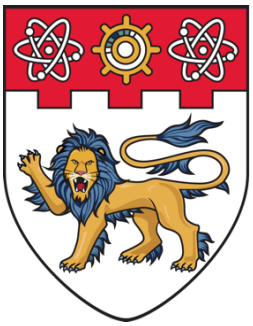


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

GRAPH MINING IN FINTECH-DRIVEN PAYMENT INDUSTRY FOR RISK MANAGEMENT

CHEN ZHE

School of Computer Science and Engineering

PayPal Innovation Lab (PPIL)

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

19/01/2021

. . . . . .

Date

. . . . . .

Chen Zhe

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

19/01/2021

. . . . . .
Date

. . . . . .
Sun Aixin

Authorship Attribution Statement

Please select one of the following; *delete as appropriate:

*(B) This thesis contains material from [3] paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Please amend the typical statements below to suit your circumstances if (B) is selected.

Chapter 3 is published as Zhe, C., Sun, A., & Xiao, X. (2019, July). Community detection on large complex attribute network. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2041-2049).

The contributions of the co-authors are as follows:

- Prof Xiao provided the initial project direction and scope.
- I, Prof Sun and Prof Xiao analyzed the graph and attribute data on transaction network and co-design the framework of the model.
- I conduct the experiment and model evaluation on both open datasets and PayPal seller community detection study.
- Prof Sun and Prof Xiao provide guidelines on performing model on large scale network data
- I prepare the paper draft and the draft was revised by Prof Sun and Prof Xiao.
- Prof Sun, Prof Xiao provide suggestions on paper writing style and proper presentations.
- Prof Sun edited the paper draft.

Chapter 4 is published as Zhe, C., Sun, A., & Xiao, X. (2021). Incremental Community Detection on Large Complex Attributed Network. ACM Transactions on Knowledge Discovery from Data (TKDD), 15(6), 1-20.

The contributions of the co-authors are as follows:

- Prof Sun provided the initial project direction and scope.
- I and Prof Sun analyzed the graph and attribute data on transaction network and co-design the framework of the model.
- Prof Sun provide guidelines on incremental model evaluation.
- I conduct the experiment and model evaluation on both open datasets and PayPal incremental network community detection study.
- I prepare the paper draft and the draft was revised by Prof Sun and Prof Xiao.
- Prof Sun and Prof Xiao provide suggestions on paper writing style, proper presentations and drawing model pipelines.
- Prof Sun edited the paper draft.

Chapter 6 is published as Zhe, C., Sun, A (2020, Nov). Anomaly detection on dynamic bipartite graph with burstiness. IEEE International Conference on Data Mining 2020.

The contributions of the co-authors are as follows:

- Prof Sun provided the initial project direction and scope.
- I and Prof Sun analyzed the CSP log network data and co-design the framework of the model for anomaly detection.
- Prof Sun provide guidelines on burstiness detection and anomaly detection.
- I conduct the experiment and model evaluation on both open datasets and CSP log network anomaly detection study.
- I prepare the paper draft and the draft was revised by Prof Sun.
- Prof Sun provides suggestions on paper writing style, proper presentations and drawing model pipelines.
- Prof Sun edited the paper draft.

20/06/2021

. . . . . .
Date

. . . . . .
Chen Zhe

Acknowledgments

I would take this opportunity to express my deepest and sincerest gratitude to my advisors at Nanyang Technological University (NTU) and National University of Singapore (NUS). They are my supervisors **Prof. Sun Aixin**, **Prof. Xiao Xiaokui** and my TAC members **Prof. Cong Gao**, **Prof. Li Xiaoli**. Their expertise was invaluable in formulating my research path and methodology. They gave me many useful suggestions throughout my whole PhD studies. Without their tremendous encouragement and guidance, it would be impossible for me to complete my PhD study.

I would also like to thank my schoolmates and colleagues in PayPal for their continuous encouragement and help. I would particularly to single out my managers in PayPal for their continuous technical and resource support. In particular, I appreciate **Ms. Lim Siewhoon**, **Mr. Tang Ferdinand**, and **Mr. Tso Jerry**. I want to thank them for all of the opportunities I was given to further my research.

Last but not least, my special thanks to my family. I would like to thank my parents for their constant understanding and love. I would like to show my greatest appreciation to my wife, Ms. Shi Haiyan for all the support she did in my study and life.

Contents

List of Figures

List of Tables

Abstract

Financial Technology, also called "Fintech", is now one of the most popular terms that describe the novel technologies adopted by financial industries. Fintech brings new opportunities for financial services. Take the payment industry as an example, FinTech allows payments to be processed instantly from mobile phones and computers. All transactions are encrypted as data and are performed over the internet. The convenience brought by Fintech also leaves the door open to many risks. A few major risks are market risk, cyber risk, and fraud risk [42]. In this thesis, we study three real risk management applications that are derived from the global leading FinTech-driven payment enterprise PayPal. In specific, the three applications include seller community detection, risky seller detection, and cyber network anomaly detection.

Seller community detection aims to identify the community-wise relationship between sellers on the payment network. Understanding the community structure of a payment network is important to manage the risk and compliance (*e.g.,* identify risky/illegal seller community). The payment network from PayPal contains millions of sellers and billions of transactions, and the sellers are described in a large number of attributes with incomplete values. To detect communities, an algorithm should ideally consider both seller attributes and transaction connectivity. Further, the algorithm has to be able to handle incomplete and complex attributes. We focus on those requirements and propose a framework named AGGMMR to effectively address the challenges from scalability, mixed attributes, and incomplete value. Network

from industry usually keeps growing. It is infeasible to run community detection algorithm from scratch whenever new vertices or edges are joining the network, especially when the network is enterprise-scale (*e.g.,* millions of vertices, billions of edges). Based on this motivation, we extend the AGGMMR and propose inc-AGGMMR to detect communities on dynamic networks with an incremental approach.

Risky seller detection aims to detect sellers on payment network that may result in revenue loss due to various reasons (*e.g.,* fraud, bankruptcy, bad suppliers). To detect risky seller on PayPal payment network, it is crucial to model both seller's transaction connectivity and topology. Transaction connectivity captures the interactions between seller/buyers and transaction topology reveals the seller's business model (*e.g.,* supplier, drop-shipper, or retailer). Based on this motivation, we design a dual-path graph convolution model named DP-GCN to effectively identify risky seller with a vertex classification framework.

Cyber network anomaly detection aims to identify abnormal behaviors on computer network. In specific, we study the problem of detecting malicious domains on PayPal Content Security Policy (CSP) log network. CSP is a security standard that provides an additional layer of protection from cross-site scripting (XSS), clickjacking, and other code injection attacks. CSP log network models the traffics between external domains and PayPal internal servers. This application helps PayPal prevent risks from information interception and account takeover. Analyzing the traffic connectivity pattern is an effective way to detect network anomalies. In the CSP log network, the anomaly may arise from not only connectivity patterns but also burstiness in network activities. We model the CSP log network as a bipartite graph and propose a framework named BEA to detect anomalies from both connectivity patterns and burstiness.

Comprehensive experimental evaluations on open datasets and applications on PayPal networks demonstrate the effectiveness and practicality of the proposed frameworks and models.

Chapter 1

Introduction

1.1 Research Background

In the last decade, digitalization has a strong impact on the financial service industry. Financial Technology (FinTech) becomes one of the hottest topics according to the statistics of FinTech from Google Trends 1.1. FinTech stands for the future of financial services and it is the synergy between finance and technology. The rapid growth of new financial services by FinTech drives payment service industries towards operational innovation in order to gain sustainable competitive advantage. Nowadays, FinTech allows most of payment services being exclusively based on data [102]. In specific, user is able to send safe and fast money transfers simply from their browsers or smartphones with simplified account-opening procedures. It reduces friction for cross-border or cross-currency transactions. These technology-led, payment services become increasingly transparent and those services are offered by non-bank, but technology companies. However, as technology remains a double-edged sword, the concerns about risk have also increased. The increasing volume of digital payments may not be subjected to the same level of regulatory oversight as banks under the current structure [28]. In other words, a new type of risk is arisen along with the growth of technology. In [37], three key risks for digital payments are identified: (1). Reduced

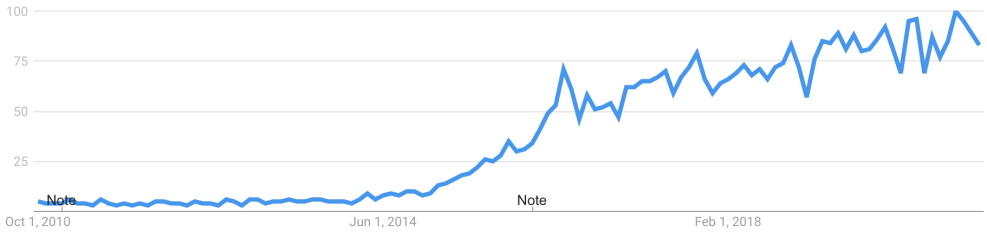


Figure 1.1: The google trends on keyword "Fintech".

information: making identity verification more difficult. (2). Faster transaction speeds: reducing time to perform precautionary checks. (3). Novice participants: while low market barriers foster innovation, they also leave inexperienced service providers responsible for mitigating financial crime risks. Relevant examples of those risks in FinTech driven payment services are frauds, cyber attacks, online illegal businesses, bad sellers. This calls for related industries to adapt stronger risk measurements. Thus FinTech risk management becomes a central point of interest for payment service industries, and requires research and development of novel measurements [42]. In recent years, the advance of AI and machine learning allow the FinTech industry to emerge as promising solutions to protect from those various risks.

Many data in the payment service industry are in the form of network graph, such as buyer-seller transactions in payment network, client-server traffics in backend service network. In this thesis, we study three important risk management applications on PayPal network graph:

- **Seller Community Detection on Payment Network:** Understanding payment network's community structures is crucial to ensure it is sustainable and long-lasting. Knowing a seller's community is also important from many aspects - fraud management, compliance, legal and marketing.

- **Risky Seller Detection on Payment Network:** The risky sellers are those that result in revenue loss due to many different reasons *e.g.,* fraud, bankruptcy, bad suppliers. Identifying risky seller in advance can greatly help to cover the loss and protect buyers in the payments.

