405	732
CA	31255	111564	54,85410	120
HP	34564	4,21578	4,21578	124
HT	27770	3,52807	3,52807	124
FW	46952	274,086	876,993	48
SU	194,085	9,24886	1,443,339	92
WT	1,140,149	3,309,592	7,833,140	73
SO	2,601,977	36,233,450	63,497,050	92

Table 4.1: Test suite of real world networks used for our experiments. For AS and CA the time span is measured in days, for all others in months. Combined entire edge stream for any dataset comprises the temporal edges; unique edges are temporal edges with duplicates removed.

4.5 Empirical results

In this section we present the empirical results to demonstrate the effectiveness of our prediction algorithms as proposed in section 4.3.

4.5.1 Results on real world networks

Our prediction scheme consists of two steps and we evaluate each of them separately.

Extent of overlap

We measure the effectiveness of the prediction scheme using cross-validation technique. More specifically we consider each network and predict the overlap at different time steps for both betweenness and closeness. If for a given time step t, the original overlap value is o_t and the predicted value is p_t, we define the error in prediction $error_t$ as

$$error_t = \frac{|o_t - p_t|}{o_t} * 100$$

Note that for this prediction we assume that the actual overlap values for a certain time stretch till $t-1$ is available. In our experiments this time stretch has been set to 20 time steps until $t-1$[2].

In Table 4.2 we present, along with the mean original overlap values, the average and standard deviation of the error percentage across all the time steps for each dataset for both betweenness and closeness centrality. We present results considering top 10 high closeness and betweenness centrality vertices. However, our scheme also works very well for even a much more restricted set of even five vertices (see Table 4.2)[3]. We observe that the error is very low for the networks in $GGGG$ (all parameters good) category (AS, CA) while those in $BBBB$ (all parameters bad) category (WT, FW, SU) show much higher error rates.

Comparison with other time series models: We further look into other models of time series prediction (AR, MA and ARMA in specific) and mean prediction error (both betweenness and closeness) for these models are reported in Table 4.3. For networks in $GGGG$ category, these simpler models are able to predict the extent of overlap with very low error but their efficiency reduces as we move toward the other classes of networks.

Table 4.2: Classification as well as the prediction performance for the datasets used for evaluation. Each dataset is classified as a four tuple (EF, CFX, ED, CV) (column 1) with G representing good and B representing bad. Mean (μ), std. dev. (σ) are reported for both prediction error and $F1$-score (columns 3 to 8). The categories are colored as per the groups they belong. Note that the higher the number of Gs in the category, the more accurate the prediction results.

N/w Category	N/w Name	Mean overlap	Close. Pred. (top 5, μ, σ)	Bets. Pred. (top 5, μ, σ)	Close. Pred. (top 10, μ, σ)	$F1$-score (top 10, μ, σ)	Bets. Pred. (top 10, μ, σ)	$F1$-score (top 10, μ, σ)
$GGGG$	AS	0.79	.62,10.61	7.78,10.86	5.69,6.37	0.81,0.06	6.97,7.68	0.72,0.08
$GGGG$	CA	0.82	11.68,26.82	3.59,5.67	8.76,6.02	0.77,0.08	9.17,6.47	0.64,0.07
$GGBG$	HT	0.56	12.52,11.71	20.79,16.10	26.96,17.44	0.42,0.35	20.74,14.86	0.52,0.30
$GGBG$	HP	0.43	26.76,17.44	20.74,14.86	11.64,5.76	0.42,0.33	14.22,11.23	0.46,0.29
$BBBG$	SO	0.29	34.92,33.39	45.30,38.48	27.96,21.69	0.35,0.26	26.15,24.72	0.39,0.30
$BBBB$	WT	0.05	47.10,36.56	59.84,43.36	41.77,31.33	0.32,0.17	36.55,28.70	0.31, 0.22
$BBBB$	FW	0.09	131.89,148.42	169.57,158.51	109.90, 92.39	0.24,0.25	56.19, 34.95	0.20,0.19
$BBBB$	SU	0.03	44.45,40.14	167.25,130.42	147.06,106.53	0.02,0.09	32.58,40.14	0.18,0.21

Prediction using previously predicted values: Note that in the experiments discussed

[2]We tried with other stretches of size 15, 25 etc. The results do not seem to be affected by such minor variations. Ideally this size should not be too large thus consuming a lot of data for prediction, nor it should be too small thus having too few points to correctly predict. Through experimentation, we find that a size close to 20 strikes an ideal balance.

[3]Note that if we keep increasing the number of top vertices, the prediction results can only get better. Through experiments, we observe that small numbers like 5 and 10 are judicial choices.

Table 4.3: Prediction performance of AR, MA and ARMA time-series prediction models across all the datasets. Both mean (μ) and std. dev. (σ) are reported in each case. Predictions are made considering top 10 central vertices.

	AR		MA		ARMA	
Datasets	$BC(\mu, \sigma)$	$C^lC(\mu, \sigma)$	$BC(\mu, \sigma)$	$C^lC(\mu, \sigma)$	$BC(\mu, \sigma)$	$C^lC(\mu, \sigma)$
AS	7.72, 10.75	6.93, 12.37	7.58, 10.01	6.37, 9.53	7.48, 10.16	6.45, 10.37
CA	10.78, 8.82	9.26, 6.13	10.45, 9.11	9.15, 6.17	10.56, 8.97	9.23, 6.12
HT	21.72, 12.65	27.58, 11.09	22.15, 11.71	28.56, 19.21	22.67, 14.63	26.82, 14.32
HP	19.73, 14.97	26.96, 17.44	21.31, 15.29	24.37, 13.35	20.41, 14.98	25.56, 18.14
SO	35.02, 33.30	33.62, 32.46	34.92, 33.39	32.36, 31.29	34.56, 32.42	32.12, 32.62
WT	62.09, 43.81	45.86, 35.98	60.54, 45.79	44.63, 35.64	60.55, 44.39	44. 56, 34.88
FW	135.48, 122.68	241.68, 215.57	132.58, 123.45	163.68, 149.42	132.26, 122.83	157.70, 150.34
SU	44.59, 43.49	167.25, 145.42	41.16, 38.23	169.26, 143.45	45.97, 44.42	166.52, 141.78

above, at each point of prediction (t) we consider the original values of the series between $t-1$ and $t-20$. We further explore the case where instead of using the original values we use the predicted values. More precisely, for predicting the values at point t, we use the predicted values between $t-1$ and $t-20$. Note that to avoid cold start problem we use the original values for the first point of prediction but as we move along we keep on using the predicted values instead of the original ones.

Consequently, we observe that the prediction error increases drastically in case of FW and SU while for AS and CA, the effect is negligible (see Table 4.4). We report the results considering the top 10 vertices; however, the results considering five vertices show exactly similar trend. This once again demonstrates that our classification can identify networks where high centrality vertices can be predicted accurately.

Table 4.4: Percentage error in prediction (mean(μ), std. dev. (σ)) for all the datasets using predicted values repeatedly for prediction. The results are reported considering top 10 central vertices.

Dataset	AS(μ, σ)	CA(μ, σ)	HP(μ, σ)	HT(μ, σ)	SO(μ, σ)	WT(μ, σ)	FW(μ, σ)	SU (μ, σ)
CC	6.30,7.37	13.02,11.25	14.56,10.21	32.46,30.77	37.58,35.07	76.40,91.85	158.86,121.21	172.16,114.23
BC	8.21,13.30	10.92,16.12	25.91,14.29	19.54,12.48	32.98,25.45	44.42,28.57	73.66,30.95	41.02,26.27

Identifying top central vertices

Once the network has arrived in time, we further refine our prediction to identify exactly the top 10 central vertices based on betweenness and closeness centrality values. We further

obtain the predicted set of top central vertices based on the prediction scheme described in section 4.3.2. To measure the effectiveness of our scheme we compute the $F1$-score between the predicted and the original set. We repeat the result for each time step and the average $F1$-score as well as its standard deviation are reported in Table 4.2. The results are separately reported for betweenness and closeness centrality. We again observe that the best prediction result is obtained for the networks in the $GGGG$ category. For the networks in the $BBBB$ category the obtained $F1$-score is the least. Once again, while we report results considering the top 10 vertices, the results for the top five vertices show an exactly similar trend.

4.5.2 Necessity of computing the core

At this point one might question whether or not computing the innermost core is indeed required. We posit that the innermost core acts as a 'container' for the high central nodes in the predominantly G class of networks. In order to show the utility of the innermost core computation, we perform the following experiment: instead of predicting the high degree nodes from within the innermost core as the high central nodes we directly predict the globally highest degree nodes agnostic of which core they belong to. This results in drastically poorer $F1$-score values as noted in Table 4.5.

4.5.3 Comparison with baselines

In this section we compare our method with three existing baselines taken from [89]. Here the authors estimate the centrality scores of nodes at a future time step t using r previous centrality scores. However, these baselines require that all the nodes be present at all time points. Note that this is not the case for our datasets. Hence to give full advantage to the baseline models, we choose $r >> 1$ so that we can obtain a non-zero average centrality value for each node in the network (even though it does not appear at all time points). For readability we briefly describe the three baselines below. For the first baseline – Uniform – the authors estimate centrality of a node at time step t by taking an uniform average of the node's centrality in r previous time steps. In the second (W1) and third (W2) baselines the

centrality of a node at time step t are calculated as weighted averages of the r time steps. For W1, the weights of the previous r centrality values go as $(\frac{1}{d})$ where d is the distance of the prediction point t from the previous time point being considered. For W2, the weights of the previous r centrality values go as $(\frac{1}{\sqrt{d}})$ where d is again the distance of the prediction point t from the previous time point being considered. Note that the maximum value of d is r. Note that all these schemes are very compute intensive, unlike our approach, as they require explicit computation of the centrality values in all the r earlier time steps for prediction in the current time step t. In Tables 4.5 and 4.6 we compare the $F1$-scores of our prediction algorithm with the above baselines for closeness and betweenness respectively. We outperform the baselines in almost all cases.

Table 4.5: $F1$-score results for the predicted top-10 central nodes for closeness averaged over multiple temporal snapshots. Mean (μ) and std. dev. (σ) of results are reported and the obtained results are compared against the existing baselines. The value of r is set to 20. The best results are marked in **bold**.

Networks	{$F1$-score Close (μ, σ) (Our method: High degree from core)}	{$F1$-score Close (μ, σ) (Global degree)}	{$F1$-score Close (μ, σ) (Uniform)}	{$F1$-score Close (μ, σ) (W1)}	{$F1$-score Close (μ, σ) (W2)}
AS	**0.81,0.08**	0.24,0.08	0.74,0.05	0.75,0.09	0.75,0.08
CA	**0.77,0.08**	0.1,0.03	0.73,0.08	0.74,0.07	0.74,0.09
HT	**0.42,0.3**	0.12,0.08	0.19,0.03	0.18,0.12	0.14,0.09
HP	**0.46,0.29**	0.07,0.08	0.23,0.14	0.0,0.0	0.0,0.0
SO	**0.39,0.22**	0.26,0.31	0.21,0.15	0.21,0.15	0.23,0.15
WT	**0.31,0.19**	0.15,0.11	0.25,0.16	0.26,0.15	0.26,0.15
FW	**0.24,0.19**	0.21,0.16	0.13,0.14	0.07,0.11	0.07,0.11
SU	0.02,0.21	0.007,0.19	**0.04,0.024**	0.02,0.04	0.02,0.04

Table 4.6: $F1$-score results for the predicted top-10 central nodes for betweenness averaged over multiple temporal snapshots. Mean (μ) and std. dev. (σ) of results are reported and the obtained results are compared against existing baselines. The value of r is set to 20. The best results are marked in **bold**.

Networks	{$F1$-score Bets (μ, σ) (Our method:High degree from core)}	{$F1$-score Bets (μ, σ) (Global degree)}	{$F1$-score Bets (μ, σ) (Uniform)}	{$F1$-score Bets (μ, σ) (W1)}	{$F1$-score Bets (μ, σ) (W2)}
AS	**0.72,0.08**	0.24,0.08	0.69,0.05	0.7,0.09	0.68,0.08
CA	**0.64,0.08**	0.1,0.03	0.59,0.08	0.61,0.07	0.61,0.09
HT	**0.46,0.3**	0.12,0.08	0.08,0.03	0.14,0.12	0.14,0.09
HP	**0.52,0.29**	0.07,0.08	0.16,0.18	0.06,0.11	0.06,0.11
SO	**0.39,0.22**	0.26,0.31	0.32,0.19	0.21,0.15	0.28,0.15
WT	**0.31,0.19**	0.15,0.11	0.29,0.16	0.26,0.15	0.3,0.15
FW	**0.2,0.19**	0.21,0.16	0.13,0.14	0.07,0.11	0.13,0.11
SU	**0.18,0.21**	0.07,0.19	0.14,0.024	0.02,0.04	0.11,0.17

4.5.4 Experimental evaluation of computational complexity

We also estimate the amount of time required to actually find the high central nodes (using traditional shortest-path based techniques) for the real world networks where the prediction accuracy ($F1$-score) is high. We report these results in Table 4.7, which show that the time

required to predict node ids with high centrality (inclusive of the time required to compute the classification parameters) is substantially low compared to actually finding them.