- **Cyber Anomaly Detection on CSP Network:** Detecting anomalous on backend service network, allows us to identify fraudulent connections and irregular behaviours, thus to prevent fraud, account takeover and payment information stolen.

Due to the nature of network data in the above applications, we cast each of the applications into a specific graph mining task. This allows graph mining techniques to be applied on those networks to solve the related risk management problem.

1.2 Research Motivations and Objectives

There are many challenges preventing existing graph methods in research from being directly applied on those real networks. The main challenges come from scalability, complex data type, incomplete data, and multiple information sources. A large payment network contains millions of sellers and billions of transactions; many existing methods are not scalable enough to be applied on this level of scale. Besides that, the data in industry may come with complex types. There are both numerical and categorical attributes. Moreover, not all vertices on network have complete attributes (*e.g.,* business-related attributes are only available for sellers). Finally, real network application needs to consider multiple information sources (*e.g.,* both business attributes and connectivity for payment network community detection, both burstiness and interaction for cyber network anomaly detection), but many algorithms are designed only for a specific information source. Thus in this research, we aim to bridge the gap between research and industry solutions. We carefully

study the three important applications in payment service industries, and outline the motivations as follows:

- **Seller Community Detection on Payment Network:** The goal of the community detection aims to identify seller communities on payment network. However, finding communities in such an attributed network is not an easy task. Most state-of-the-art algorithms only consider either attributes or network connectivity information, but not both. On the other hand, the optimal balance of using each type of information is unknown because community detection is unsupervised in nature. Moreover, vertex attributes may provide contradictory information with connectivity structure. A large payment network contains millions of sellers and billions of transactions, running community detection algorithm on such a large attributed network is challenging for several reasons. First, the network is very large. Many existing algorithms are not scalable enough to be applied on this level of scale. Second, the network contains heterogeneous vertices. There are both sellers and buyers, and business-related attributes are only available for large sellers. Algorithms that require all vertices to share the same set of attributes cannot be applied. Third, there are both numerical and categorical attributes. Further, the categorical attributes can be many-value or multi-value. Besides that, the payment network keeps growing, it is necessary to detect communities for previous unseen vertices. It is not viable to perform community detection from scratch whenever new vertices get added up in the graph, especially most of real networks are large scale. More importantly, simply including new vertices to detect all vertices' community again may not guarantee to get consistent result. As network keeps evolving, it is difficult to map the communities with new vertices to the previously detected communities.

- **Risky Seller Detection on Payment Network:** In recent years, graph-based methods, especially graph convolution networks have shown

great success in payment network fraud/risk detection [77, 87]. However, a major limitation of such graph convolution networks is that they are only able to model the connectivity relationship between vertices since convolution propagates information across the edges of the graph, which is unable to model other vertex properties. A large payment network contains millions of sellers and transactions. The risky sellers are defined as sellers that result in revenue loss due to many different reasons (*e.g.,* fraud, bankruptcy, bad suppliers). In order to successfully detect those risky sellers in advance, one would ideally want to model sellers' connectivity relationships (*e.g.,* interactions between other sellers, buyers and suppliers) as well as the topology structure (*e.g.,* topology on business models, such as supplier, drop-shipper or retailer).

- **Cyber Anomaly Detection on CSP Network:** The goal of anomaly detection aims to identify the malicious traffic or client who has abnormal activities on cyber network. PayPal CSP log network models the traffics between external domains and PayPal internal servers. In this network, the abnormal activity refers to not only connectivity patterns of an external domain but also its burstiness in network activities. For example, a burst of traffics may indicate a distributed attack from intruders. However, burstiness does not adhere to the common definition of graph anomaly, thus existing anomaly detection methods mostly do not consider the impact of burstiness in graph and not directly applicable to our application.

Despite the business value from all of these three risk management tasks, this dissertation explores three important vertex properties from a research perspective, namely the *vertex modularity*, *vertex topology* and *vertex connectivity*. In seller community detection, vertex modularity is measured to unveil the relationship between vertices, thus detect the underlying community from the network. In risky seller detection, vertex topology structure is

leveraged to identify seller business models, thus to effectively detect risky sellers. In cyber anomaly detection, the connectivity pattern between vertices is modeled, and those anomalies are the vertices that have different patterns from the majority. Based on both business and research value from the three applications, our research focuses on developing practical and effective methods which not only solve these problems but also achieve the state of the art performance.

1.3 Core Contributions

We model the payment network and cyber network as graphs and developed graph solutions/frameworks for each application:

- **Seller Community Detection on Payment Network:** We propose a community detection framework named AGGMMR (Augmented Graph with Modularity Maximisation and Refinement) to detect communities from large-scale attributed graph. Our framework is designed to partition an attributed graph based on both its attributes and connectivity information through a greedy modularity maximisation model. It consists of three phases: (i) augmented graph construction and weight initialisation, (ii) weight learning with modularity maximisation, and (iii) modularity refinement. Specifically, an augmented graph is constructed through centre-based attributed clustering to retain attributes relationships between vertices. All attribute relationships are then transformed into edges in the augmented graph. Modularity maximisation model is then applied to partition the augmented graph, which now contains both attributes and connectivity information. Along with the partition, a fast learning algorithm is proposed to automatically adjust weights on those attributes relationships according to their contributions towards our objective. Because the result of a greedy modularity maximisation model is heavily affected by its processing

order, we propose a modularity refinement algorithm to automatically optimise the result through a fast greedy search. As payment network keeps growing, we further extend the AGGMMR and propose inc-AGGMMR to detect communities on dynamic network.

We evaluate AGGMMR and inc-AGGMMR on four open networks which are publicly available, CiteSeer, Cora, Terrorist Attack, and SinaNet networks. Additionally, we also evaluate both methods on a generated synthetic network. As a case study, we evaluate the algorithms on PayPal payment network. The PayPal network consists of 100 *million* vertices and 1.5 *billion* edges. Among the 100 million vertices, 3 *million* are merchants which are described by 68 attributes. Through the evaluation on public datasets, synthetic dataset and the large scale PayPal networks, we show the effectiveness of AGGMMR and inc-AGGMMR, and their superiority over 7 baselines Louvain [18], Distance Dynamics [115], NAGC [79], CODICIL [108], K-means, GCN [61] and GraphSage [47].

- **Risky Seller Detection on Payment Network:** We propose DP-GCN, a dual-path graph convolution vertex classification framework, to effectively detect risky seller, based on both vertex connectivity relationship and topology structure similarity. We propose role-based convolution, a novel and generalizable model to enable the convolution process to be applied on additional vertex properties aside from connectivity. We evaluate our DP-GCN on four open network datasets and two synthetic datasets. We provide a study on three large-scale PayPal payment network datasets. Together with ablation study and embedding analysis, we show DP-GCN's effectiveness and interpretability for the risky seller detection task.

- **Cyber Anomaly Detection on CSP Network:** We propose BEA, a novel unsupervised learning framework for anomaly detection on dy-

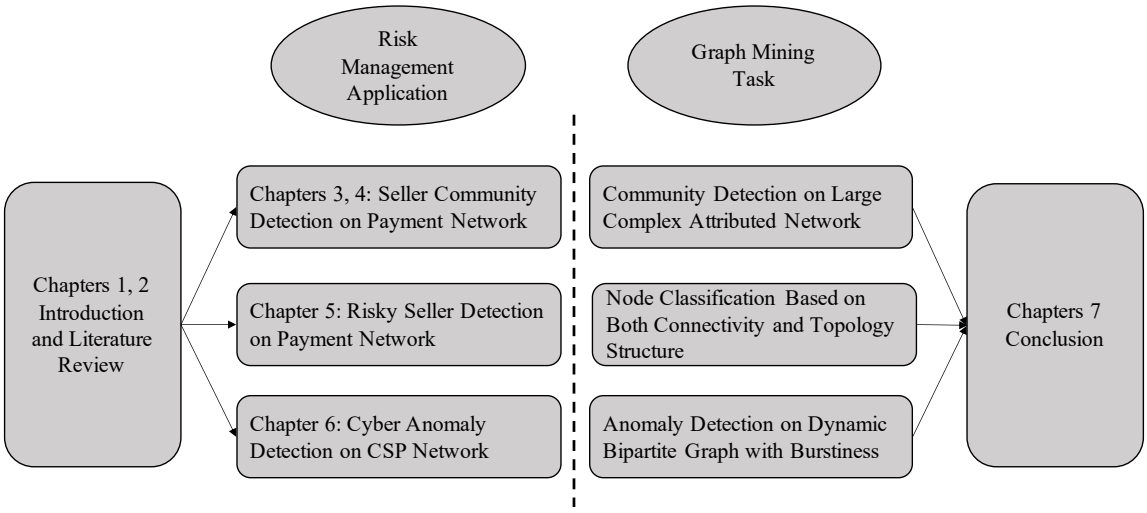


Figure 1.2: The organization of the thesis

namic bipartite graphs. It detects anomaly based on two important assumptions: burstiness and connectivity pattern. Experiments on three open datasets and PayPal CSP log network demonstrate the effectiveness of BEA. Further, we define two-step pattern embedding on bipartite graph, which enables graph convolution process on different types of vertices in bipartite graphs. We introduce a strategy to model vertices from real-world applications into source vertices and target vertices by their characteristics. This strategy motivates us to detect burstiness on sources and targets, which is more reasonable and effective.

1.4 Organization of the Thesis

The organization of the thesis is shown in Figure 1.2. The content for the rest of the thesis is outlined as follows:

- Chapters 3 and 4 present a study on graph community detection. In these chapters, we study the problem of community detection on real payment network. Both vertex connectivity and attribute are important for community detection. In these studies, we propose a

novel community detection framework AGGMMR and an extension Inc-AGGMMR to detect vertex community on large complex attributed network.

- Chapter 5 presents a study on graph vertex classification. Vertex classification formulated as a problem of learning the representation of vertex from graph. Most of the existing graph algorithms focus on exploring the vertex representation based on only connectivity, but ignore the vertex topology structure similarity. Vertex residing on different parts of the graph may have similar topology structure. Modeling both topology structure and connectivity is crucial for risky seller detection applications. In this chapter, we study the vertex classification on real risky seller detection and we propose a novel framework DP-GCN for vertex classification based on both vertex connectivity and topology structure.