Table 4.7: Experimental evaluation of the computational complexity for prediction and parameter calculation. Evaluation was done on a workstation desktop running 64bit Ubuntu 14.04 with Intel Xeon E312xx family processor and 32GB RAM.

Network	Time (secs)	Time (secs)	Time (secs)
	Traditional method	Classification parameters	Prediction method
AS	34.19	4.23	1
CA	1946.2	5.67	2
HP	271	4.89	1
HT	84	3.11	1

4.5.5 Results on synthetic networks

We further evaluate our classification and prediction frameworks on a set of 20 synthetic networks of different sizes, different core structures and different number of temporal snapshots generated using two network generator tools – Musketeer [65] and Dancer [14]. While $N5, N6, N7, N12$ have been generated by the Dancer tool, the remaining networks have been generated by the Musketeer tool. For all the 20 new networks, we report the number of temporal snapshots and the number of nodes and edges in the largest snapshot in Table 4.8. For each network, we first obtain the cumulative distributions corresponding to all the four parameters (EF, CFX, ED, CV) by estimating these values for a certain number of time steps. Since we already have the two well-defined clusters from the 8 real datasets and their corresponding centroids, we simply calculate the distance of the distribution using D-statistic from the two centroids and assign the network to the cluster corresponding to the nearest centroid (see Rocchio classification [10]). This scheme enables us to avoid re-clustering every time a new dataset is available.

As a following step, likewise real world networks, here also we predict the overlap values and the exact nodes. We report the corresponding error percentages and $F1$-scores in Table 4.8. Once again, as observed earlier, the results are best for the $GGGG$ class of networks.

Table 4.8: Results for the test suite of synthetic networks generated by the Dancer tool ($N5$, $N6$, $N7$, $N12$) and the Musketeer tool (remaining networks). Average error in predicted overlap as well as the mean $F1$-score of the predicted top-10 central nodes averaged over multiple temporal snapshots is reported in terms of mean(μ) and std. dev. (σ). Networks are grouped with respect to the class they belong to.

Network Name	Nodes	Edges	Time steps	Bet. overlap (μ, σ) pred. err.	Close. overlap (μ, σ) pred. err.	$F1$-score (μ, σ) Bet.	F1 Score (μ, σ) Close.	Network Category
$N1$	6437	16936	720	14.71, 13.38	17.78,17.31	0.73,0.10	0.82,0.09	$GGGG$
$N2$	4432	8462	720	13.78,12.22	14.67,13.75	0.79,0.09	0.81,0.08	$GGGG$
$N3$	39545	12290	120	24.40,18.07	28.39,20.95	0.78,0.09	0.67,0.11	$GGGG$
$N4$	22871	44566	120	26.22,17.51	27.19,19.09	0.92,0.07	0.75,0.10	$GGGG$
$N5$	2782	170037	50	10.28,6.67	12.6,8.73	0.89,0.08	0.94,0.01	$GGGG$
$N6$	1315	5136	50	10.54,4.62	13.17,6.19	0.54,0.09	0.71,0.13	$GGGB$
$N7$	10316	67949	50	34.11,25.52	15.57,19.97	0.44,0.26	0.82,0.26	$GGGB$
$N8$	25801	301995	120	22.09,24.15	22.26,21.35	0.21,0.21	0.31,0.25	$GGBG$
$N9$	21021	204460	120	18.74,19.08	28.54, 25.72	0.22 ,0.20	0.34,0.29	$GGBG$
$N10$	15662	128463	82	30.45,24.71	41.68,30.40	0.38,0.26	0.45,0.29	$GGBG$
$N11$	12832	91984	82	33.88,24.81	34.97,26.03	0.4,0.28	0.47,0.33	$GGBG$
$N12$	3525	194655	50	21.65,14.25	27.61,10.91	0.58,0.09	0.37,0.12	$GGBB$
$N13$	207177	624796	90	30.09,27.90	43.63,30.33	0.56,0.11	0.45,0.12	$BBBG$
$N14$	164574	438655	90	35.09, 27.90	43.6,30.33	0.41,0.18	0.39,0.12	$BBBG$
$N15$	73499	166774	65	21.82,28.35	24.33,26.28	0.28,0.15	0.31,0.02	$BBBB$
$N16$	55681	111188	65	35.18,30.89	11.90, 29.83	0.33,0.18	0.36,0.19	$BBBB$
$N17$	19591	28431	48	112.34,75.96	119.90,138.66	0.22,0.24	0.28,0.27	$BBBB$
$N18$	15403	20063	48	99.61,70.48	78.68,62.60	0.19,0.23	0.23,0.27	$BBBB$
$N19$	8472	14107	90	52.54,70.68	78.58, 62.60	0.27,0.16	0.13,0.19	$BBBB$
$N20$	6758	9895	90	51.54,70.69	50.59,76.21	0.31,0.19	0.12,0.21	$BBBB$

4.6 Validation

In this section we demonstrate that the behavior of our predicted high centrality vertices is very similar to the actual high centrality vertices in a practical context. We select one representative example from two extreme classes of networks (AS, category $GGGG$ and WT, category $BBBB$) and show that for those which conform well to our hypothesis, the predicted high central vertices and the actual high central vertices behave similarly, while for those which do not conform well, any random selection of vertices behaves similar to the actual high centrality vertices.

4.6.1 Validation for closeness centrality

To validate for the closeness centrality, we obtain the top 10 high central vertices, the top 10 predicted high central vertices and 10 random vertices from the graph G_t, the t^{th} snapshot in the time series. We use the vertices from each set as seeds to disseminate a message in the network. At each iteration a seed vertex u propagates its message to all its neighbors, and these new vertices who just received the message is appended into the seed set. We stop the iteration when all the vertices have received the message. We perform this experiment for all the networks in the time series and report the results in Figure 4.9.

For the *AS network*, the actual top central vertices are the fastest propagator of the message in the network, and this trend remains consistent in all the time steps. Our predicted top central vertices almost always show similar behavior to the actual vertices. The randomly selected vertices, in contrast, take longer to spread the message. For the *WT network*, however the trend from the actual, the predicted, and the random vertices are very similar qualitatively (and quantitatively as well) to one another.

4.6.2 Validation for betweenness centrality

In order to validate for betweenness centrality we posit that since high fraction of shortest paths should pass through the high betweenness vertices, removing them would increase the diameter by a significant margin. We obtain the actual top 10 betweenness centrality vertices, the top 10 predicted vertices and a set of 10 random vertices. In each case, we remove these vertices from the network and calculate the diameter.

For the *As network*, the removal of the actual top 10 betweenness centrality vertices leads to the larger diameter and the size is very similar to that obtained for the predicted high betweenness vertices. Removing random vertices affects the diameter the least and is much lower than the other two above cases. For the *WT network*, the effects on the diameter due to removal of the actual top, the predicted top as well as the random vertices are the same. This indicates that the betweenness centrality values in a network like WT are very uniform across vertices and therefore selecting high betweenness central vertices are of not much

advantage in the practical context.

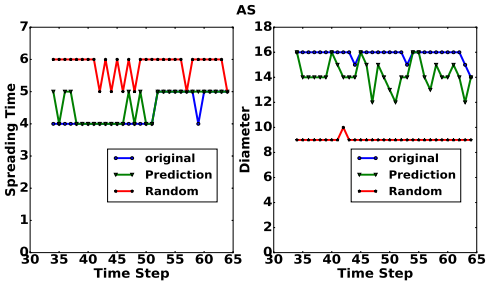


Figure 4.7: AS network

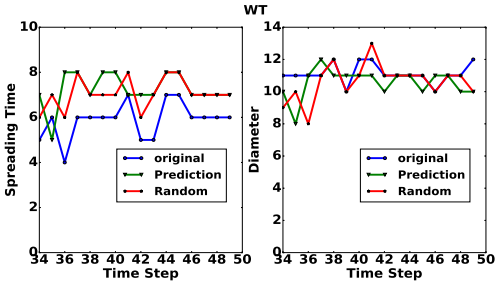


Figure 4.8: WT network

Figure 4.9: The left panel shows validation results for AS network and the right panel for the WT network. Left: Time for spreading a message with high closeness centrality vertices as initial seeds. Right (betweenness): The diameter size after removing high betweenness centrality vertices. Color online.

4.7 Theoretical insights

We introduce the *core connectedness* (CC) property to formalize our hypothesis and to quantitatively distinguish between the networks whose high centrality vertices are in a small and dense innermost core from those where this property does not hold.

Defining core connectedness: Consider a graph G with a core-periphery structure. The shells are consecutively numbered as $S_1, \ldots, S_i, S_{i+1}, \ldots, S_{max}$, where S_1 is the outermost shell and S_{max} is the innermost shell. For ease of expression we will use shell numbers interchangeably with their integer values, i.e. $S_i - S_j$ to denote the difference $i - j$.

We define a path between two specified vertices as a sequence of alternating vertices and edges where no vertex is repeated. The shortest path (distance) is the smallest such sequence. We denote the length of the distance between two vertices v and u as $P_{v \to u}$. If there is no path between v and u then $P_{v \to u} = \infty$. For each pair of vertices v and u we obtain the following two paths; **(i)** The shortest path between v and u, with the constraint that the sequence contains at least one vertex from S_{max}. The length of this path is denoted as $P^{max}_{v \to u}$. **(ii)** The shortest path between v and u, with the constraint that the sequence does not contain any vertex from S_{max}. The length of this path is denoted as $P^{O}_{v \to u}$. Let the

length of the shortest path between v and u, without any constraints be $P^X_{v\to u}$.

Given these definitions at least one of the paths lengths , $P^O_{v\to u}$ or $P^{max}_{v\to u}$ (but perhaps not both) would be equal to $P^X_{v\to u}$.

We define *a core connected (CC) network is one where* $P^{max}_{v\to u} \leq P^O_{v\to u}$ *for* $v, u \in V$, $(v, u) \notin E$ *and* $P^{max}_{v\to u} \neq \infty$.

In other words, in a core connected network, if two non neighboring vertices v and u, have a path through the innermost core then that path is the shortest path between them. An example of a core connected network is given in Figure 4.10.

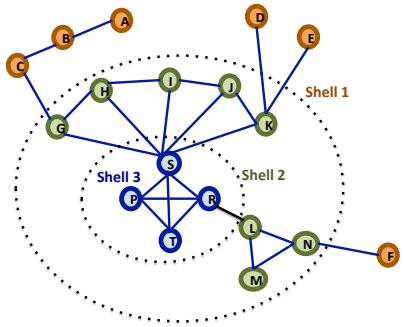


Figure 4.10: Example of a core connected network. (color online) The network has three shells. Lengths of paths between all non-neighboring vertices that pass through shell 3 are also the absolute shortest between them. Example $P^{max}_{C\to E}$=4 while $P^O_{C\to E}$=6. Color Online.

For networks that are core connected, the CC strength of a network can be computed as the number of paths that pass through the core to the total number of paths. It is easy to see that the higher the CC strength, the higher the likelihood that path-based high centrality vertices will be in the innermost core.

Condition for core connectedness We now present the structural properties that facilitate core connectedness. Before we do so, we define the following terms;

- *Inter-core edges:* An edge (v, u), such that $v \in S_i$ and $u \in S_j$ and $i \neq j$. Example; Edges (C, G), (J, S) in Figure 4.10.
- *Intra-core Edges:* An edge (v, u), such that $v \in S_i$ and $u \in S_i$. Example; Edges (G, H), (S, T) in Figure 4.10.

- *Border Vertices:* Vertices that are end points of inter-core edges. Example; Vertices C, G, J, S in Figure 4.10.

- *Maximum distance to border vertices (d_t):* The most number of hops required for a vertex in shell S_t reach a border vertex in the same shell. Example; In Figure 4.10, shell 1 has $d_1 = 2$ for the path from A to C, shell 2 has $d_2 = 1$, for the path from N to L, shell 3 has $d_3 = 1$, for the path from S to T.

- *Diameter of shell (f_t):* The diameter of the subgraph induced by the components in shell S_t. If there are multiple components the longest diameter is selected. Example; shell 1 has $f_1 = 2$ for the path from A to C, shell 2 has $f_2 = 4$, for the path from G to K, shell 3 has $f_3 = 1$, for the path from S to T.

- *Minimum distance in shell (g_t):* The smallest distance between two non-neighboring vertices in the subgraph induced by the components in shell S_t. If there are multiple components the smallest path is selected. The smallest value of g_t among all shells is g_{low}. Example; shell 1 has $g_1 = 2$ for the path from A to C, shell 2 has $g_2 = 2$, for the path from G to I, shell 3 has $g_3 = 0$, as there are no non-neighboring vertices in this shell.

- *Inter-edges to innermost shell (I_t):* The fewest number of inter-edges traversed to go from shell S_t to S_{max}. The endpoint vertices in each shell is selected to minimize this values. The largest value of I_t over all shells is denoted as I_{high}.

Given these parameters, **we posit that a network is core connected if** $2I_{high}+2\Sigma_{t=i}^{t=max-1}d_t+f_{max} \leq g_{low}$ (Condition C1).

To show how this leads to core connectedness we first observe that the length of the path from vertex v to u comprises of the following factors; *(i)* number of inter-edges traveled, *(ii)* number of hops at each shell to border vertices and *(iii)* number of hops at each shell to non-border vertices.

Upper bound on paths through S_{max}: We now consider the length of the path from v to u that passes through S_{max}. This is given by the length of the path from v to any vertex in S_{max} plus the length of the path from u to any vertex in S_{max}, and the length of the path connecting these two vertices that were reached in S_{max}. For example, the path from A

to F in Figure 4.10 comprises of the path from A to S, the path from F to R, and the path connecting S to R.

Note, given the properties of the graphs under consideration, while going from an outer core to the innermost core we only need to pass each intermediate shell at most once. Moreover in the intermediate the shells we will only travel to border vertices.

Thus, given $v \in S_i$ and $u \in S_j$, $P^{max}_{v \rightsquigarrow u} = I_i + \Sigma^{t=max-1}_{t=i} x^i_t d_t + I_j + \Sigma^{t=max-1}_{t=j} x^j_t d_t + y_{max} f_{max}$, where x^i_t, x^j_t and y_{max} are real numbers denoting the fraction of the d_t and f_{max} traveled in each shell.

The upperbound on $P^{max}_{v \rightsquigarrow u}$ is obtained by taking the largest inter-edges to S_{max} and setting i and j to 0 and x^i_t, x^j_t and y_{max} to 1. Therefore $P^{max}_{v \rightsquigarrow u} \leq 2I_{high} + 2\Sigma^{t=max-1}_{t=0} d_t + f_{max}$.

Lower bound on paths not through S_{max}: For paths not through S_{max}, the path from v to u can visit any of the shells, except S_{max}, multiple times. Moreover, we can combine the paths traveled due to bridge vertices and other intra-edges to write $P^{O}_{v \rightsquigarrow u} = Q + \Sigma^{t=max-1}_{t=0} y_t g_t$.

Here Q is the number of inter-edges traversed, and $y_t \geq 1$ is a real number denoting the distance traveled in each shell in terms of g_t. If no intra-edges are travelled in shell S_t then $y_t = 0$, otherwise $y_t \geq 1$.

Since a path from v, u passes through S_{max}, therefore at least one intra-edge will be included in the path from v, u that does not pass through S_{max}. Thus at least one $y_t \geq 1$. Let this be in shell S_x Let g_{low} be the lowest g_t among all S_t except for the innermost shell. Therefore $P^{O}_{v \rightsquigarrow u} \geq y_x g_x \geq g_x \geq g_{low}$.

For the graph to be core connected $P^{max}_{v \rightsquigarrow u} \leq P^{O}_{v \rightsquigarrow u}$. Using the upper and lower bounds of the terms we see that this condition will be satisfied if $2I_{high} + 2\Sigma^{t=max-1}_{t=0} d_t + f_{max} \leq g_{low}$, thus proving our earlier statement.

This constraint is however very strict as it encompasses the entire network, and thus includes vertex pairs with no paths through S_{max}, such as A and C in Figure 4.10. A less strict, but equally effective, constraint for core connectedness can be obtained if we con-

sider each pairs of components, instead of the complete shells. For vertices in two components C_i and C_j to maintain $P^{max}_{v \rightsquigarrow u} \leq P^{O}_{v \rightsquigarrow u}$ the following has to hold $I_{max,i} + \Sigma_{t=i}^{t=max^C-1} d_t + I_{max,j} + \Sigma_{t=j}^{t=max^C-1} d_t + f^{i,j}_{max} \leq min(g_i, g_j)$. Here the parameters are defined in terms of the components instead of the shells, max^C is the maximum number of components traversed, and $f^{i,j}_{max}$ is the diameter of the subgraph induced by border vertices in S_{max} that connect to vertices in C_i and C_j.

In Figure 4.10 we can see that the larger component of shell 2 maintains core connectedness as per the relaxed condition. Here $I_{3,2} = 1$, $d_2 = 0$ for the larger component, $f_3^{2,2} = 0$, since for the larger component the only border vertex is S. Adding the terms we get the left hand side as 2. The right hand side, smallest distance between two non-neighboring vertices is also 2. By checking the shortest paths between the vertices in shell 2 we can see that the core connectedness property is indeed maintained.

4.8 Summary of the chapter

To summarize, our **key contributions** are as follows:

- Propose a hypothesis that in many real world time-varying networks, a majority of the highly central vertices reside in the innermost core of the network. Subsequently, develop a set of novel heuristics to classify networks based on the extent to which the highly central vertices are part of the innermost core. To perform the classification, these heuristics do not require the explicit computation of the high centrality vertices as the network varies over time.
- Develop an efficient algorithm to precisely predict the high betweenness and closeness centrality vertices. To the best of our knowledge, this is *the first algorithm to precisely predict the vertices* and not simply the average centrality value of the vertices in the network.
- Validate our results in practical application scenarios, namely message spreading and increasing the diameter, to show that the effect of our predicted vertices is similar to the actual ones. For networks, where our prediction accuracy is low, we show that

even a random selection of vertices can produce same results as the actual high closeness/ betweenness centrality vertices. This indicates that for such networks, there are no significantly high closeness/betweenness centrality vertices. Thus, in these cases, prediction of high closeness/betweenness centrality vertices is of no practical use.

- Finally, we also outline a brief theoretical sketch demonstrating why our method works.

Chapter 5

Learning representations from core periphery structure

5.1 Introduction

The conventional paradigm of handcrafted feature engineering to generate node representations in networks has been largely overhauled due to the advances in techniques which automatically discover and map a node's structural properties into a latent space. These techniques are useful because manual feature engineering requires extensive domain knowledge as well as tedious exploration of structural properties such as degree, centrality, clustering coefficient etc. Without loss of generality, representation learning encompasses the task of transforming a graph $G(V, E)$ from $V \rightarrow \mathbb{I}^{|V|\times|V|}$ to the mapping $V \rightarrow \mathbb{R}^{|V|\times d}$ with the constraint $d << |V|$.

5.2 Core2vec: Learning node representations based on network core information

In this section we first outline the limitations of the current random walk based techniques and then outline our proposal.

5.2.1 Random walk based techniques

This problem is efficiently solved by applying a *skip-gram model with negative sampling* [124], which is a celebrated technique for learning meaningful vector representations for words. To represent a target word, nearby co-occurring words in the sentence are considered as context words. Adapting this framework for graphs, there have been several works such as [139, 167] which learn social representations of a graph's vertices, by learning from neighbor nodes generated from short random walks. *These walk sequences act as proxy for context words in a sentence*. Apart from capturing local proximity, global information is also captured [63] through generation of flexible contexts by parameterized random walks.

5.2.2 Limitations of random walk based techniques

One of the key drawbacks in these works is the assumption that the context nodes can be always efficiently generated by walk sequences from a source node thus building a sample set appropriately representing the structural and the functional properties of the source node. This perhaps is a fair assumption in social networks which are inherently assortative. However, this might not be applicable for several classes of networks such as biological (protein interaction), technological (router-router interaction), and semantic (e.g., Wordnet) networks which are disassortative.

5.2.3 Our proposal

We propose a solution (see section 5.2.6) to the above problem by developing an algorithmic framework, *core2vec* which utilizes intermediate-scale structure of the network, i.e., the core periphery structure, for learning the feature representation of a node. A core-periphery structure in its simplest form refers to a partition of a network into two groups of nodes called core and periphery, where core nodes are densely interconnected (i.e., adjacent), and peripheral nodes are adjacent to the core nodes but not to other peripheral nodes. Many techniques exist [12, 144] which attempt to discover multiple nested cores in the network. This partition of the network into nested cores of disjoint layers represent separate structural/functional properties of nodes in the network.

We leverage this nested “onion like structure" in real world complex networks, to develop a flexible biased random walk which seeks similar core nodes as context nodes for a source vertex. More specifically we develop a strategy to guide a random walk sequence to identify similar core nodes both in close proximity as well as distant neighborhood. We further design an objective function, which computes the average likelihood of predicting the source node given the set of context nodes, which we obtain through our exploration procedure. This objective function can be optimized efficiently using stochastic gradient descent and consequently leads us to our optimal set of feature vectors. Core information for a node can be computed efficiently in $O(|E|)$ [12] and nodes with similar core ids have equivalent connectivity profiles over the entire network. We perform experiments with real world networks to analyze the performance of our scheme (see section 5.2.8). These experiments show that our scheme brings nodes with similar core ids closer (*closeness*) as well as separates nodes with different core ids (*separability*) farther in the vector space compared to state-of-the-art methods like node2vec [63], DeepWalk [139] and LINE [167], thus establishing the necessity of our approach.

5.2.4 Objective function

We exploit the idea of a skip-gram model with negative sampling, which is popular for language modeling, in the context of graphs. Given a set of vertices $\mathbb{C}_v$ our task is to maximize

$P(\mathbb{C}_v|v)$ where $\mathbb{C}_v$ comprises the context vertices of v.

To complete our objective we formulate the optimization problem as

$$\underset{max}{f} = \sum_{v \in V} P(\mathbb{C}_v|v) = \sum_{v \in V} \prod_{\mathbb{C}_{v_i} \in \mathbb{C}_v} P(\mathbb{C}_{v_i}|v) \tag{5.1}$$

where $\mathbb{C}_{v_i}$ is a context node belonging to the context set of v. $P(\mathbb{C}_{v_i}|v)$ can be computed as

$$P(\mathbb{C}_{v_i}|v) = \frac{\exp(v^w.\mathbb{C}^w_{v_i})}{\sum_{v \in V} \exp(v^w.\mathbb{C}^w_{v_i})} \tag{5.2}$$

Here $v^w, \mathbb{C}^w_{v_i}$ denotes the vector representation of the node v and context node $\mathbb{C}_{v_i}$. Since this formulation is difficult to optimize directly we introduce negative sampling analogous to word2vec [124].

5.2.5 Context nodes

We generate context nodes for each individual source node by performing L random walks of fixed walk length (l)[1] with the source node as the starting vertex. Similar core nodes can sometimes be adjacent to the source node or they can be separated by multiple hops. Hence in our exploration strategy we assume roles of both forms of extreme sampling strategies – the breath-first sampling as well as the depth-first sampling [63].

5.2.6 Methodology

Consider a random surfer which has started from node i and is currently at node j, where it is not necessary that $(i, j) \in E$. The decision for the next destination (k) for the random walk given that $(j, k) \in E$ is given by $p_{i,j,k} = \frac{\pi_{ijk} * w_{jk}}{Z}$ where π_{ijk} denotes the unnormalized transition probability, Z is the normalization constant and $w_{j,k}$ is the weight of edge (j, k). $w_{j,k}$ is 1 in case the graph is unweighted. The unnormalized transition probability for our approach is given below.

[1] In our experiments we have set $l = 40$ and $L = 10$.

$$\pi_{i,j,k} = \begin{cases} \frac{1}{(|c_i - c_j|+1)*\mathcal{D}*\lambda} & (j,k) \in E, i = k; \\ 1 & (j,k) \in E, i \neq k; \\ \frac{1}{(|c_i - c_j|+1)*\mathcal{D}*\gamma} & (j,k) \in E, i \neq k, (k,i) \notin E \\ 0, & \text{otherwise} \end{cases}$$

Here $p_{i,j,k}$ is the probability of the random surfer starting from vertex i, currently at vertex j to transition to vertex k. c_j, c_k signifies the core id for vertex j, k respectively. λ and γ can be tuned for the purpose of exploring in the close neighborhood of the source node or traverse distant neighborhood of the source node. The pseudocode for *core2vec* is presented in Algorithm 4.

Hence λ refers to the propensity to sample from near nodes from the source vertex. γ refers to the propensity to sample distant nodes from the source vertex. $\mathcal{D}$ is the penalty parameter which penalizes a random jump of large core difference. We add 1 to the core difference to eliminate the chance of getting 0, in such cases where neighbors belong to the same core. The pseudocode for *core2vec* is presented in Algorithm 4. In procedure kCore (see Algorithm 5) we demonstrate the standard $k - core$ decomposition algorithm [12] which partitions the nodes in the graph into disjoint sets of cores. We utilize this core based partition along with hyperparameters $\gamma, \lambda, \mathcal{D}$ to obtain the probability transition matrix $\mathcal{P}$ using the formulation we described earlier in this section. For all nodes in the network we obtain a neighborhood set using the genWalks (see Algorithm 6) procedure. This neighborhood set we finally feed into the *SGD* module for optimization and generation of the features.