- Chapter 6 presents a study on bipartite graph anomaly detection. Existing graph anomaly detection usually focuses on analyzing the vertex connectivity pattern. Thus it is important to detect anomalies that have inconsistent connectivity patterns with others. In this chapter, we study the graph anomaly detection on a real CSP log network. The anomaly on the CSP log network may arise from not only the connectivity pattern but also burstiness in network activities. Hence in this study, we propose an anomaly detection framework BEA that detects anomaly by studying both vertex connectivity pattern and burstiness.

- Chapter 7 concludes the thesis and outlines some promising future works.

Chapter 2

Literature Review

This chapter reviews the relevant research works in the domain of the three graph applications. We do a comprehensive review of all three applications. Section 2.1 reviews the algorithms for community detection. It includes community detection algorithms based on both attributes and graph connectivities. Section 2.2 reviews vertex classification algorithms for risky seller detection on network. Finally, Section 2.3 reviews the algorithms for anomaly detection on cyber network. Several algorithms on graph anomaly detection and burstiness detection are introduced as both techniques are important for the application.

2.1 Community Detection on Payment Network

To detect communities, an algorithm has to take advances from both attribute and connectivity information. We review three categories of community detection algorithms: connectivity-based, attribute-based, and community detection using both types of information. After that, we review the incremental community detection methods that are able to detect communities on dynamic network.

2.1.1 Connectivity-based Community Detection.

Assume we have a graph $G = (V, E)$. V denotes set of vertices and E denotes set of edges. The connectivity information on graph usually represented by an $|V|$ by $|V|$ adjacency matrix A as follows:

$$A_{|V|\times|V|} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

The i-th row of A represents the connectivity relationships of vertex v_i to $\forall v_j \in V$. For unweighted graph, $A[v_i][v_j] = 1$ if v_i is connected to v_j (as shown above); for weighted graph, $A[v_i][v_j] = w$ where $w \in \mathbb{R}^+$ is the weight on the connection. A classic method that using connectivity information is graph cut, which partitions the network by minimising the cost from cut. Minimum cut and normalised cut are the most well-studied algorithms in this area. Besides these two, edge betweenness can also be used to measure communities. One representative algorithm is Girvan-Newman algorithms [41]. Spectral clustering [92] analyses the spectrum of graph Laplacian matrix and partition the graph based on graph theory. However, due to high computational cost, those algorithms are not easily working on large-scale networks [90,103,118]. Matrix factorization methods [15,138] detecting community by decomposing connectivity matrix into community assignment matrix. Label propagation [152] finds communities by propagating labels within the network. It is scalable, but recent research shows it is not robust when running on many common scenarios [135]. Modularity functions, introduced by Newman and Girvan [91] have been extensively used to measure communities. In particular, modularity is a measure of the strength of the community structures. Network with high modularity have dense intra-community edges and have loose inter-community edges. The modularity Q is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - \frac{k_i k_j}{2m}] \delta(C_i, C_j) \tag{2.1}$$

Where m represents sum of the edge weights in the graph; A_{ij} represents the edge weight between vertices i and j; k_i and k_j represents the total edge weights attached to vertex i and j. $\delta(c_i, c_j) = 1$ if vertices i and j are in the same community, otherwise $\delta(c_i, c_j) = 0$. Modularity maximisation is a NP-hard problem and need to be solved by greedy method. Thus, a large number of heuristic algorithms have been developed [40].

One representative algorithm is Louvain [18], which has been widely used in industrial applications due to its scalability and high quality performance. Louvain algorithm contains two phases. At first phase, all vertices are assigned to its own communities. Then each vertex v_i is processed by removing it from its own community and added to each of its neighbours v_j' communities. The change of modularity can be calculated by:

$$\Delta Q = [\frac{\sum_{in} + 2k_{i,in}}{2m} - (\frac{\sum_{tot} + k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2] \tag{2.2}$$

Where $\sum_{in}$ is sum of all edges' weights within community that vertex v_i is adding to. $\sum_{tot}$ is sum of all edges' weights connect to vertices within community that vertex v_i is adding to. k_i is the weighted degree of vertex v_i and $k_{i,in}$ is the weighted degree between vertex v_i and other vertices within community that v_i is adding to. After the modularity change calculated for vertex v_i adding to all of each neighbours' communities, v_i is finally assigned to the neighbours' community that results largest modularity increase. If there is no possible modularity increase, v_i is remain in its current community. This process runs repeatedly with all vertices on the graph untill there is no modularity increase. At second phase, each community is grouped as a vertex. Thus a new graph is created with communities calculated in the first phase as vertices. Then the first phase is re-applied on this new network to get community assignment with a different resolution.

There are also solutions trying to partition communities from other perspectives. Those algorithms differ in the ways to measure distance based on network connectivity information, *e.g.,* random walk, graph drawing, and

game theory [14, 16, 30, 70, 82, 115, 119]. Those methods measure the connectivity similarity between vertices through different objectives. However, they are not extended to adapt to our complex attributed network settings.

2.1.2 Attribute-based Community Detection

Attribute-based community detection assumes the vertices in the same community are similar in attributes. Based on the natures of generated communities, attribute-based community detection methods can be broadly classified into following categories.

Similarity Measures Those methods use similarity function to measure the attribute similarity between vertices, then group the similar vertices into the same community. K-means is one of the most widely used similarity-based algorithms due to its simplicity and effectiveness. K-means algorithm firstly pick K vertices as community centroids. Then it assigns vertex to a community such that the euclidean distance between the vertex and the community's centroid is at the minimum. Finally it updates the attributes value of community centroids and repeat the process until there is no change to the centroids. There are a huge number of extensions [13, 20, 117]. Examples include kernel method on K-means to identify nonlinear structure in data [113], K-modes and K-prototypes that enable K-means to perform on categorical data [53]. In particular, the K-modes algorithm use mode instead of mean to update K centroids. The most frequently appeared categorical value will be updated as value to the centroids. In this way, categorical value can be directly clustered by the algorithm without any special encoding. The K-prototype algorithm integrates the k-means and k-modes algorithms to cluster the vertices with mixed-type features. The advantage of similarity-based community detection is relatively low time complexity and high computing efficiency in general.

Density Measures Density-based community detection methods assume that communities are defined by dense regions in the data space.

DBSCAN is the most well-known density-based algorithm. DBSCAN [35] classifies vertices into core points, border points and outlier points based on the number of neighbours a vertex has. Then the core points and their density-reachable border points are clustered into communities. To cope the efficiency problem of original DBSCAN, AnyDBC [83] is proposed which firstly compress data into smaller density-connected primitive communities, then perform density-reachable query to detect communities. In this way, AnyDBC requires substantially fewer queries compared to the original DBSCAN. EPDCA [74] further extends the DBSCAN algorithm to work on mixed-type features by using entropy for mixed-type distance measure. The advantage of density-based methods are the capability to discover communities of arbitrary shape. In opposite, their disadvantage is also obvious. They may fail in case of detecting varying density communities and are unreliable in high-dimensional data.

Hierarchical The idea of hierarchical community detection is to construct the hierarchical relationship among vertices in order to detect community. BIRCH [144], CURE [44], and ROCK [45] are the agglomerative method which assume each vertex is a community at beginning, then the closest vertices are merged to form new communities until there is only one community left. BIRCH summarizes data into Clustering Feature (CF) tree, thus is more efficient for community detection on large data. CURE uses scattered vertices to represent communities, in this way, it is able to capture communities with arbitrary shapes. ROCK algorithm merge vertices by considering the number of common neighbors between them. Thus it is applicable to categorical data. DIANA [60] and MONA [60] are divisive method which assume the entire network is a single community at beginning, then split it into smaller communities in each iteration. In specific, DIANA split the vertex with maximum average dissimilarity into a new community in each iteration and moves all vertices to that community if they are more similar to it than to the remainder. MONA only works on binary attribute data. In

each iteration, all communities are split according to the values of a single attribute. A community will be divided into one community with all vertices having value 0 for that attribute and another community with all vertices having value 0 for that attribute. communities are split until all vertices in the same community have same values for all attributes. The hierarchical community detection is easy to understand and implement. However, it usually is not efficient for large data.

Others Gaussian mixture model (GMM) is a probabilistic model that assumes all the vertices are generated from a mixture of a finite number of Gaussian distributions. Thus it estimates the probability density function from the data and assigns vertices into communities based on their probability to each mixture. Comparing to K means, GMM does not assume community to be of any geometry shape but it would be more difficult to interpret due to its complexity. Chameleon [59] construct K nearest neighbour (KNN) graph from data, then partition the graph into graphlets and finally merge those graphlets into communities. Self-Organization Map (SOM) [63] uses neural network to project data vertices into lower dimensional space. As typical SOM has a predefined fixed size map, growing SMO [11] is proposed to make it more flexible by growing the map with training data. GMixSOM [120] extends growing SOM for mixed-type features. The main drawback of SOM is that it requires necessary and sufficient neuron weights to cluster data, otherwise the result may not be that accurate.

Despite there are many different methods to detect communities based on attribute, all of these methods are designed to only focus on vertex attribute but neglect the information from the vertex connectivity. Thus those methods may fail to detect the communities in many real-life networks.

2.1.3 Connectivity & Attributes Community Detection

Recently, a few algorithms are proposed to detect communities based on both connectivity and attribute information. Those algorithms can be grouped

into three categories [36]: connectivity-based, attribute-based, and hybrid approaches.

Connectivity-based approach maps attribute information to connectivity structures. Then the attributed network community detection problem is transformed into a pure connectivity-based clustering problem. The key is to correctly map the attributes and use them to construct an augmented graph. SA-Cluster [150] and Hetero-Attractor [128] use dummy vertices to map attribute values. Each attribute value is represented by a single dummy vertex on the graph. Those dummy vertices are connected to all vertices in the network to retain their attribute relationships. Thus both methods are not that scalable for high dimensional data. CODICIL [108], SAC2 [27] use attributes to find nearest neighbours for each vertex, then reconstruct graph by connecting each vertex with their neighbours to retain attribute information. As KNN requires large space complexity, these methods are not suitable for large data either. In another hand, the attribute-based approach integrates connectivity information together with attribute distance/similarity to cluster the graph. In the propagation algorithm [73], it assumes vertices that belong to the same community should receive similar amount of content/attributes. So the problem is transformed into an optimisation problem that minimises the error between vertices' content propagation and expected content propagation in the respective communities. Communities can also be partitioned by embedding vertices through optimising both attribute distance and connectivity distance at the same time [69]. The CESNA [136] algorithm utilizes a generative model to predict community assignments. In [52], the method separates each attribute into a single layer of graph, then ensembles the results from each layer to get the whole graph clustering. Despite all of the above methods consider both types of information, they do not consider the importance between connectivity and attributes are varying when detecting different communities. Recently, various graph convolution networks (GCN) [61] (GCN is also reviewed in section 2.2.3) are proposed

to learn vertex embedding by aggregating and transforming attributes from its connectivity-based neighbours. Thus the embedding of vertex learnt by GCN will contain both attributes and connectivity information. Those embeddings can be directly used for community detection. However, the typical architecture of GCN is designed for general vertex embedding. Thus the performance of typical GCN on community detection task may not as stable as modularity maximisation. In summary, all these algorithms are designed to fuse attribute and connectivity information by leveraging different objectives. However, none of those methods explicitly tackle the problem of handling complex attributes in real-life network (at least the complex PayPal data in our applications). Besides that, how those algorithms adjust the weights between attributes and connectivity information in an unsupervised manner are not clearly discussed.

2.1.4 Incremental Community Detection on Payment Network

The incremental community detection is the community detection algorithm on the influx of new data that automatically formed new communities or updated the existing community [23]. Incremental K means [98] groups the new data and update new clusters to previous clustering results. In particular, it sets a distance threshold, if new data is farther from all centroids than the threshold, a new community will be created, otherwise, the new data will be clustered to one of the existing communities. Incremental DBSCAN is an extension of the classic DBSCAN clustering algorithm. Instead of using a traditional vector-based model, a graph-based model is used for data representation. When new data arrives, the graph structure can be easily updated by the graph model. Incremental DBSCAN is density-based method, thus does not work well on detecting varying density communities. Incremental Clustering based on Information Bottleneck theory (ICIB) [75] has similar procedure as incremental K means. The first data is selected randomly and

clustered as one community. It measures the loss of mutual information when adding new data to existing clusters. There are also connectivity-based incremental algorithms. As one of the modularity-based algorithms, [95] adds new edges to the graph one by one and calculates modularity gain under different cases. It adjusts community assignments for vertices when new edges coming. This algorithm chooses different strategies to update different types of incoming edges, however, there is no theoritical guarantee on the effectiveness of this framework. QCA [93] tracks each vertex's adding or removing as a simple event and updates community assignment accordingly. The algorithm proposed by [143] treats dynamic graph as a sequence of graph snapshots to detect communities. Recently, the authors of [86] introduce non-negative matrix factorisation to find communities on online graph based on both attribute and connectivity information. However, it is not that efficiency for running on large data. The GraphSage [47] is an extension of GCN and it is able to learn the vertex embedding on incremental network. It learns functions that generates vertex embedding by sampling and aggregating attributes from vertex connectivity-based neighbours. However, as GraphSage is originally designed for general vertex embedding, it may not works as stable as modularity maximisation for community detection task.

Given a large-scale network with complex attributes in our problem setting, existing algorithms are not (at least not directly) applicable. Those algorithms require all vertices to have same set of attributes. Besides that, how those algorithms adjust the weights between attributes and connectivity information in an unsupervised manner is not clearly discussed. Due to the difficulty of detecting communities in an incremental manner, it is even harder to have an algorithm that fulfills all those requirements.

2.2 Risky Seller Detection on Payment Network

We aim to model both vertex connectivity relationship and local topology structure similarity to detect risky seller. This application can be directly modelled by vertex classification task. Thus our research is related to studies in four areas: vertex connectivity embedding, vertex topology structure embedding, graph neural network and graph fraud detection.

2.2.1 Vertex Connectivity Embedding

The idea of vertex embedding is to transform vertex into low dimensional vector representations. The vector representations of vertices can be directly feed into downstream data mining tasks, *e.g.,*risky seller detection. Most of the recent vertex embedding methods for graphs are largely based on the skip-gram model [24, 88, 89]. In particular, Deepwalk [97] follows this approach to embed vertices to preserve vertices' neighborhood relationships through random walk. Node2vec [43] optimizes this approach by adopting breadth-first and depth-first search in the random walk process. These methods have become increasingly popular in both industry and research communities, as they outperform many existing methods with similar objectives. The idea of the skip gram model in language is that words with similar meaning is surrounded by a similar context. However, this assumption may not be always true for graph. Unlike words are universal with semantic meanings, graph vertex are not universal and only meaningful in a particular graph. Thus those vertex connectivity embedding methods usually capture proximity among the vertices, which works on isolated graphs and unable to generalize to unseen vertices [7]. Besides skip gram-based methods, there are vertex embedding methods trying to preserve particular properties. SDNE [126] introduces a deep model for vertex embedding. It adapts deep autoencoder with multiple non-linear layer to preserve the neighbor structure of vertices.

ComE [21] embeds vertex with community information. SiNE [127] uses two deep networks to learn the embedding and preserve the network structure by considering difference between positive edge and negative edge in signed graph. Due to the characteristics of vertex connectivity embedding, those methods only focus on the connectivity relationship between vertices; hence these methods do not consider the roles of vertices based on their local topology structures in the networks.

2.2.2 Vertex Topology Structure Embedding

Since connectivity of vertices does not guarantee their similarity, the idea of topology structure is more effective to model the structural relatedness among vertices. Recently, many methods are proposed to capture the topology structure of vertices in embedding. Struc2vec [104] constructs a multilayer weighted graph that encodes the topology structure similarity between vertices. If two vertices have similar degree and their neighbors also have similar degree, then these two vertices are similar in topology structure. However, the struc2vec algorithm suffers from high computational complexity. Graphwave [33] captures the topology structure of vertices by leveraging spectral graph wavelet diffusion patterns. It assumes that vertices with different topology structures have different ways of diffusion spreading. The work in [9] shows graphwave is suffered from high space complexity when applied to large graphs. RoIX [50] extracts topology structure related features for vertices *e.g.,* average neighbour degree, maximum neighbour degree, then performs soft-clustering to cluster vertices based on those features. The drawback of this algorithm is that those features are manually engineered. DeepGL [107] constructs role features by local graphlet decomposition. It learns graph functions where each represents a composition of relational operators that are applied to a graphlet/motif feature. Hone [106] describes a framework based on weighted k-step motif graphs to learn low-dimensional

topology structure embeddings. Role2vec [8] derives topology structure features by computing motif patterns of vertices. It then computes random walk transition probabilities based on topology structure similarities and embeds vertices using feature-based random walk. All of these methods analyze the graphlet/motif of vertices to learn their topology embedding. The main drawback of these algorithms are that they are not feasible to compute large size motifs/graphlet on graph due to the high complexity. Recently, Riwalk [78] is proposed to model vertex topology structure by its degree and relative position. It can better integrate graph kernels with vertex embedding methods to leverage recent advancements in vertex embedding. In our risky seller detection application, risky seller can be detected by not only through their transaction topology similarity, but also their transaction connectivity relationships. Thus topology structure embedding need to be properly integrated with vertices connectivity embedding in order to effectively detect risky seller in our task.

2.2.3 Graph Neural Networks

With the advancement of deep learning, graph neural networks, especially graph convolution networks (GCN) have gained significant research interests [38, 47, 61, 76, 112, 124]. As described in section 2.1, we use notation $G(V, E)$ to represent graph and symbol A to represent adjacency matrix. Let X^{l+1} denote the embedding of vertex i in the l^{th} layer. Then the graph convolution can be formulated as:

$$X^{l+1} = \sigma(WX^l\hat{A}) \tag{2.3}$$

where $\hat{A}$ is symmetrically normalized from A. σ denotes the non-linear transformation. W is the weight matrix.

For the task of vertex classification, the state-of-the-art solutions are dominated by GCN models. Different from skip-gram, GCN learns vertex representation by aggregating vertex's local neighbourhood through functions.

GraphSage [47] extends the vanilla GCN by using neighborhood sampling during the aggregation process. It introduces several aggregation functions such as "Mean Pooling", "Max Pooling" and "LSTM function". Besides, GAT [124] introduces self-attention mechanism to graph neural network. The self-attention measures the importance of different vertices in neighborhood when performing aggregation. However, the recent work [19] shows GAT can only compute a restricted kind of attention which hinders its expressiveness. DIFFPOOL [139] utilizes GCN with graph pooling to embed graph with hierarchical information. It is useful for entire graph classification as it considers hierarchical graph structure. LGCL [38] ranks and selects fixed number of neighboring vertices for each feature to enable typical convolution operation. Cluster-GCN [25] samples subgraph by clustering. It is capable to handle large data by training with this subgraph sampling strategy. DEMO-Net [133] models vertex degree for vertex embedding. In specific, it introduces degree-specific mapping function for feature aggregation in the convolution. It is not designed for vertex topology learning originally, but the vertex degree it modelled is correlated to the vertex topology. There are also many other extensions of GCN for specific domains, *e.g.,* GEM [76] for anomaly account detection, and R-GCN [112] for relational graph. HeteGraph [12] adapts GCN for heterogeneous graph and proposed two different architectures for recommendation problem. T-GCN [145] integrates GRU to model temporal dependences of traffic graphs for traffic forecasting. Those methods are designed for specific applications thus their architectures are adjusted to better fit the corresponding problems. In summary, GCN has huge advantage to learn the vertex representation which is directly linked with its architecture. However the GCN is still somewhat of a not fully explored field even it has already provide huge benefits in graph applications.