Algorithm 4: Procedure: LearnFeatures

Input: $G = (V, E, W)$, dimensions d, walks per node L, walk length l, context size c, exploration parameters λ, γ, penalty parameter $\mathcal{D}$, probability transition matrix $\mathcal{P}$

```
core_dict ← kCore(G) ;
P ← PreprocessProb(G, λ, γ, D, core_dict) ;
G' ← (V, E, P) ;
for all nodes u ∈ V do
    walks to empty ;
    walks ← genWalks(G', u, l, L) ;
f ← SGD(c, d, walks) ;
return f ;
```

Algorithm 5: Procedure: kCore

Input: $Graph, G(V, E)$
Output: $C[K], K = |V|$
$k \leftarrow 1$;
while $|V| \geq 0$ **do**
  while *true* **do**
    remove all vertices with degree $\leq k$;
    until all remaining vertices have degree $\geq k$;
    $\forall$ vertex (v) removed , $C[v] \leftarrow k$;
  $k \leftarrow k + 1$;
return (C);

Algorithm 6: Procedure: genWalks

Input: $G'(V, E, \mathcal{P})$, *start node* (u), *walk length* (l), *total walk* (L)
Output: $walk$
$walk \leftarrow u$;
$walks \leftarrow \{\}$;
for $num\_walks = 1$ ***to*** L **do**
  for $walk\_iter = 1$ ***to*** l **do**
    $curr = walk[-1]$;
    $\mathcal{N}_{curr} \leftarrow Neigbours\_set(curr, G)$;
    $s \leftarrow Sample(\mathcal{N}_{curr}, \mathcal{P})$;
    $walk \leftarrow s$;
  $walks \leftarrow walk$;
return $walks$

5.2.7 Dataset

Network data used for experiments

We use two very well-known network datasets – **Les Miserables (Lemis)** and **Jazz musicians (Jazz)** as benchmarks to carry out our experiments to show the efficacy of our approach.

Les Miserables (Lemis): This has been taken from [95].

Jazz musicians (Jazz): This dataset has been collected by Gleiser et. al [61].

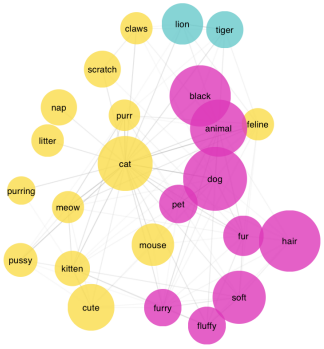


Figure 5.1: A snapshot of a few words obtained from the SWOW word association network.

Dataset for validation

Training data: We use the English word association data collected from the two notable crowd-sourcing efforts – (i) University of South Florida word association project (USF) and (ii) small world of word project (SWOW)[2]. In each case, a group of participants were given a cue word and asked to report the first few words that come to their mind in response to the cue. According to cognitive theories [140, 165] the cue word acts as a stimulus and the responses map how pairs of words are associated in a participant's brain. Hence word associations reveal mental representations of lexicons which are often not captured from textual input because natural language follows predefined syntax. Normative forms of words were reported by each project. Details about the data can be obtained from [43, 131].

- *USF*: In this project 300 participants where each given 60 cue words and asked to produce two related words for each cue word in the order of primary association and secondary association. Word network constructed from this graph contains 10590 unique words and 63788 word pairs (i.e., edges) with an average degree of 12.02.
- *SWOW*: In this project data was collected from $85,496$ participants, where each participant was given 15–20 cue words and asked to report three responses as primary association, secondary association and tertiary association. The word network con-

[2]http://www.smallworldofwords.com/new/visualize/

structed from this dataset comprises $39,026$ unique words and 400622 word pairs (i.e., edges) with average degree of 20.53. A representative snapshot of a small part of this network is shown in Figure 5.1.

Test data: We use three datasets from where we obtain the ground-truth similarity between word pairs. These are: (i) Mturk-771 [67], (ii) WordSim353 [53] and (iii) SimLex-999 [74]. Mturk-771 and WordSim353 score words on both similarity as well as relatedness. However SimLex-999 scores words only on the basis of high semantic similarity.

- *The Mturk-771*: This is a collection of 771 English word pairs along with human-assigned relatedness judgement [67].
- *WordSim353*: WordSim353 is a collection of 353 word pairs whose similarity is assessed and scored by 29 volunteers. This data has been collected by [53].
- *SimLex-999*: This dataset measures similarity, rather than relatedness of 999 word pairs [76].

Hence words such as “coffee", “cup" will have lower scores in SimLex-999 compared to the others because even though they are used in the same context, their meanings are very dissimilar.

5.2.8 Experiments

We establish the necessity of our approach based on the following two metrics – (i) closeness and (ii) separability.

Closeness $(\mathcal{C})$

This metric estimates the cohesiveness of a core’s nodes around its *centroid* (i.e., the mean of all the vectors corresponding to the nodes within a core). Closeness is calculated as the average of cosine similarities of all nodes in a core with that of the core’s centroid. The higher the value of closeness the more compact the core is.

Separability $(\mathcal{S})$

Separability determines the overall separation among the different distinct cores. This is calculated as the average Euclidean distance between pairwise core centroids. The higher the value of separability the more well-separated the cores are.

The different methods are compared in Table 5.1. The closeness and separability increases by tuning the penalty parameter $(\mathcal{D})$ and in both networks we obtain better scores compared to different naïve random walk approaches like node2vec, DeepWalk and LINE.

5.2.9 Need for core2vec

We conduct experiments on two benchmark real world graphs Lemis and Jazz (see section 5.2.7). For both the networks we run the *core2vec* algorithm which learns node representations in latent dimensions. We further transform the embedding onto the 2D plane using PCA. We report the 2D plots of the embeddings obtained in Figures 5.2 and 5.3. Each color denotes nodes of a particular core number. The objective here is to get nodes of a core to form dense and well-separated clusters.

Table 5.1: The values of $\mathcal{C}, \mathcal{S}$ for different methods. In case of core2vec, $\mathcal{D} = 3.5$, $\lambda = 0.35$ and $\gamma = 2.5$.

Algorithm	Les Miserables network	Jazz musicians' network
core2vec	$\mathbf{0.124}, \mathbf{0.414}$	$\mathbf{0.543}, \mathbf{0.404}$
node2vec	0.097, 0.357	0.422, 0.276
DeepWalk	0.076, 0.311	0.317, 0.239
LINE	0.085, 0.345	0.329, 0.255

Results with no core information

The topmost panel in Figure 5.2 shows the results obtained for the Jazz network by setting the core difference $|c_i - c_j|$ to 0 and $\mathcal{D} = 1$. This essentially converts the resulting exploration strategy to a biased random walk similar to node2vec [63]. Our results show that using such walks without core information does not bring similar core nodes together. This observation is also valid for the Lemis network as is evident from the topmost panel in Figure 5.3.

Results with core information in place

Incorporating the core information however results in similar embeddings for similar core nodes (see the last three panels of Figures 5.2 and 5.3). The closeness and separability increases by tuning the penalty parameter $(\mathcal{D})$ and in both networks we obtain better scores compared to different naïve random walk approaches like node2vec, DeepWalk and LINE. The different methods are compared in Table 5.1. However, note that if one keeps increasing $\mathcal{D}$ the closeness and separability does not keep indefinitely increasing. The best values of closeness and separability are obtained using core2vec for $\mathcal{D} = 3.5$ thus showing its superiority over the other methods in obtaining core based embeddings. In both the networks we set $\lambda = 0.35$ and $\gamma = 2.5$. For further discussion on selection of the hyperparameters see section 5.2.12.

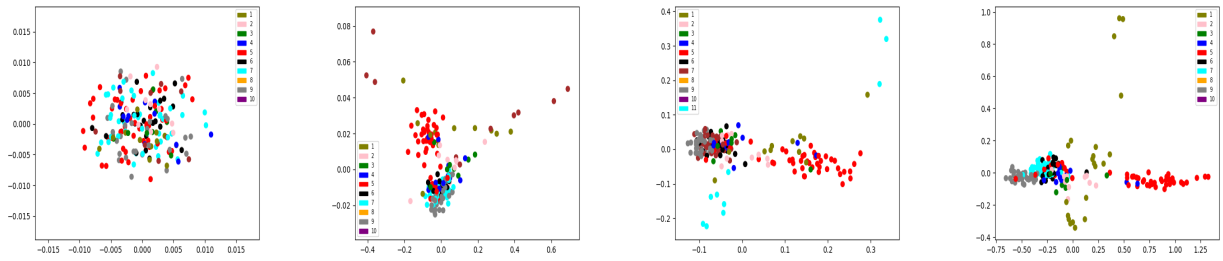


Figure 5.2: Jazz network with increasing $\mathcal{D}$ and core separation. Figures, left to right in that order, were generated with $\mathcal{D}$ values of $1, 1.5, 3.5, 4.5$ respectively.

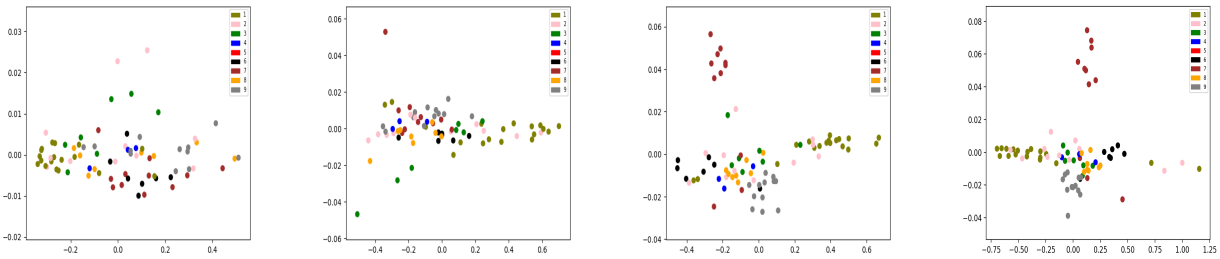


Figure 5.3: Les Miserables network with increasing D and core separation. Figures, left to right in that order, were generated with $\mathcal{D}$ values of $1, 1.5, 2.5, 3.5$ respectively.

5.2.10 Validation

Word association networks

Linguists and cognitive psychologists report [13, 43, 76] that by the time children are 4 years old they hear approximately 10 to 50 million words which only increases manifold as their age progresses. The ability of humans to learn and recall such massive information is through associations. This claim is supported by the finding that word pairs that are semantically associated but mean different things, such as “sky", “blue" or “banana", “yellow" activate the same regions of the human brain. Networks constructed from sentimentally aligned words which have similar associations or relatedness, have dense cores of highly connected words also known as kernel lexicons [33, 82] linked with a relatively sparse periphery.

Outline of the validation framework

We hypothesize that for word association networks with well-defined core-periphery structures, the embeddings obtained from core2vec should be more representative than the state-of-the-art methods like node2vec, DeepWalk and LINE.

In order to establish the above hypothesis we first obtain embeddings of nodes for each of the two word association networks introduced in section 5.2.7. We obtain the embeddings using core2vec as well as the other baselines – node2vec, DeepWalk and LINE.

Next we consider three ground-truth datasets – SimLex-999, WordSim-353 and Mturk-771 that contain a set of word pairs and their similarity/relatedness scores. We rank these word pairs based on these scores. In parallel, we obtain the similarities of exactly these word pairs by estimating the cosine similarities of their corresponding embeddings obtained from the word association networks. We again rank these word pairs based on these cosine similarities. Finally, we estimate the Spearman's rank correlation coefficient between the two rankings (one from the ground-truth similarities and the other from the embedding similarities).

5.2.11 Results

The key results for the two different association networks - SWOW and USF – are shown in Tables 5.2 and 5.3 respectively. The correlation values in the tables indicate that core2vec outperforms all the baselines. The p values are further noted to demonstrate that our observations are significant. Depending from where the ground-truth similarities are drawn, in some cases we even obtain an improvement as high as **46%**. An interesting point is that the best benefit of core2vec is obtained when the ground-truth similarities are drawn from the SimLex dataset. This shows that core2vec is able to better capture strong semantic similarities in comparison to mere relatedness.

To further understand our results, we plot in Figure 5.4, the embeddings learned by *node2vec* (top panel) and *core2vec* (bottom panel) for the same set of words projected on a 2D plane (using PCA). Figure 5.4 clearly shows that words with similar meanings or words which usually have more similar contexts are noticeably clustered better in case of *core2vec*.

5.2.12 Hyperparameters

Our framework has three hyper-parameters – λ, γ and $\mathcal{D}$. Low values of λ will restrict exploration strategy preferentially within close proximity. Low values of γ will result the random walk sample neighbors from distant hops. The penalty parameter $\mathcal{D}$ penalizes random walks of high core difference. After extensive experimentation we observe that the hyper-

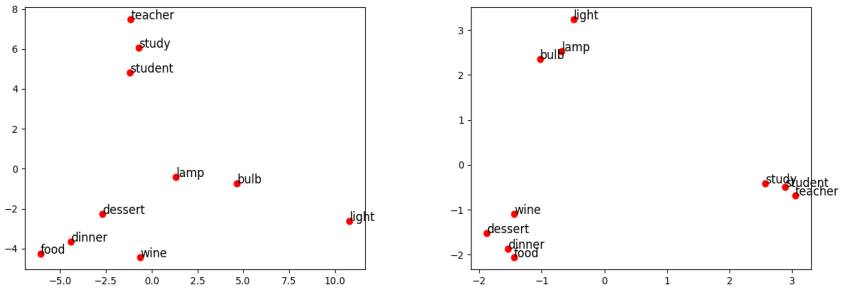


Figure 5.4: core2vec brings semantically similar words closer in the vector space. The values of $\mathcal{D}$, λ and γ are 3.5, 0.3 and 3 respectively.

Table 5.2: Results (Spearman's correlation coefficient, p value of significance) for SWOW word association network. The values of $\mathcal{D}$, λ and γ are 3.5, 0.3 and 3 respectively.

Algorithm	SimLex	WordSim353	Amazon MTurk
core2vec	$\mathbf{0.548, 3.34e^{-79}}$	$\mathbf{0.654, 2.49e^{-42}}$	$\mathbf{0.692, 1.33e^{-109}}$
node2vec	$0.467, 2.00e^{-55}$	$0.640, 3.72e^{-40}$	$0.664, 3.05e^{-98}$
DeepWalk	$0.444, 9.80e^{-50}$	$0.639, 5.86e^{-40}$	$0.653, 3.49e^{-94}$
LINE	$0.449, 9.53e^{-51}$	$0.635, 2.14e^{-39}$	$0.571, 3.01e^{-67}$

parameter values that work best for a network is dependent on the structure of the network being considered. However, increasing $\mathcal{D}$ does not indefinitely increase closeness and separability. A systematic grid search allows us to identify the best choice for each network.

Further, note that higher values of the walk length (l) and number of walks (L) usually yield better results. Higher values of walk length and number of walks increases the overall sampling budget for learning representations [63, 139]. Random walks over graphs are proxy for context size in sentences. It has been shown by Mikolov et. al. [123], in their work on generating word2vec that increasing context size helps in getting richer representations. Since core2vec utilizes the same framework, we speculate that increasing sampling budget can yield better results. However this comes at the expense of computational budget and

Table 5.3: Results (Spearman's correlation coefficient, p value of significance) for USF word association network. The values of $\mathcal{D}$, λ and γ are 3.5, 0.3 and 3 respectively.

Algorithm	SimLex	WordSim353	Amazon MTurk
core2vec	$\mathbf{0.425, 2.11e^{-39}}$	$\mathbf{0.476, 5.29e^{-32}}$	$\mathbf{0.621, 1.33e^{-52}}$
node2vec	0.136, $2.00e^{-15}$	0.439, $4.52e^{-30}$	0.593, $7.35e^{-48}$
DeepWalk	0.116, $9.80e^{-20}$	0.429, $2.66e^{-25}$	0.604, $1.79e^{-48}$
LINE	0.111, $9.53e^{-12}$	0.424, $5.94e^{-24}$	0.598, $9.26e^{-47}$

we get diminishing returns.

5.2.13 Scaling experiments

Here we attempt to empirically test how well our model scales with respect to the number of nodes in the graph. We note that the theoretical time complexity of the kCore procedure is linear in the number of edges (i.e., $O|E|$) which would make the overall complexity of our model at least $O|V^2|$. However, since most large graphs under study are sparse our algorithm completes in sub-quadratic time in practice.

We consider 6 Erdős-Rényi random networks. Probability of edge formation is set to $\frac{\hat{d}}{N}$ where $\hat{d}$ is the average degree and is set to 30. N is the number of nodes which varies assuming values like 10, 100, 300, 1000, 3000, 30000. Note that the logarithms of these numbers increase from 1 to 4.5 linearly. We record the logarithm of the time taken by our algorithm to run with these networks as inputs and plot the result in Figure 5.5. The plot is close to linear with slope one indicating that the time taken by core2vec is mostly linear in the number of nodes.

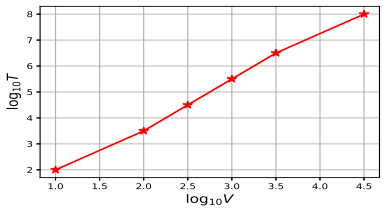


Figure 5.5: Logarithm of the time taken by core2vec vs $log_{10}|V|$.

5.2.14 Discussion

The main difference between core2vec and existing approaches is that the latter approaches are agnostic to the inherent network hierarchy. Core2vec is however explicitly conditioned on the organization of the nodes in the network. More specifically the random walk decisions taken at each step in core2vec is conditioned on the core membership of the current node and its neighbor. Hence neighborhood multiset is not likely to contain members with very large core distance from the starting node. This approach is based on the assumption that similar core nodes have more in common compared to distant core nodes. We speculate that core2vec will work well in networks which have hierarchical structure. It should work comparatively with baselines in case of networks without hierarchies such as regular networks because without core based guidance it would eventually resort to bfs/dfs based exploration for collecting neighborhood.

5.3 Detecting high central nodes

In this section, we present a network representation learning driven approach for detecting influential nodes in the network. While the steps presented in section 3.4 can locate influential nodes, i.e., high closeness and betweenness nodes in the network, in practice, the process of community identification followed by clique percolation is very expensive and does not scale with the size of the graph[3].

A more feasible method would be to apply distant supervision, where a machine learning

[3]Specifically, the clique percolation algorithm takes an exorbitantly large amount of time for all those communities that embed large size cliques.

model is trained on auxiliary tasks based on noisy labels that can be generated by some heuristic driven domain knowledge. Once the model is trained on automatically generated noisy labelled data, it is applied in target task where labelled data is difficult to obtain. Such approaches have shown promising results in natural language processing domain especially in tasks such as relation extraction [125], sentiment classification [155] and machine translation [154].

We express the task of influential node discovery as a semi-supervised learning problem, as follows. Consider a graph G with V vertices, E edges with $\mathcal{X}$ being a $|V \times d|$ feature matrix such that $v_i \rightarrow \mathbb{R}^d$. Let node labels, here "influential" or "not influential", be available for a subset of nodes, the goal is to predict the labels for the rest of the nodes.

To do this classification, we leverage a variant of graph convolution networks (GCN) [91] . The choice of GCN is motivated by its recent success in solving various graph problems e.g., node classification. As we shall see, this novel technique not only leads to a reasonably good selection of important nodes but also scales well with an almost 11x speedup over the traditional method.

5.3.1 Motivating GCN based approach

It can be argued based on the experimental results in Table 3.3, we can simply use the top-k high (based on degree) core vertices and use them as proxy for high central nodes. Note from our observation in section 3.4, that high central nodes are *spread out in the original network*, but *connected in the core of the meta second order network*. This immediately disqualifies taking the top-k core nodes, since most of them are unlikely to be clustered together at the inner core.

Approximate algorithms for finding centrality are generally based on sampling a few select nodes and then finding the shortest path from these nodes [8, 22, 57, 113, 130]. However, unlike the GCN approach that recursively attempts to find the best nodes by observing the neighbors, neighbors of neighbors etc. of the training data points, the approximate centrality finding approach only considers a few shortest paths and ignores the long-range relationships in a network.

Primer to GCN: Here is a short overview of GCNs. The node representation after a single layer of GCN is defined as

$$\boldsymbol{H} = f\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}})\mathcal{X}\boldsymbol{W}\right) \tag{5.3}$$

$\boldsymbol{W} \in \mathbb{R}^{d\times d}$ are the model parameters, $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ where $\boldsymbol{A}$ is the adjacency matrix and $\tilde{\boldsymbol{D}}_{ii} = \sum_j (\boldsymbol{A} + \boldsymbol{I})_{ij}$. f is an activation function, here we used ReLU, $f(x) = \max(0, x)$. Equation 5.3, for each vertex, and with bias included, is expressed as;

$$\boldsymbol{h}_v = f\left(\sum_{u\in\mathcal{N}(v)} \boldsymbol{W}\boldsymbol{h}_u + \boldsymbol{b}\right), \quad \forall v \in V. \tag{5.4}$$

Here, $\boldsymbol{b} \in \mathbb{R}^d$ denotes bias, $\mathcal{N}(v)$ corresponds to immediate neighbors of v in graph G including v itself and $\boldsymbol{h}_v$ is the obtained representation of node v. Multi-hop dependencies between nodes can be computed using multiple GCN layers. The representation of node v after k^{th} layer is given as

$$\boldsymbol{h}_v^{k+1} = f\left(\sum_{u\in\mathcal{N}(v)} \left(\boldsymbol{W}^k\boldsymbol{h}_u^k + \boldsymbol{b}^k\right)\right), \forall v \in V. \tag{5.5}$$

where, $\boldsymbol{W}^k, \boldsymbol{b}^k$ are the layer specific parameters of GCN.

5.3.2 Modification of vanilla GCN

The available model of GCN assumes that the adjacency matrix encodes the similarity between two nodes, i.e., two nodes are connected if they have overlap in latent characteristics. However, for our problem, adjacency matrix entries may contain a large percentage of edges which do not fulfill this homophily criteria. However in our case, the similarity is based on vertices being in the same shell (i.e., having the same core number), which is a stricter condition than adjacency. Though this is not true for all networks, yet we propose that dense structure within the inner core can act as label data which can be used by a learning algorithm to discover other possible high central nodes in the network. One of the drawbacks in learning algorithms like GCN is that it inherently assumes that the ad-

jacency matrix encodes the similarity between two nodes, i.e., two nodes are connected if they have close functions. To address this issue, we modify the equation 5.3 to equation 5.6 by element wise multiplication with the matrix $\mathcal{C}$ where $c_{ij} = 1$ if the absolute difference between the shell numbers of node $v_i, v_j \leq |\Delta|$. This operation acts as a mask, eliminating edges which connects distant shells and increases sparsity of the network. Sparsity has been shown to improve GCN in multiclass classification [3]; however, to the best of our knowledge this is *the first application of core based graph sparsification for substructure discovery.*

$$\boldsymbol{H} = f\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\left(\tilde{\boldsymbol{A}} \odot \mathcal{C}\right)\tilde{\boldsymbol{D}}^{-\frac{1}{2}})\mathcal{X}\boldsymbol{W}\right) \tag{5.6}$$

5.3.3 Network suite

All the real world networks used in our experiments are available publicly at [2, 101, 146]. Networks N1-N4 were generated using the synthetic network generator Musketeer [65]. A summary of the properties are noted in Table 5.4.

5.3.4 Train and test setup

To train the GCN, we first compute the core numbers of the nodes in the original network. The complexity of this operation is $O(|E|)$. Nodes in the innermost cores are the best and most easily obtainable proxies to an initial set of seeds needed to train the GCN. We use a random set of vertices sampled from within the innermost core as the positive training labels. Equal number of negative examples are selected randomly from the outer shells. The remaining unlabelled nodes are labeled during the training phase. The total labelled data is 20% of the total number of nodes, similar to that used in [91].

We divide the remaining 80% nodes into 10% validation set and 90% test set. We calculate $\mathcal{X}$ using scalable embedding technique node2vec [63]. We use Adam [90] for optimization with learning rate of 0.01 and weight decay of 0.0001. We use a three layer convolution network with the final embedding dimension size of 200. Input dimension size is 512 fol-

Network	**Nodes**	**Edges**	α	$\mu(d_v)$	$\mu(C_lC)$	**LCN**
as1	4333	7948	1.251	4.024	0.227	11
as2	6474	12572	1.235	4.292	0.252	12
caida	26475	53381	1.164	4.032	0.208	22
bio	7393	25569	1.991	6.917	0.011	11
power	4941	6594	2.844	2.669	0.0801	5
sw	994	4645	1.168	9.32	0.34	11
astro	18772	198050	1.842	21.106	0.630	56
condmat	23133	93439	2.237	8.083	0.633	25
hepph	34546	420877	2.053	24.368	0.289	30
hepth	27770	352285	1.775	25.375	0.313	37
cora	23166	89157	2.147	7.697	0.265	13
ytube	13723	76765	1.826	11.187	0.136	25
enron	35692	183831	1.544	10.02	0.496	43
N1	67292	284931	1.944	9.02	0.384	53
N2	54292	196782	1.32	13.9	0.356	39
N3	87392	453734	2.114	4.2	0.296	47
N4	25382	153831	1.844	6.2	0.386	43

Table 5.4: Test suite of networks and their properties. α: power-law exponent, $\mu(d_v)$: average degree, $\mu(C_lC)$: average clustering co-efficient, (LCN): largest core number in the network. The contents in the parenthesis indicate the abbreviations we shall use for the network names in the rest of the chapter.

lowed by the hidden units of size 400, 300 and 200 respectively. The core difference value of $|\Delta|= 6$ worked well through tuning of the validation set. The last hidden layer of the GCN provides a probability of the positive/negative labels for each node. We prepare a rank list of the nodes based on the highest positive confidence values.