Recently, there are also studies trying to leverage dual-path convolution architecture. The work from [154] builds a PPMI matrix as the second path adjacency matrix to model graph global consistency. The authors in [149] use

a dual-path architecture to embed image and text. It is a non-graph model but leverages dual-path to capture the relationship between image convolution and text convolution. AM-GCN [129] models vertex features with K-nearest neighbour (KNN) graph and propagates vertex features through both feature space and connectivity space simultaneously to do better information fusion. Despite both methods proposed dual path structure, they model the graph connectivity but not topology for vertex embedding. DC-GCN [51] is a recent work for image question answering. It uses dual-path architecture to combine image and text for better answer prediction.

Graph convolution is a natural way to model the relationship between vertices. Various GCN models are developed to tackle different problems. Different from all these works, our proposed model considers both vertex connectivity and vertex topology structure similarities to tackle our application. In particular, the role-based topology convolution module in our model is a novel convolution module. This new module is highly generalizable and can be used to model other types of vertex properties. Moreover, the dual-path architecture designed in our model enables the two paths to be interactive, through a self-attention mechanism.

2.2.4 Graph Fraud (Risky Seller) Detection

Most of graph-based fraud detection focuses on analyzing graph connectivity and communities [100]. The system in [131] adapts HITS algorithm to detect fraudsters by propagating fraud score through graph. The limitation of such method is that it is not able to detect fraudster in unseen communities in the graph. CatchSync [56] detects fraud by analyzing vertex properties (*e.g.,* degree, HITS score, edge betweenness). The works [17, 85, 130] detect fraud by analyzing fraudulent communities. GCNEXT [64] is a detection model using expanded balance theory and GCN. In summary, those methods assume fraudsters in network are usually connected in communities, thus they are able to detect fraudsters by exploring vertices connectivity relationships. In

our work, through extensive experiments, it is observed that modelling the topology structure of vertices is a more important and effective direction to detect fraudsters (risky seller). Thus in our work, we proposed a novel model that on top of GCN to process both vertices connectivity and topology structure for fraud (risky seller) detection.

2.3 Anomaly Detection on CSP Network

Our research is related to studies in three areas, namely, general anomaly detection methods, vertex embedding (already reviewed in 2.2), and burstiness detection. In the following, we review the related studies in each area.

2.3.1 General Anomaly/Outlier Detection Methods.

In general anomaly detection, anomalies are defined as patterns in data that do not conform to normal behaviours and they may be caused by malicious activities or intrusions [6].

Among the many solutions proposed for anomaly detection, one-class support vector machines (SVM) is a widely used classification-based technique as it works effectively for imbalanced classification. It has been the winning solution for many tasks in detecting intrusion anomaly [122]. However, the SVM approach suffers from high time complexity when perform on large data. Another technique is Isolation Forest [72], which detects anomaly purely based on the concept of isolation without employing any distance or density measure. It builds an ensemble of isolation trees, then the anomalies are those instances that have the highest susceptibility to be isolated. Isolation Forest requires less memory and detects anomaly fastly. But from empirical test, it usually does not bring the optimal result. Clustering-based method groups similar data objects into clusters. Then anomalies are identified if they do not belong to the clusters of the majority. Gaussian Mixture Model (GMM) is one of the effective clustering-based techniques with many

variations to detect anomaly through probability density analysis. Recently, GMM is combined with deep learning to detect anomalies *e.g.*,deep autoencoding GMM (DAGMM [155]). DAGMM leverages GMM over the learned low dimensional embedding to deal with density estimation tasks and shows stronger ability to differentiate anomalies, compared to conventional GMM. In particular, it adapts an autoencoder network architecture to learn the embedding, and then uses GMM objective function to train the network. The network output is the score that represents mixture memberships. In summary, those methods provide varies advantages when detecting anomaly by features. However, they are not directly applicable to detect graph anomaly unless graph data is transformed by additional embedding methods. For graph anomaly detection, it is important to detect structural anomalies that have inconsistent connectivity patterns with others. A simple approach is to extract useful features from host degrees, ego network density, or community assignment [109, 110]. These methods require manually engineer the features. Goutlier [5] detects anomalies that contain unusual bridging edges between two different communities in a graph. This method is therefore sensitive to the results of graph clustering. ANOMRANK [140] detects anomaly by measuring the changes of graph from two perspectives: sudden change in edge weights, and sudden change in graph structure. However, this method may suffer from the problem of cold start when many previous unseen vertices joining the graph. Netwalk [141] detects anomalies on dynamic graph through clustering. It firstly learns the representation of vertex by clique embedding, then utilizes K-means clustering algorithm to detect vertex anomaly that does not belong to any existing clusters. Addgraph [147] assumes the temporal feature is important. It uses GRU to encode short-term and long-term patterns of vertices. In summary, non-graph anomaly detection methods focus on metadata-related features. They usually are not designed to detect the types of anomaly in graph. In opposite, graph-based anomaly detection methods aim to detect anomaly by identifying abnormal

vertex connections or communities. In our task of detecting anomaly in PayPal CSP log network, it is important to analyze both vertex connectivities and vertex activity (especially burstiness) due to the characteristic of this particular anomaly type.

2.3.2 Burstiness Detection

Burstiness detection is to detect an unexpectedly large number of events within a period of time. It usually indicates occurrences of important or abnormal events. There are two types of typical burst detection methods: threshold-based methods [49] and state-based methods. Threshold-based methods detect burst by analysing event frequency with predefined statistical thresholds [49, 66, 153]. The drawback of threshold-based method is that it requires to predefine the threshold and sometimes the optimal threshold is difficult to be found. On the other hand, Kleinberg's burst model [62] is one of the classic state-based models. It models bursts with an infinite state automaton in which each state represents a level of burst. Similarly, Li *et al.* [67] proposed a labeled Hidden Markov model to capture burst by measuring the time interval between two events. Recently, there are also works trying to combine burst detection techniques with graph algorithm. Zhao *et al.* [146] adapt burst detection to analyse evolving of graph. To the best of our knowledge, our work is the first that adapts burst event detection framework with dynamic graph algorithm to address anomaly detection on dynamic bipartite graphs [1].

[1] PayPal CSP log network can be naturally modelled by bipartite graph with external domains and internal servers

Chapter 3

Seller Community Detection on Payment Network [1]

3.1 Introduction

Network data in the industry can be very complex with a large number of vertices of different types. Further, vertices may be associated with different types of attributes (*e.g.,* numerical and categorical) with missing values. In PayPal payment network, a vertex represents a seller or buyer while an edge represents their transaction relationships. There are attributes associated with sellers describing their business but not buyers. Understanding community structures in such a network benefit many applications such as risk management, compliance management, and targeted marketing. However, finding communities in such an attributed network is not an easy task. Most state-of-the-art algorithms only consider either attributes or network connectivity information, but not both types of information. On the other hand, the optimal balance of using both types of information is unknown because community detection is unsupervised in nature. To make the task even more challenging, vertex attributes may provide contradictory information with connectivity structure [111].

[1] The work in this chapter has been published in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2041-2049), 2019.

In this chapter, we propose a community detection framework named **AGGMMR** (Augmented Graph with Modularity Maximisation and Refinement) to detect communities from large-scale attributed graph. Our framework is designed to partition an attributed graph based on both its attributes and connectivity information through a greedy modularity maximisation model. It consists of three phases: (i) augmented graph construction and weight initialisation, (ii) weight learning with modularity maximisation, and (iii) modularity refinement. Specifically, an augmented graph is constructed through centre-based attributed clustering to retain attribute relationships between vertices. All attribute relationships are then transformed into edges in the augmented graph. The modularity maximisation model is then applied to partition the augmented graph, which now contains both attributes and connectivity information. Along with the partition, a fast learning algorithm is proposed to automatically adjust weights on those attributes relationships according to their contributions towards our objective. Because the result of a greedy modularity maximisation model is heavily affected by its processing order, we propose a modularity refinement algorithm to automatically optimise the result through a fast greedy search.

We evaluated AGGMMR on four open networks which are publicly available, CiteSeer, Cora, Terrorist Attack, and SinaNet networks. Then, we evaluate AGGMMR on PayPal payment network to detect seller communities. The PayPal network consists of 100 *million* vertices and 1.5 *billion* edges. Among the 100 million vertices, 3 *million* are sellers which are described by 68 attributes. Running community detection algorithm on such a large attributed network is challenging for several reasons. First, the network is very large. Many existing algorithms are not scalable enough to be applied on this level of scale. Second, the network contains heterogeneous vertices. There are both sellers and buyers, and business-related attributes are only available for large sellers. Algorithms that require all vertices to share the same set of attributes cannot be applied. Third, there are both

numerical and categorical attributes. Further, the categorical attributes can be many-value or multi-value (details in our case study).

Through the evaluation on public datasets and the large-scale PayPal networks, we show the effectiveness of AGGMMR, and its superiority over 7 baselines Louvain [18], Distance Dynamics [115], NAGC [79], CODICIL [108] K-means, GCN [61] and GraphSage [47].

We summaries the main contributions of the work as belows:

- We proposed a novel community detection framework to detects communities in complex attributed network through modularity maximisation.
- Complexity analysis is provided to show the good scalability of the framework.
- We perform evaluation of the framework on open datasets, synthetic dataset and the large scale PayPal network to show its effectiveness against 7 baselines. Besides that, we also conduct analysis of weight change and modularity refine to study the learning process of the framework.

3.2 Graph Definition

A network **graph** can be represented as $G = (V, E)$, where V denotes the set of vertices and E denotes the set of edges. The graph connectivity information can be represented by a $|V|$ by $|V|$ adjacency matrix A_c, where $A_c[v_i][v_j] = 1$ if v_i and v_j are connected. Further, each vertex can be associated with vertex features. We use $F \in \mathbb{R}^{|V| \times d}$ to denote the feature matrix, where d is the feature dimension. Usually, vertex features are the features that are specific to each individual vertex, *e.g.,* business features of sellers in a payment network.

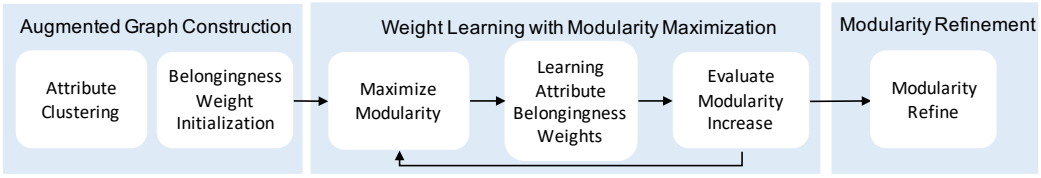


Figure 3.1: Flow of Framework

3.3 The AGGMMR Framework

Our AGGMMR framework is designed to partition an attributed graph based on its attributes and connectivity information, through a greedy modularity maximisation model.