5.3.5 Ground-truth nodes

To evaluate our results we compare the rank list obtained from the GCN with the actual seeds across the (scattered) rich clubs. To prepare a list of such ground-truth seeds we randomly sample a set of k nodes from the cliques in the innermost core of the meta network as found in section 3.4. We select the seeds such that all the cliques are well represented. We rank the k nodes based on their degree within their respective cliques. All the nodes in the same clique will have the same degree, thus the same rank position. Nodes from different cliques may have different degrees. We prepare multiple such randomly sampled ground-truth rank lists of seeds.

Results: We use $AP@k$ measure to evaluate the performance of vertex recommendation, which measures the rank accuracy of the recommended vertex list [134]. As shown in section 3.7.1, larger rich clubs lead to more resilience. Seeds from larger rich clubs (or cliques representing them) will have larger degree and thus higher rank.

We set $k = 20$, and compare the rank lists obtained from the GCN and the ground-truth seeds. A case is said to be a match at a particular position of the rank list if the node predicted by the GCN at that position matches directly with the ground-truth seed at that position or to a neighbor of that seed, that belongs to its clique.

The results for $AP@20$ for two different ground-truth rank list of seeds (Set_A, Set_B) are given in Table 5.5[4]. The results are representative and are similar across other sets of ground-truth rank lists (evident from the low standard deviation over the different sets). Our masked variant of GCN is significantly better in discovering seed nodes when compared against GCN in [91] for most of the networks. The induced subgraph of the seed and its neighbors is then used to construct the (scattered) rich clubs.

We also compare the time to predict seed nodes using GCN versus explicitly computing the high centrality nodes in Table 5.6. As can be seen, the time for prediction is substantially lower. This result is representative and holds for all the other networks.

[4]We also experimented with $AP@5$, 10 and 15 which showed similar trends

Network	Kipf GCN		Masked GCN	
	Set_A	Set_B	Set_A	Set_B
as1	0.41 ± 0.01	0.32 ± 0.01	0.64 ± 0.01	0.94 ± 0.01
as2	0.24 ± 0.01	0.29 ± 0.01	0.94 ± 0.01	0.90 ± 0.01
caida	0.24 ± 0.02	0.63 ± 0.01	0.8 ± 0.01	0.88 ± 0.01
bio	0.46 ± 0.05	0.64 ± 0.01	0.85 ± 0.01	0.89 ± 0.01
enron	0.63 ± 0.01	0.48 ± 0.02	0.42 ± 0.02	0.88 ± 0.01
sw	0.1 ± 0.01	0.26 ± 0.08	0.73 ± 0.05	0.92 ± 0.01
N1	0.23 ± 0.01	0.19 ± 0.05	0.33 ± 0.01	0.36 ± 0.02
N2	0.48 ± 0.01	0.42 ± 0.02	0.63 ± 0.02	0.82 ± 0.01
astro	0.35 ± 0.05	0.45 ± 0.05	0.6 ± 0.01	0.67 ± 0.01
condmat	0.2 ± 0.03	0.43 ± 0.1	0.76 ± 0.08	0.88 ± 0.02
cora	0.4 ± 0.02	0.45 ± 0.1	0.78 ± 0.02	0.96 ± 0.01
hepph	0.24 ± 0.09	0.28 ± 0.01	0.7 ± 0.05	0.9 ± 0.05
hepth	0.5 ± 0.04	0.4 ± 0.03	0.6 ± 0.09	0.86 ± 0.04
ytube	0.4 ± 0.02	0.65 ± 0.03	0.85 ± 0.05	0.95 ± 0.05
pow	0.31 ± 0.1	0.46 ± 0.08	0.33 ± 0.1	0.52 ± 0.05
N3	0.1 ± 0.01	0.1 ± 0.01	0.83 ± 0.02	0.65 ± 0.02
N4	0.15 ± 0.01	0.1 ± 0.01	0.54 ± 0.02	0.73 ± 0.02

Table 5.5: $AP@20$ for the prediction of seed vertices for different networks. Lower (Higher) scores in each method are highlighted.

Network	Time (secs) Direct centrality (T)	Time (secs) GCN method	Improvement Ratio (T/GCN)
enron	15487	3929	3.9
condmat	14540	1676	8.6
astro	17890	1588	11.2
hepth	14967	1310	11.42
hepph	35696	2134	16.72
ytube	2400	257	9.33

Table 5.6: Comparison of time to find high centrality vertices. Evaluation was done on a workstation desktop running 64bit bit Debian GNU/Linux 9.5 with Intel Xeon CPU E5-2620v3 (2.40 Ghz) and 32GB RAM. Both the algorithms have been executed sequentially and no GPU unit has been used for the GCN model to keep the comparison fair.

5.4 Summary of the chapter

To summarize, our **key contributions** are as follows:

- We develop a word2vec inspired skip-gram based framework for learning rich vector representation of nodes in graph data. We propose a random walk based exploration

strategy with added network structure based constraint which guides nodes to obtain similar neighborhood. To the best of our knowledge this is the first work which utilize the *onion layered* network hierarchy for random walk design.

- We propose two metrics, i.e., *closeness* and *separability* through which we show that our scheme brings nodes with similar core ids closer and distances nodes with different core ids farther in the vector space compared to state-of-the-art methods.
- We compare our algorithmic framework against competing methods on downstream word similarity task and obtain significant improvement in performance (atmost 46% in certain cases).
- We further adopt semi supervised distant learning on graph data and propose an approach to identify path based influential nodes in graph data leveraging core periphery structure.

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the main contributions of the thesis and take a stock of our achievements vis-a-vis the objectives set up in the introductory chapter. Finally, we wrap up by pointing out some of the possible future directions of research that have been opened up by this thesis.

6.1 Summary of Contribution

In this thesis our primary objective has been to study distinguishing properties of *k*-core subgraph in several real world networks and develop applications based on such characteristics. To accomplish these goals, we have studied *k*-core subgraph in both static and dynamic setting across a host of real world networks. We also developed a novel network representation approach which capitalizes on the ordered partition of the graph into different levels obtained by *k*-core decomposition. *On of the key findings in this thesis is that* k*-core subgraphs can be considered as a important tool for extracting top path based central nodes in the network without any explicit centrality computation*. A detailed summary of our contributions are listed below.

6.1.1 Implications of central node localization in graph degeneracy

In the first chapter, we showed through data driven approaches that *k*-core subgraph acts as a container for major path based central nodes in some real world networks. For this group of networks we showed topological signatures which discriminates them from a group of networks where path based central nodes do not occur in *k*-core subgraph. To the best of our knowledge, this is the first comprehensive work which seeks to investigate utility of *k*-core subgraph in discovering central nodes and show how their location in some networks led to higher resilience as well as candidates for information propagation. The contributions of this work can be summarized as below.

- We show that in some networks a large fraction of high central (closeness, betweenness) nodes are located in the innermost *k*-core subgraph. We calculate the Jaccard coefficient between set of innermost *k*-core nodes and equal number of top central nodes. We find the Jaccard coefficient for closeness to be ~ 0.86 and for betweenness to be ~ 0.8. However in the second class of networks we find Jaccard coefficient to be ~ 0.05 for both centrality metrics. We term the *k*-core subgraph in the first class of networks as rich centrality clubs.

- We show that not all networks have clique like graph degeneracy. We discover a class of networks where a set of nodes in graph degeneracy are part of multiple communities. Specifically, we find group of networks where high core nodes are distributed in several network communities. Due to such organization they play prominent role in information propagation in the network. However we also find a group of networks where high core nodes are located in at most two communities.

- We show that second eigenvalue of the normalised Laplacian, calculated from the shell volume of a network serves as an excellent metric in discriminating networks into two categories. In the first group inner shells increasingly act as bottleneck as we move inwards. This causes multiple shortest paths in the network to lie through the *k*-core subgraph.

- We develop novel network perturbation models and show how centrality resilience is impacted due these processes. Our results show that networks with rich centrality

clubs are robust to random perturbations. However they show increased vulnerability in case of targeted attacks.

- Finally we show that subgraphs with high central nodes can be discovered by exploring second order connectivity in the network. To the best of our belief this is the first work which explores utility of k-core decomposition in extracting rich centrality clubs. For networks such as *protein, facebook* where top 20 closeness and betweenness nodes are not present in the innermost k-core, we find that constructing meta-network and applying k-core decomposition on the derived network helps in extracting 95% of the top 20 central nodes in the original network.

6.1.2 Prediction of central nodes in dynamic networks

One of the key problems in temporal graphs is understanding whether influential nodes in the current timepoint retain their importance in future state of the network. In this chapter we focus on answering this question by leveraging k-core structure in evolving graphs. We propose novel heuristics based on the network structure, which are designed to estimate, to what extent path based central nodes in the current timepoint retain their strategic position in the network in future. Besides feature engineering contribution, we show utility of time series models such as ARIMA, ARMA, MA in predicting actual high central nodes in the network. In summary, the major contributions of our work are listed as follows:

- We develop a two-step algorithm for predicting the high centrality vertices for dynamic networks. Initially, we estimate the overlap between the set of high centrality vertices of the current time step to the set of high centrality vertices of the future time step. Toward this goal we employ sophisticated time series models such as ARIMA and show that this approach results in low mean percentage error. In the next step, assuming that the network snapshot is already available in time, we analyze its innermost core to find the ids of the high centrality vertices.

- A key contribution of our work is that we develop a set of *novel heuristics to classify networks based on the extent to which the highly central vertices are in the innermost core*. Our heuristics do not require the explicit computation of the centrality values.

We also separately report our predictions for each class of networks. We empirically demonstrate that the higher the number of high centrality vertices in the inner core, the higher is the accuracy with which we can predict these vertices for future time steps. For real networks that maintain this property to the largest extent, our $F1$-score for prediction is **0.81** for closeness and **0.72** for betweenness. Similarly, for synthetic networks that maintain this property to best extent, the $F1$-score for prediction is **0.94** for closeness and **0.92** for betweenness.

- In addition to $F1$-scores, we further validate our results by comparing how the predicted and actual high centrality vertices perform in a practical context. For high closeness centrality vertices, we compare the time to spread a message when the high centrality vertices are taken as seeds, and for the high betweenness centrality vertices, we compare how the length of the diameter increases as the high betweenness centrality vertices are deleted from the network. For these experiments we select a set of random vertices as control, and compare the performance of the actual high centrality vertices and the predicted high centrality vertices. For networks where we could predict the results with high accuracy, the effect of the original and predicted vertices are very similar, and these results are markedly different from the effect of the randomly selected vertices.

6.1.3 Representation learning using core periphery structure

This chapter unfolds two novel approaches of network representation learning which is supported by *k*-core structure of the network. Unsupervised network representation learning borrows many of its key techniques from skip-gram based word representation developed for natural language processing domain. Supervised network representation learning emerged from advances in computer vision domains. However both these methods have found significant adoption among network mining practitioners because they lend scalable methods for automatic generation of features for individual nodes. The major contributions in this chapter are summarized below

- We design a novel network embedding model *core2vec*, with a unique exploration strategy for context nodes guided by global information. Our technique does not

require any compute intensive meta information like community label or domain specific node level attributes.

- We demonstrate that our model can favorably map similar core nodes closer in space and distant core nodes farther in space.
- We illustrate an application of our embedding in word analogy task, where compare similar words scored by our method against manually curated gold standard scores. Our method can capture similar words better from the baseline techniques.
- We propose a novel semi-supervised learning approach for ranking influential nodes based on core-periphery structure.

6.2 Future direction

In this final section, we outline a few out of several possible directions of future research that have been opened up by this thesis.

- This thesis shows comprehensively how nodes in the k-core subgraph can be used as influential spreaders of information. There exist series of works which connect information propagation with influence maximisation. Influence maximisation problem revolves around finding a small subset of nodes which can be used as initial seed set to initiate information dissemination in the network for optimal spreading. It has been shown in [86] that for well defined diffusion processes such as independent cascade model and linear threshold model, optimal seed set extraction methods are NP-hard and approximate algorithms need to be applied. A possible direction could be how prior knowledge of network structure obtained by k-core decomposition helps is improving the theoretical bound of the best possible set.
- Spectral analysis of networks has been shown in this thesis as a potential approach to discriminate networks based on connectivity profiles. This has been shown to identify networks where path based central nodes can be located without explicit computation. However such analysis is difficult to perform on very large graphs because eigenvalue computation on large matrices is expensive. However there has

been some recent work such as Benson et. al. [48] which may alleviate such computational hurdles on large graphs.

- Our prediction framework in the second contributory chapter raises further scope of detecting anomalous users, hateful content propagators in microblogging platforms such as *Twitter, Gab*. It has been shown by Mathew et. al. [118, 119] that inciteful and derogatory content propagators often lie in strategic information pathways in the respective platforms and hence automatic detection using our method could be helpful for platform governance.

- In the final contributory chapter, we developed network embedding approaches leveraging k-core organisation. We showed novel application in word association graphs, where our approach obtains better representation for semantically similar words. Distributed representation of words learned from large scale corpus has helped in solving several challenging natural language processing task. These representations can be further improved by incorporating distributional thesaurus network which has been successfully shown by Jana et. al [83]. These developments create opportunity to apply *core2vec* on distributional thesaurus network and compare these representations with other similar approaches.

Bibliography

[1] Caida web site. *http://www.caida.org*.

[2] E. T. Aaron Clauset and M. Sainz. The colorado index of complex networks. *https://icon.colorado.edu*, 2016.

[3] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G. V. Steeg, and A. Galstyan. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067*, 2019.

[4] A. Adiga and A. K. S. Vullikanti. How robust is the core of a network? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 541–556. Springer, 2013.

[5] F. Agneessens, S. P. Borgatti, and M. G. Everett. Geodesic based centrality: Unifying the local and the global. *Social Networks*, 49:12–26, 2017.

[6] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378, 2000.

[7] I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Lanet-vi in a nutshell, 2006.

[8] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, WAW'07, pages 124–137, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] J. Bae and S. Kim. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications*, 395:549–559, 2014.

[10] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[11] A.-L. Barabási et al. *Network science*. Cambridge university press, 2016.

[12] V. Batagelj and M. Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[13] M. Benedek, Y. N. Kenett, K. Umdasch, D. Anaki, M. Faust, and A. C. Neubauer. How semantic memory structure and intelligence contribute to creative thought: a network science approach. *Thinking & Reasoning*, 23(2):158–183, 2017.

[14] O. Benyahia, C. Largeron, B. Jeudy, and O. R. Zaïane. Dancer: Dynamic attributed network with community structure generator. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 41–44. Springer, 2016.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[16] S. P. Borgatti, K. M. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social networks*, 28(2):124–136, 2006.

[17] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. *Social networks*, 21(4):375–395, 2000.

[18] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[19] D. Braha and Y. Bar-Yam. From centrality to temporary fame: Dynamic centrality in complex networks. *Complexity*, 12(2):59–63, 2006.

[20] D. Braha and Y. Bar-Yam. Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions. In *Adaptive Networks*, pages 39–50. Springer, 2009.

[21] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[22] U. Brandes and C. Pich. Centrality estimation in large networks. In *INTL. JOURNAL OF BIFURCATION AND CHAOS, SPECIAL ISSUE ON COMPLEX NETWORKS' STRUCTURE AND DYNAMICS*, 2007.

[23] A. D. Broido and A. Clauset. Scale-free networks are rare. *Nature communications*, 10(1):1017, 2019.

[24] J. Bruna and X. Li. Community detection with graph neural networks. *stat*, 1050:27, 2017.

[25] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[26] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900. ACM, 2015.

[27] T. Carnes, C. Nagarajan, S. M. Wild, and A. Van Zuylen. Maximizing influence in a competitive social network: a follower's perspective. In *Proceedings of the ninth international conference on Electronic commerce*, pages 351–360. ACM, 2007.

[28] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1, 2008.

[29] H. Chan, L. Akoglu, and H. Tong. Make it or break it: Manipulating robustness in large networks. In *SDM*, pages 325–333. SIAM, 2014.

[30] H. Chan, S. Han, and L. Akoglu. Where graph topology matters: the robust subgraph problem. In *SDM*, pages 10–18. SIAM, 2015.

[31] H. Chen, B. Perozzi, Y. Hu, and S. Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[32] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[33] M. Choudhury, D. Chatterjee, and A. Mukherjee. Global topology of word co-occurrence networks: Beyond the two-regime power-law. In *Proceedings of the 23rd international conference on computational linguistics: Posters*, pages 162–170. Association for Computational Linguistics, 2010.

[34] F. Chung. Laplacians of graphs and cheeger inequalities. 1996.

[35] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.

[36] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[37] M. Cinelli, G. Ferraro, and A. Iovanella. Resilience of core-periphery networks in the case of rich-club. *Complexity*, 2017, 2017.

[38] A. Clauset. Finding local community structure in networks. *Physical review E*, 72(2):026132, 2005.

[39] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 16, 2008.

[40] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani. Detecting rich-club ordering in complex networks. *Nature physics*, 2(2):110, 2006.

[41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[42] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi. Structure and dynamics of core/periphery networks. *Journal of Complex Networks*, 1(2):93–123, 2013.

[43] S. De Deyne, A. Perfors, and D. J. Navarro. Predicting human similarity judgments with distributional models: The value of word associations. In *Coling 2016*, pages 1861–1870, 2016.

[44] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[45] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical Rev. Lett.*, 94(16):160202, 2005.

[46] O. Diekmann and J. A. P. Heesterbeek. *Mathematical epidemiology of infectious diseases: model building, analysis and interpretation*, volume 5. John Wiley & Sons, 2000.

[47] F. Ding, Y. Shi, and T. Chen. Performance analysis of estimation algorithms of nonstationary arma processes. *IEEE Transactions on Signal Processing*, 54(3):1041–1053, 2006.

[48] K. Dong, A. R. Benson, and D. Bindel. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1152–1161, 2019.

[49] V. M. Eguiluz and K. Klemm. Epidemic threshold in structured scale-free networks. *Physical Review Letters*, 89(10):108701, 2002.

[50] E. Estrada. Topological structural classes of complex networks. *Physical Review E*, 75(1):016103, 2007.

[51] E. Estrada. *The structure of complex networks: theory and applications*. Oxford University Press, 2012.

[52] Z. A. Farhath, B. Arputhamary, and L. Arockiam. A survey on arima forecasting using time series model. *Int. J. Comput. Sci. Mob. Comput*, 5:104–109, 2016.

[53] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM, 2001.

[54] S. for Industrial and A. Mathematics. *SIAM journal on matrix analysis and applications*. SIAM, 1988.

[55] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[56] R. J. Gallagher, J.-G. Young, and B. F. Welles. A clarified typology of core-periphery structure in networks. *arXiv preprint arXiv:2005.10191*, 2020.

[57] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 90–100, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[58] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*, pages 61–70. ACM, 2012.

[59] G. Ghoshal and A.-L. Barabási. Ranking stability and super-stable nodes in complex networks. *Nature communications*, 2:394, 2011.

[60] C. Giatsidis, F. D. Malliaros, D. M. Thilikos, and M. Vazirgiannis. Corecluster: A degeneracy based graph clustering framework. 2014.

[61] P. M. Gleiser and L. Danon. Community structure in jazz. *Adv. in comp. sys.*, 6(04):565–573, 2003.

[62] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[63] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[64] J. Guo, H. He, and C. Sun. Arima-based road gradient and vehicle velocity prediction for hybrid electric vehicle energy management. *IEEE Transactions on Vehicular Technology*, 2019.

[65] A. Gutfraind, I. Safro, and L. A. Meyers. Multiscale network generation. In *IEEE Information Fusion*, pages 158–165. IEEE, 2015.

[66] H. Habiba, C. Tantipathananandh, and T. Berger-Wolf. Betweenness centrality measure in dynamic networks. *Department of Computer Science, University of Illinois at Chicago, Chicago*, 2007.

[67] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren. Large-scale learning of word relatedness with constraints. In *KDD*, pages 1406–1414. ACM, 2012.

[68] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.

[69] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[70] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[71] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[72] S. Hempel, A. Koseska, J. Kurths, and Z. Nikoloski. Inner composition alignment for inferring directed networks from short time series. *Phys. Rev. Lett.*, 107(5):054101, 2011.

[73] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[74] F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Comp. Ling.*, 41(4):665–695, 2015.

[75] S. A. Hill and D. Braha. Dynamic model of time-dependent complex networks. *Physical Review E*, 82(4):046105, 2010.

[76] T. Hills. The company that words keep: comparing the statistical structure of child-versus adult-directed language. *J. of child language*, 40(3):586–604, 2013.

[77] P. Holme. Core-periphery organization of complex networks. *Phys Rev E*, 72(4):046111, 2005.

[78] P. Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015.

[79] P. Holme and J. Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[80] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[81] Q. Huang, C. Zhao, X. Zhang, X. Wang, and D. Yi. Centrality measures in temporal networks with time series analysis. *EPL (Europhysics Letters)*, 118(3):36001, 2017.

[82] R. F. i Cancho and R. V. Solé. The small world of human language. *Proc. of the R. Soc. of London B: Bio. Sci.*, 268(1482):2261–2265, 2001.

[83] A. Jana and P. Goyal. Can network embedding of distributional thesaurus be combined with word vectors for better representation? *arXiv preprint arXiv:1802.06196*, 2018.

[84] S. Jiang, L. Guo, X. Zhang, and H. Wang. Lightflood: Minimizing redundant messages and maximizing scope of peer-to-peer search. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):601–614, 2008.

[85] A. Jimenez-Cortadi, F. Boto, I. Irigoien, B. Sierra, and G. Rodriguez. Time series forecasting in turning processes using arima model. In *International Symposium on Intelligent and Distributed Computing*, pages 157–166. Springer, 2018.

[86] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[87] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. *arXiv preprint arXiv:1901.06032*, 2019.

[88] H. Kim and R. Anderson. Temporal node centrality in complex networks. *Physical Review E*, 85(2):026107, 2012.

[89] H. Kim, J. Tang, R. Anderson, and C. Mascolo. Centrality prediction in dynamic human contact networks. *Computer Networks*, 56(3):983–996, 2012.

[90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[91] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[92] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse. Identification of influential spreaders in complex networks. *Nat. Phys.*, 6(11):888–893, 2010.

[93] S. Kojaku and N. Masuda. Finding multiple core-periphery pairs in networks. *Physical Review E*, 96(5):052313, 2017.

[94] S. Kojaku and N. Masuda. Core-periphery structure requires something else in the network. *New Journal of Physics*, 20(4):043012, 2018.

[95] J. Kunegis. Konect: the koblenz network collection. In *WWW*, pages 1343–1350. ACM, 2013.

[96] R. Laishram, A. E. Sariyüce, T. Eliassi-Rad, A. Pinar, and S. Soundarajan. Measuring and improving the core resilience of networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 609–618. International World Wide Web Conferences Steering Committee, 2018.

[97] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[98] K. Lerman, R. Ghosh, and J. H. Kang. Centrality metric for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 70–77. ACM, 2010.

[99] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[100] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[101] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM TIST*, 8(1):1, 2016.

[102] C. Li, Q. Li, P. Van Mieghem, H. E. Stanley, and H. Wang. Correlation between centrality metrics and their application to the opinion model. *The European Physical Journal B*, 88(3):65, 2015.

[103] H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui. Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*, 2019.

[104] J.-H. Lin, Q. Guo, W.-Z. Dong, L.-Y. Tang, and J.-G. Liu. Identifying the node spreading influence with largest k-core values. *Physics Letters A*, 378(45):3279–3284, 2014.

[105] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[106] B. Liu, X. Tang, J. Cheng, and P. Shi. Traffic flow combination forecasting method based on improved lstm and arima. *arXiv preprint arXiv:1906.10407*, 2019.

[107] C. Liu, S. C. Hoi, P. Zhao, and J. Sun. Online arima algorithms for time series prediction. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[108] H.-L. Liu, C. Ma, B.-B. Xiang, M. Tang, and H.-F. Zhang. Identifying multiple influential spreaders based on generalized closeness centrality. *Physica A: Statistical Mechanics and its Applications*, 492:2237–2248, 2018.

[109] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3):62, 2018.

[110] Y. Liu, T. Safavi, N. Shah, and D. Koutra. Reducing large graphs to small supergraphs: a unified approach. *Social Network Analysis and Mining*, 8(1):17, 2018.

[111] V. H. Louzada, F. Daolio, H. J. Herrmann, and M. Tomassini. Smart rewiring for network robustness. *Journal of Complex networks*, 1(2):150–159, 2013.

[112] L. Lv, K. Zhang, T. Zhang, D. Bardou, J. Zhang, and Y. Cai. Pagerank centrality for temporal networks. *Physics Letters A*, 383(12):1215–1222, 2019.