Illustrated in Figure 3.1, the framework consists of three steps. We first construct an augmented graph through a center-based attributed clustering to retain attribute relationships between vertices. In the second step, the modularity maximisation model is used to partition the augmented graph. A fast learning algorithm is proposed to automatically adjust weights on attribute relationships according to their contributions towards our objective. In the last step, we propose a modularity refinement algorithm to automatically optimise the result through a fast greedy search. This is to reduce the effect of processing order in the greedy modularity maximisation model in the previous step. Next, we detail the three steps.

3.3.1 Augmented Graph Construction

One way to retain attributes information on the graph is to plot all attribute values as dummy vertices and connect them with original vertices [150]. However, this method is applicable when attributes are all categorical and the number of attributes is relatively small.

Example 3.3.1. *Assume there are 100 attributes each with 10 different values in a graph consisting of* $1M$ *vertices. This method will generate* 100×10 *dummy vertices and* $100 \times 1M$ *dummy edges.*

Instead of directly using attribute values as a dummy vertex, we propose to summarize the values into attribute centers. Specifically, we perform center-based attribute clustering on all vertices that have attributes. A dummy vertex is created to represent the center of each attribute cluster (*i.e.,* **attribute center**) and is added to the graph. Then, **belongingness edges** between those attribute centers and their member vertices are added to the augmented graph to retain their attribute relationships.

Example 3.3.2. *Assume there is a graph with* $1M$ *vertices and* $10,000$ *attribute centers. Only additional* $10,000$ *dummy vertices and at most* $1M$ *belongingness edges are needed to construct an augmented graph. Useful attribute relationships are effectively captured in this much smaller augmented graph.*

Note that our construction method is not limited to a graph with only categorical attributes, but all types of attributes as long as the attributes are available for clustering. Our method is also flexible because all kinds of center-based attribute clustering algorithms can be easily adopted here, to cater to different application requirements.

To indicate the strength of belongingness relationship between each vertex and its nearest attribute center, we use *attribute distance* to initialize the weight of a belongingness edge. An **attribute distance** is the distance between each vertex and their nearest attribute center, calculated by the center-based attribute clustering algorithm. For example, we can use Euclidean distance if k-means algorithm is used to cluster attribute values. We denote attribute distance by $d(v_i, v_c)$ between vertex v_i and attribute center v_c.

In our implementation, we first compute Euclidean distances as attribute distances, and then map the distances into probability values. Specifically, we map Euclidean distances into higher dimensional space using the RBF kernel.

$$P(v_i, v_c) = exp(-\frac{d(v_i, v_c)}{2\sigma^2}) \tag{3.1}$$

As the kernel distance embeds isometrically in a Euclidean space, the RBF kernel function is an effective metric for weighting observations [99]. Then the **weight initialization** on belongingness edges are by Equation 3.2.

$$w(v_i, v_c) = dt(v_i) \times P(v_i, v_c) \tag{3.2}$$

Here, $dt(v_i)$ is the weighted degree of vertex v_i in the connectivity graph, *i.e.,* the original graph before adding attribute centers as dummy vertices. This weighting scheme is designed to balance the weights between attribute information and connectivity information for each vertex in the augmented graph at the initial stage.

3.3.2 Weight Learning with Modularity Maximisation

In the augmented graph, both attributes and connectivity relationships are represented by edges. We, therefore, use a connectivity-based clustering method to partition the augmented graph. Intuitively, densely connected vertices should be in a community as they share either strong attributes or strong connectivity relationships, or both. Modularity maximisation is one of such objective functions that have shown to be effective [36, 101, 115]. We adopt the modularity maximisation model from [91]

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - \frac{da(v_i) \times da(v_j)}{2m}] \delta(v_i, v_j) \tag{3.3}$$

where m corresponds to the cardinality of edges in the graph, $da(v_i)$, $da(v_j)$ are the weighted degrees of vertices v_i and v_j in the augmented graph, respectively. A_{ij} is the ij-th component of the adjacency matrix of the graph, and A_{ij} equals to edge weight if vertices v_i and v_j are adjacent, and 0 otherwise. $\delta(v_i, v_j)$ equals to 1 when v_i and v_j belongs to the same community, and to 0 otherwise.

In our implementation, we adopt the Louvain [18] algorithm for modularity maximisation. In the beginning, each vertex is assigned with an individual

community. In every iteration, each vertex is compared with its neighbours' community assignments, and assigned to the one with maximum modularity gain. The computation of modularity gain is based on the weights of the edges and we refer to [18] for details.

On augmented graph, vertices are partitioned on both attributes and connectivity relationships. Since both types of relationships are represented by edges, there are three situations that two vertices are assigned to the same community through modularity maximisation: (i) They are densely connected and have strong attribute relationships. (ii) They are densely connected but they have trivial attributes relationships. (iii) They are not densely connected but their attribute relationships are strong enough to connect them.

In a real-life network, some attribute relationships could be trivial for many communities, *e.g.,* the sellers who have loans are connected based on this 'haveLoan' attribute and this attribute is only useful when clustering business loans community. We want to minimise the influence from such attribute relationships when it is trivial. At the same time, we aim to increase the importance of meaningful attribute relationships. To this end, we propose an **unsupervised weight learning algorithm**, align with modularity maximisation objective, to automatically adjust the weights for attribute relationships according to their contributions in the clustering.

Assume most of the vertices from the same attribute center have been assigned to the same community in an iteration, then it indicates that the attribute-based relationship from this attribute center provides a positive contribution in our community detection task. In opposite, if most of the vertices from an attribute center have been assigned to a large number of different communities, then this attribute-based relationship is very weak and might introduce noise to our task. The weights of belongingness edges to attribute centers therefore can be adjusted accordingly. To update weights

of belongingness edges, we first calculate a clustering contribution score for each attribute center.

The **contribution score** for an attribute center, denoted by Θ_a is calculated through $\Theta_a = \frac{|V_a|}{|C_a|}$. V_a is the set of vertices that connect to this attribute center; C_a is the set of communities that the member vertices in V_a are assigned to, through modularity maximisation in the current iteration. The value of Θ_a is bounded between 1 to $|V_a|$ as $|C_a|$ varies from 1 to $|V_a|$. The more vertices an attribute center connects, the higher potential contributions this attribute center will have. That is, an attribute center connecting to 10,000 vertices and all its vertices distributed in the same community contributes more than an attribute center who connects only 10 vertices in the same situation.

To meet the constraint that the total edge weights does not change, *i.e.,* $\sum_{i=1}^{n} w_i^{t+1} = \sum_{i=1}^{n} w_i^t$, where w_i^{t+1} is the weight of a belongingness edge in iteration $t+1$, we redistribute the weights belongingness edges as follows:

$$w_i^{t+1} = \frac{1}{2}(w_i^t + \delta w_i^t) \qquad \delta w_i^t = \frac{\Theta_a}{\Sigma\Theta} \times W \qquad W = \sum w^t \tag{3.4}$$

Theorem 1. *In each iteration, the weights are adjusted towards the direction of increasing the modularity objective*

Rewriting the modularity maximisation (Equation 3.3) on this augmented graph, we have

$$\begin{aligned} Q &= \frac{1}{2m}(Q_s + Q_d) \\ Q_s &= \frac{1}{2m}\sum_{lk}[A_{lk} - \frac{da(v_l) \times da(v_k)}{2m}]\delta(v_l, v_k) \\ Q_d &= \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{da(v_i) \times da(v_j)}{2m}]\delta(v_i, v_j) \end{aligned} \tag{3.5}$$

where v_l,v_k are vertices belonging to a same attribute center and v_i,v_j are vertices belonging to different attribute centers. $\delta(\cdot, \cdot)$ is the same as in

Equation 3.3, and its value is 1 if the two vertices are in the same community and 0 otherwise.

The modularity of a graph is in proportion to the sum of differences between connections and expected connections from every pair of vertices that are in the same community. In the above equation, we use Q_s to represent the sum of modularity calculated from the pairs of vertices in the same attribute center and use Q_d to represent the sum of modularity calculated from the pairs of vertices in different attribute centers. In weight learning, it only affects the modularity of Q_s while not the modularity of Q_d. Note that in each iteration, an attribute center will also been assigned to one of its member's communities according to its relationships with its members. When we increase the weights of attribute relationships from an attribute center, we are increasing A_{lk} between the member vertices to that attribute center. In this way, we increase Q_s more as most of the vertices are likely to be assigned into the same community with their attribute center. In opposite, when we decrease the weights of attribute relationships, Q_d decreases less since most of the vertices connect to that attribute center are assigned into different communities. After each iteration, the modularity will be compared with the value in the previous iteration. The learning algorithm converges once there is no more modularity increase in an iteration.

3.4 Modularity Refinement

We now present the last step in the AGGMMR framework, modularity refinement, to optimise our partition result from the greedy modularity maximisation model.

3.4.1 Drawbacks of Greedy Method

Greedy modularity maximisation reduces the computation cost significantly, however, the quality of result highly depends on the order of processing.

The greedy modularity maximisation process can be demonstrated from the perspective of dynamic programming. To find the community assignments which maximise the global modularity of a graph G, in each step, we assign a vertex into one of its neighbours' communities. Naturally, the subproblems of this dynamic programming are to find optimal community assignments for previous n vertices. This recursion can be expressed as

$$Q[n] = max(Q_1[n-1] + q_{n-1,n}, Q_2[n-2] + q_{n-2,n}.....) \tag{3.6}$$

The $Q_k[n-k]$ represents the optimal community assignments for previous $n-k$ vertices while $q_{n-k,n}$ represents the later k vertices' assignments. In this greedy method, this recursion is

$$Q[n] = Q_1[n-1] + q_{n-1,n} \tag{3.7}$$

The community assignment of the current vertex depends on the previous assignments. We assume the previous assignments for $n-1$ vertices are always the optimum solution even after we have assigned the current vertex. However, this is not true and the assignments for initial vertices are not always reliable. In the first iteration, when we process the first half vertices, we have no information about the remaining vertices' community assignments. That is, the greedy method only takes very limited information when it processes most of the vertices in the network. This process may easily trap the result into a huge different local optimum due to a domino effect. In following iterations, even we have already obtained all vertices' community assignments, the situation will not get better due to a *mutual effect*.

Assume vertices v_b and v_c are processed after vertex v_a, and both are assigned into the same community with v_a. When we re-evaluate v_a's community assignment, the community assignments of vertices v_b and v_c will back affect vertex v_a's community re-assignment by keeping it from re-assigning to other communities. We define such effect as **mutual effect**. We use an example in Figure 3.2 to further illustrate this.

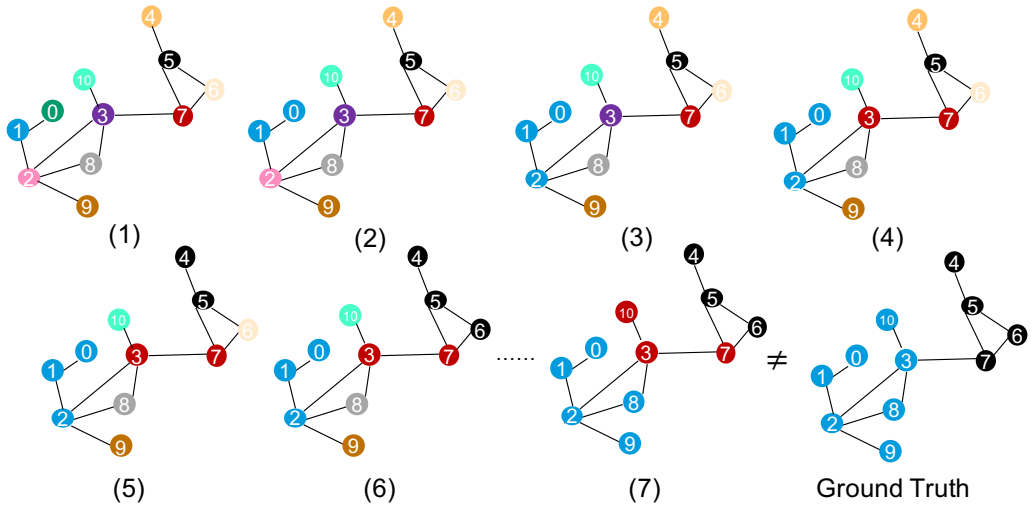


Figure 3.2: The effect of processing order in an example graph (edge weight not indicated in the graph, best view in color).

In Figure 3.2, the number on each vertex represents its order of processing. Each colour represents an individual community assignment. At step (1), each vertex is initialised to its own individual community. Then every vertex is compared with its community assignment with neighbours' communities according to its modularity gain. When processing v_3 in step (3), it has neighbour vertices v_2, v_7, v_8 and v_{10}. At that moment, vertices v_8, v_9 and v_{10} are in their individual communities because they have not been processed.

In the ideal result, indicated by ground truth, vertices v_8, v_9 and v_{10} are in the same community with v_2 according their connectivity. Thus v_3 should also be assigned to the same community for its dense connections with v_2, v_8, v_9 and v_{10}. However, at step (3), we do not have those information. In the end, v_3 is assigned to the same community with v_7. This assignment also influences further assignments after we processing v_3. As in this example, vertices v_3, v_7, and v_{10} eventually form a community in step (7).

This scenario happens frequently since the earlier assignment of a vertex, the less information can be used for that assignment. A single edge with a large weight will easily trap two vertices into a bad assignment when we do

not have enough information. In a payment network, one occasional transaction involving a large amount will trap two sellers into the same community and thus affect further assignments.

3.4.2 Modularity Refinement Algorithm

Completely getting rid of the aforementioned scenario *i.e.,* finding the optimum solution for a modularity maximisation problem, is np-hard and not practical. However, we can do an effective greedy refinement to optimise the result as much as possible. The idea of refinement is to give each vertex a second chance to re-assign its community after we have the whole picture of assignments, and at the same time to eliminate mutual effect when re-assigning a vertex.

In refinement, all vertices' community assignments will be re-evaluated in the same order from the previous iteration. When re-evaluating a vertex v_i, v_i will be compared with three types of neighbours: (i) a neighbour has the same community assignment as v_i, assigned before v_i's assignment, (ii) a neighbour has same community assignment as v_i, assigned after v_i, and (iii) the neighbour has different community assignment from v_i.

We temporarily mask a neighbour who is assigned the same community after v_i an individual community. The masked vertices are those whose community assignments are directly affected by the current vertex v_i in the greedy modularity maximisation process. Note that those vertices processed after v_i but have different community assignments from v_i are not masked. Now, we re-evaluate v_i's community assignment. If v_i's assignment is changed to one of its neighbors, say v_n's assignment, because of the largest modularity gain, then we have two possible cases. If v_n is a masked neighbor, then v_i keeps its original assignment, *i.e.,* v_i's community assignment is unchanged during the re-evaluation. However, if v_n is not a masked neighbor, then v_i will be re-assigned to v_n's community.

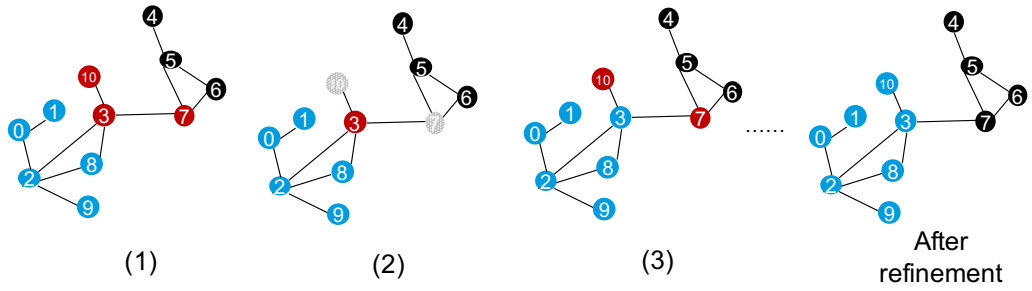


Figure 3.3: Example of Refinement (best view in color)

Figure 3.3 provides an example to illustrate how refinement works. In ground truth, vertices v_3, v_{10} belong to the blue community on the left, while vertex v_7 belongs to the black community on the right. In step (1), we have community assignments of all vertices from greedy modularity maximisation, and vertex v_3 is assigned to a community with vertices v_7 and v_{10}. When we re-evaluate vertex v_3 in step (2), vertices v_7 and v_{10} will be temporarily masked to individual communities, because they originally were processed after v_3 and at the same time they are in the same community as v_3. After the masking, there will be no longer mutual effects between vertices v_3, v_7, and v_{10}. After re-evaluation, v_3 is re-assigned to the blue community on the left, since joining into it produces a larger modularity gain than joining either community of v_7 or v_{10}. The relationship between either v_7 or v_{10} to v_3 is not strong enough to continue keeping v_3 in their communities once we eliminate the mutual effect.

To summarize, through this refinement process, we eliminate the mutual effect but retain all other useful information on the graph as much as possible. If a vertex is re-assigned to one of those masked neighbours' communities again, it indicates these vertices have strong relationships to bound them in the same community.

Table 3.1: Statistics of the four open networks

Dataset	Vertices	Edges	Attributes
Citeseer	3,312	4,732	3,702 binary attributes
Cora	2,708	5,429	1,433 binary attributes
Terrorist	1,293	3,172	106 attributes
SinaNet	3,490	30,282	LDA topics
Synthetic	512	3,242	synthetic attributes

Table 3.2: Evaluation on Open Network Datasets.

	Purity				
Dataset	Citeseer	Cora	Terrorist	SinaNet	Synthetic
Louvain	<u>0.7355</u>	0.7651	0.7318	0.2506	**0.7225**
DDynamic	0.5041	0.5194	0.7318	0.3265	0.5395
K-means	0.5018	0.5255	**0.8118**	<u>0.7684</u>	0.5949
NAGC	0.7271	0.7858	0.7069	0.5484	0.4234
CODICIL	0.6760	<u>0.7991</u>	0.7349	0.6550	0.6243
GCN	0.6111	0.7323	0.7380	0.4947	0.6250
GraphSage	0.3898	0.4446	0.7001	0.3068	0.6358
AGGMMR	**0.7656**	**0.8342**	<u>0.7504</u>	**0.8131**	<u>0.7207</u>

3.4.3 Time Complexity Analysis

The whole framework is fast and scalable with a time complexity of $O(n \log n)$, where n is the number of vertices. For augmented graph construction phase, the running time required largely depends on the chosen attribute-based clustering algorithm. The good thing is, the attribute-based clustering algorithm is able to run separately on attributes. The time complexity for our framework itself is linear to the number of attribute centers and belongingness edges. In the phases of modularity maximisation and modularity refinement, it has the same time complexity with the Louvain algorithm, runs in $O(n \log n)$. For the phase of weight learning, it runs in $O(n)$ as it updates weights for every vertex iteratively.

Table 3.3: Evaluation on Open Network Datasets

	Fscore				
Dataset	Citeseer	Cora	Terrorist	SinaNet	Synthetic
Louvain	0.7250	0.7567	0.5498	0.1924	0.7188
DDynamic	0.5896	0.4878	0.5534	0.3380	0.4590
K-means	0.4953	0.5060	0.5706	0.7462	0.5949
NAGC	0.7001	0.7736	0.5265	0.4610	0.4199
CODICIL	0.6595	0.7800	0.5660	0.6051	0.6250
GCN	0.5848	0.7251	0.5005	0.3965	0.6206
GraphSage	0.3883	0.4446	0.5836	0.2666	0.6314
AGGMMR	**0.7501**	**0.8238**	**0.5892**	**0.7653**	**0.7207**

3.5 Experiment on Open Networks

We conduct experiments to evaluate AGGMMR on four publicly available networks, against seven baselines. All the four open networks come with ground truth for validation. For comprehensiveness, we also conduct experiments on a synthetic network generated by acMark algorithm [80].