[113] A. Mahmoody, C. E. Tsourakakis, and E. Upfal. Scalable betweenness centrality maximization via sampling. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1765–1773, 2016.

[114] A. S. Maiya and T. Y. Berger-Wolf. Expansion and decentralized search in complex networks. *Knowledge and information systems*, 38(2):469–490, 2014.

[115] F. Malliaros, C. Giatsidis, A. Papadopoulos, and M. Vazirgiannis. The core decomposition of networks: Theory, algorithms and applications. 2019.

[116] F. D. Malliaros and V. Megalooikonomou. Expansion properties of large social graphs. In *International Conference on Database Systems for Advanced Applications*, pages 311–322. Springer, 2011.

[117] F. D. Malliaros, M.-E. G. Rossi, and M. Vazirgiannis. Locating influential nodes in complex networks. *Scientific Reports*, 6:19307, 2016.

[118] B. Mathew, R. Dutt, P. Goyal, and A. Mukherjee. Spread of hate speech in online social media. In *Proceedings of the 10th ACM Conference on Web Science*, pages 173–182, 2019.

[119] B. Mathew, A. Illendula, P. Saha, S. Sarkar, P. Goyal, and A. Mukherjee. Temporal effects of unmoderated hate speech in gab. *arXiv preprint arXiv:1909.10966*, 2019.

[120] E. McKenzie. General exponential smoothing and the equivalent arma process. *Journal of Forecasting*, 3(3):333–344, 1984.

[121] S. Mehrmolaei and M. R. Keyvanpour. Time series forecasting using improved arima. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 92–97. IEEE, 2016.

[122] P. Meyer, H. Siy, and S. Bhowmick. Identifying important classes of large software systems through k-core decomposition. *Advances in Complex Systems*, 17(07n08):1550004, 2014.

[123] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[124] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Adv. in neu. inf. proc. sys.*, pages 3111–3119, 2013.

[125] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[126] B. Mohar. Isoperimetric numbers of graphs. *Journal of combinatorial theory, Series B*, 47(3):274–291, 1989.

[127] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[128] C. Moore and M. E. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678, 2000.

[129] C. Morselli, V. H. Masias, F. Crespo, and S. Laengle. Predicting sentencing outcomes with centrality measures. *Security Informatics*, 2(1):4, 2013.

[130] S. Mumtaz and X. Wang. Identifying top-k influential nodes in networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2219–2222, 2017.

[131] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. The university of south florida free association, rhyme, and word fragment norms. *Behav. Res. Meth., Inst., & Comp.*, 36(3):402–407, 2004.

[132] M. Newman. *Networks*. Oxford university press, 2018.

[133] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora. Graph metrics for temporal networks. In *Temporal networks*, pages 15–40. Springer, 2013.

[134] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. Asymmetric transitivity preserving graph embedding. In *KDDg*, pages 1105–1114. ACM, 2016.

[135] N. Park, A. Kan, X. L. Dong, T. Zhao, and C. Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. *arXiv preprint arXiv:1905.08865*, 2019.

[136] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[137] S. Pei and H. A. Makse. Spreading dynamics in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(12):P12002, 2013.

[138] C. Peng, T. G. Kolda, and A. Pinar. Accelerating community detection by using k-core subgraphs. *arXiv preprint arXiv:1403.2226*, 2014.

[139] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710. ACM, 2014.

[140] A. Prior and S. Bentin. Word associations are formed incidentally during sentential semantic integration. *Acta Psychologica*, 127(1):57–71, 2008.

[141] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2110–2119. ACM, 2018.

[142] M. Qu, Y. Bengio, and J. Tang. Gmnn: Graph markov neural networks. *arXiv preprint arXiv:1905.06214*, 2019.

[143] M. P. Rombach and M. A. Porter. Discriminating power of centrality measures. *arXiv preprint arXiv:1305.3146*, 2013.

[144] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core-periphery structure in networks (revisited). *SIAM Rev.*, 59(3):619–646, 2017.

[145] M.-E. G. Rossi, F. D. Malliaros, and M. 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*, pages 101–102. ACM, 2015.

[146] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[147] P. Rozenshtein and A. Gionis. Temporal pagerank. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 674–689. Springer, 2016.

[148] S. Sarkar, B. P. Reddy, S. Sikdar, and A. Mukherjee. StRE: Self attentive edit quality prediction in Wikipedia. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3962–3972, Florence, Italy, July 2019. Association for Computational Linguistics.

[149] A. Scherrer, P. Borgnat, E. Fleury, J.-L. Guillaume, and C. Robardet. Description and simulation of dynamic mobility networks. *Computer Networks*, 52(15):2842–2858, 2008.

[150] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[151] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences*, 108(10):3838–3841, 2011.

[152] S. Segarra and A. Ribeiro. A stable betweenness centrality measure in networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3859–3863. IEEE, 2014.

[153] S. Segarra and A. Ribeiro. Stability and continuity of centrality measures in weighted graphs. *IEEE Transactions on Signal Processing*, 64(3):543–555, 2015.

[154] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[155] A. Severyn and A. Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 464–469, 2015.

[156] M. Shanahan and M. Wildie. Knotty-centrality: finding the connective core of a complex network. *PLoS One*, 7(5):e36579, 2012.

[157] Y. Shavitt and E. Shir. Dimes: Let the internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, 2005.

[158] K. Shin, T. Eliassi-Rad, and C. Faloutsos. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 469–478. IEEE, 2016.

[159] K. Shin, T. Eliassi-Rad, and C. Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, Jun 2017.

[160] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[161] S. Sikdar, N. Ganguly, and A. Mukherjee. Time series analysis of temporal networks. *The Euro. Phys. Jour. B*, 89(1):1–11, 2016.

[162] M. Silva, H. Ma, and A.-P. Zeng. Centrality, network capacity, and modularity as parameters to analyze the core-periphery structure in metabolic networks. *Proceedings of the IEEE*, 96(8):1411–1420, 2008.

[163] S. Singh, A. Mohapatra, et al. Repeated wavelet transform based arima model for very short-term wind speed forecasting. *Renewable energy*, 136:758–768, 2019.

[164] N. Stanley, R. Kwitt, M. Niethammer, and P. J. Mucha. Compressing networks with super nodes. *Scientific reports*, 8(1):10892, 2018.

[165] L. B. Szalay and J. Deese. *Subjective meaning and culture: An assessment through word associations*. Lawrence Erlbaum Associates, 1978.

[166] J. Tang, I. Leontiadis, S. Scellato, V. Nicosia, C. Mascolo, M. Musolesi, and V. Latora. Applications of temporal graph metrics to real-world networks. In *Temporal Networks*, pages 135–159. Springer, 2013.

[167] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[168] D. Taylor, S. A. Myers, A. Clauset, M. A. Porter, and P. J. Mucha. Eigenvector-based centrality measures for temporal networks. *arXiv preprint arXiv:1507.01266*, 2015.

[169] D. Taylor, M. A. Porter, and P. J. Mucha. Tunable eigenvector-based centralities for multiplex and temporal networks. *arXiv preprint arXiv:1904.02059*, 2019.

[170] S. Tsugawa and H. Ohsaki. Analysis of the robustness of degree centrality against random errors in graphs. In *Complex Networks VI*, pages 25–36. Springer, 2015.

[171] V. Ufimtsev, S. Sarkar, A. Mukherjee, and S. Bhowmick. Understanding stability of noisy networks through centrality measures and local connections. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2347–2352. ACM, 2016.

[172] S. Vashishth, M. Bhandari, P. Yadav, P. Rai, C. Bhattacharyya, and P. Talukdar. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3308–3318, 2019.

[173] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[174] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[175] J. Wang and J. Cheng. Truss decomposition in massive networks. *Proceedings of the VLDB Endowment*, 5(9):812–823, 2012.

[176] Y. Wang, L. Xu, and B. Wu. A community detection method based on k-shell. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2314–2319. IEEE, 2015.

[177] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[178] B. Yan and J. Luo. Multicores-periphery structure in networks. *Network Science*, 7(1):70–87, 2019.

[179] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[180] L. Yang, D. Jin, D. He, H. Fu, X. Cao, and F. Fogelman-Soulie. Improving the efficiency and effectiveness of community detection via prior-induced equivalent super-network. *Scientific reports*, 7(1):634, 2017.

[181] Y. Yang, Y. Dong, and N. V. Chawla. Predicting node degree centrality with the node prominence profile. *Scientific reports*, 4:7236, 2014.

[182] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

[183] L. Yao, L. Su, Q. Li, Y. Li, F. Ma, J. Gao, and A. Zhang. Online truth discovery on time series data. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 162–170. SIAM, 2018.

[184] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[185] Y. Zhang, S. Pal, M. Coates, and D. Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5829–5836, 2019.

[186] H. Zhou, V. C. Leung, C. Zhu, S. Xu, and J. Fan. Predicting temporal social contact patterns for data forwarding in opportunistic mobile networks. *IEEE Transactions on Vehicular Technology*, 66(11):10372–10383, 2017.

[187] H. Zhou, S. Xu, and C. Huang. Temporal centrality prediction in opportunistic mobile social networks. In *International Conference on Internet of Vehicles*, pages 68–77. Springer, 2015.

[188] S. Zhou and R. J. Mondragón. Accurately modeling the internet topology. *Physical Review E*, 70(6):066108, 2004.

[189] S. Zhou and R. J. Mondragón. The rich-club phenomenon in the internet topology. *IEEE Communications Letters*, 8(3):180–182, 2004.

[190] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[191] X. Zhu, J. Lafferty, and R. Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of ..., 2005.

Appendix A

List of all Publications by the Candidate

Following is a list of all the publications by the candidate including those on and related to the work presented in the thesis. The publications are arranged in reverse chronological order and in the following sections – (i) Journals, (ii) Book Chapters, and (ii) Conferences

A.1 Journal Publications

1. Binny Mathew, Anurag Illendula, Punyajoy Saha, **Soumya Sarkar**, Pawan Goyal, Animesh Mukherjee *Hate begets Hate: A Temporal Study of Hate Speech.* Proceedings of the ACM on Human-Computer Interaction (CSCW 2020) *(To Appear)*

2. **Soumya Sarkar**, Sandipan Sikdar, Sanjukta Bhowmick, and Animesh Mukherjee. *Using core-periphery structure to predict high centrality nodes in time-varying networks.* Data Mining and Knowledge Discovery 32, no. 5 (2018): 1368-1396. accepted in ECML PKDD 2018 Journal Track

A.2 Book Chapters

1. **Soumya Sarkar**, Suhansanu Kumar, Sanjukta Bhowmick, and Animesh Mukherjee. *Centrality and Community Scoring Functions in Incomplete Networks: Their Sensitivity, Ro-*

bustness, and Reliability. In Machine Learning Techniques for Online Social Networks, pp. 135-154. Springer, Cham, 2018.

2. **S. Sarkar**, A. Karn, S. Bhowmick, A. Mukherjee. *An Empirical Study of the Effect of Noise Models on Centrality Metrics.* Dynamics on and off Complex Networks III (Editors: Ghanbarnejad, F., Saha Roy, R., Karimi, F., Delvenne, J.-C., Mitra, B. ISBN: 978-3-030-14682-5)

A.3 Conference Publications

1. Bhanu Prakash Reddy*, Sasi Bhushan*, **Soumya Sarkar***, Animesh Mukherjee (*equal contribution) *NwQM: A neural quality assessment framework for Wikipedia* In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020), pages xx–xx, 2020.

2. **S. Sarkar**, B. P. Reddy, S. Sikdar, and A. Mukherjee. *StRE: Self attentive edit quality prediction in Wikipedia.In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3962-3972, Florence, Italy, July 2019. Association for Computational Linguistics.*

3. **Soumya Sarkar**, Sanjukta Bhowmick, and Animesh Mukherjee. *On Rich Clubs of Path-Based Centralities in Networks. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 567-576. ACM, 2018.*

4. **Soumya Sarkar**, Aditya Bhagwat , Animesh Mukherjee *Core2Vec: A Core-Preserving Feature Learning Framework for Networks.* In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 487-490. IEEE, 2018.

5. Sandipan Sikdar, Tanmoy Chakraborty, **Soumya Sarkar**, Niloy Ganguly, and Animesh Mukherjee. 2018. ComPAS: Community Preserving Sampling for Streaming Graphs. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18), 184-192.

6. **Soumya Sarkar**, Suhansanu Kumar, Sanjukta Bhowmick, and Animesh Mukherjee. *Sensitivity and reliability in incomplete networks: Centrality metrics to community scoring functions.* In 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 69-72. IEEE, 2016.

7. Ufimtsev, Vladimir, **Soumya Sarkar**, Animesh Mukherjee, and Sanjukta Bhowmick. *Understanding stability of noisy networks through centrality measures and local connections.* In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 2347-2352. ACM, 2016.