3.5.1 Datasets and Evaluation Metrics

Open Network Datasets. Listed in Table 3.1, **CiteSeer** is the network extracted from CiteSeer digital library [114]. Each vertex represents a publication and an edge indicates a citation relationship. The binary attributes attached to each vertex show whether a word in the vocabulary is present or absent. The **Cora** citation network also consists of scientific publications, and the binary attributes of each vertex indicate whether a word in the vocabulary is present or absent [114]. In the **Terrorist** attack network, each vertex represents an attack while the edges connect co-located attacks. Each attack is described by a set of 106 attributes. The **SinaNet** is a microblog user relationship network extracted from sina-microblog website. Each vertex represents a user and each edge represents a relationship. Attributes

are extracted by the LDA topic model that represents users' topic distribution [55]. **Synthetic** is a synthetic attributed network generated by acMark.

We use fifty as the maximum degree to construct the network. Each vertex is described by four synthetic attributes and labeled into one of six categories [80].

FScore and Purity. The evaluation metrics used in this paper are Fscore (or F1 measure) and Purity. F1 is the harmonic mean of precision and recall. Precision measures the ratio of vertices with ground truth labels in a detected community, and recall measures the coverage of vertices with the ground truth label in a detected community. Purity is the fraction of the vertices belonging to the community label. It is averaged in a weighted way over different communities in order to create a composite community purity measure. A good community detection method should obtain both high Purity and high Fscore.

3.5.2 Comparison Methods

We compare the proposed AGGMMR with **Louvain** [18,65], **Distance Dynamics** [115], **NAGC** [79], **CODICIL** [108], **K-means**, **GCN** [61] and **GraphSage** [47]. K-means is the most widely used and efficient attribute-based clustering algorithm. Louvain is one of the most widely used modularity-based methods which is not only well known in research communities but also in industries due to its good scalability and quality performance. AGGMMR also utilizes Louvain for modularity maximization on the augmented graph constructed in the first step. Distance Dynamic is a state-of-the-art algorithm detecting community naturally through vertices' interaction dynamics. NACG is the state-of-the-art non-negative matrix decomposition-based community detection algorithm that takes both attributes and connectivity information into consideration. CODICIL is a three-step framework that transforms attributed graph into a nearest neighbour-based augmented

graph, and then finds communities with coherent attributes and connectivity structure. Compared with traditional matrix decomposition and generative methods, CODICIL is more efficient and scalable. GCN is graph convolution neural network which effectively learns vertex representation through both attributes and topology information. It has achieved state-of-the-art performance for various graph applications. Extended from GCN, GraphSage is also a widely used deep graph neural network. It captures both vertex attributes and topology information on dynamic graph through inductive representation learning.

To summarize, Louvain and Distance Dynamics only utilise connectivity information for partition while K-means only uses attribute information for partition. NACG, CODICIL, GCN and GraphSage partition the graph by utilising both types of information.

Since the K-means algorithm requires K as input, we perform parameter search for it. In specific, we test different values of K which includes but not limited to the number of communities described in ground truth & the number of communities detected by algorithms that do not need K as input. We do the same process for all algorithms that adapts K-means algorithm. For the CODICIL algorithm, we set parameter N as 50 which is the value suggested by authors, and we perform parameter search for α from 0.1 to 0.9 to balance the weights from two types of information. For GCN and GraphSage, we perform parameter search for hyper parameters (*e.g.,* number of hidden dimensions, number of layers) and adapt the optimal parameters for different datasets correspondingly. We use K-means algorithm to extract communities from vertex representations.

3.5.3 Results

From the results reported in Table 3.2 and 3.3, AGGMMR outperforms all seven baselines on almost all datasets, on both measures. The results also show that all methods based on both attributes and connectivity information

have better and consistent performance in general. Due to the existence of inconformity between attributes and connectivity information, a method that focuses only on a single type of information may achieve a higher quality result on some datasets. For instance, on SinaNet network, attributes are user's topic distribution which is directly related to users' labels. K-means algorithm achieves a much better results than other baselines because of the usefulness of the attributes for clustering.

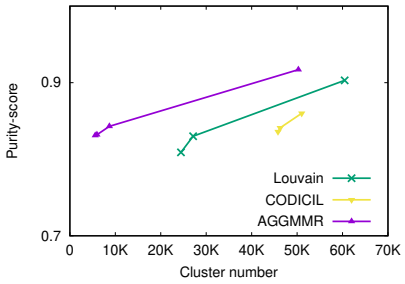
3.6 Seller Community Detection on PayPal APAC Payment Network

After showing good results on open networks, we now present the study on PayPal payment network. We illustrate how our framework can be adapted to handle complex attributes on large complex networks. The goal of our study is to uncover seller communities in PayPal's APAC network using both attributes and connectivity information.

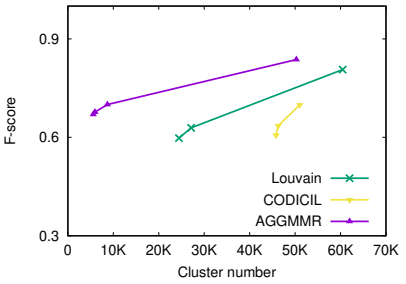
3.6.1 PayPal Networks

We use two PayPal internal datasets with (partial) ground truth labels in this case study. The **Philippine Network** consists of $1.5 million$ users with $2 million$ transactions between those users. All sellers in this network are located in Philippine while the buyers are from all over the world. It is a subset of the APAC network which contains a large number of freelancer businesses. The **APAC Network** consists of $100 million$ users with $1.5 billion$ transactions between those users. This network includes all sellers located in the APAC region while the buyers are from all over the world.

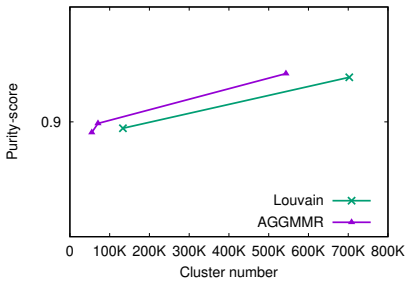
A user (*i.e.,* a vertex in the network) here can be either a seller or a buyer. The edges are weighted by the amount of the transaction between the two users. We leverage a set of 68 attributes associated with the majority



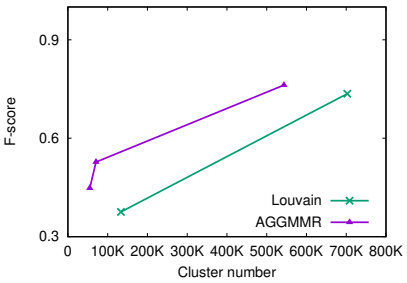
(a) Purity on PayPal's Philippines Network



(b) Fscore on PayPal's Philippines Network



(c) Purity on PayPal's APAC Network



(d) Fscore on PayPal's APAC Network

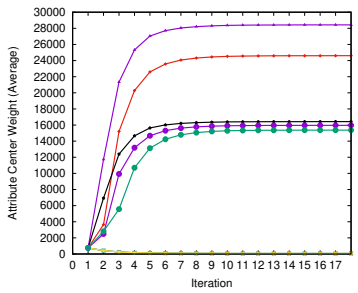
Figure 3.4: Community Quality Comparisons on PayPal's Network

of sellers in our study.[2] Those attributes contain general information about sellers' business and the PayPal products they have used. However, not all attributes are available for all sellers, *i.e.,* there are many missing attributes and missing values as expected.

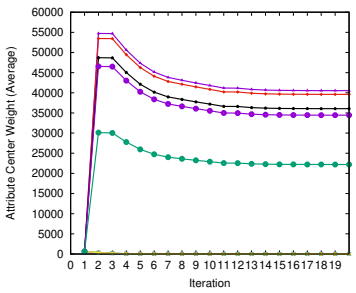
3.6.2 Handling Complex Data Types

Attributes in PayPal network are in various different data types. For example, a seller's business region is a categorical value while its payment volume through a specific PayPal product is numerical. Clustering on attributes with mixed types is challenging. Nevertheless, our framework is flexible enough to adapt different methods for attribute clustering. In this case study, we use

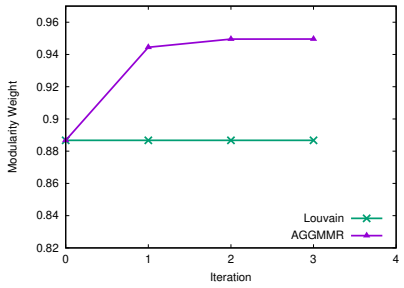
[2]The attributes are produced internally. Due to privacy consideration, we masked the schema of those attributes.



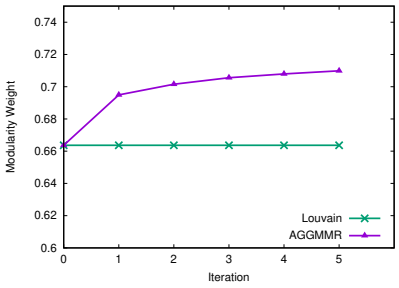
(a) Weight Learning on Philippines Network



(b) Weight Learning on APAC Network



(c) Modularity Refinement on Philippines Network



(d) Modularity Refinement on APAC Network

Figure 3.5: Weight Learning and Modularity Refinement on PayPal's Network

the K-prototype algorithm to cluster attributes and construct the augmented graph. k-prototype extends k-means clustering algorithm and is efficient for clustering large data sets with mixed attribute types [54].

Besides mixed data types, in our data set, there are two additional special data types that are unable to be processed directly by traditional data processing algorithms. One is **many-value categorical attribute** and another is **multi-value categorical attribute**. The former refers to attributes that contain a large cardinality of values. For example, the value of "country code" may contain more than one hundred country codes. One hot encoder on this type of attribute leads to sparse latent dimensions which decreases clustering performance. The latter, multi-value categorical attribute refers to attributes that contain multiton values. One example is the attribute "prod-

uct bundle". Each value of it is a set of singleton values such as *product A, product B*. Those complex data types need to be specially handled before they can be used for clustering. We extend k-prototype algorithms for this purpose.

The k-prototype algorithm clusters data against k prototypes instead of k means. Each prototype is represented by a vector which is a combination of numerical attributes and categorical attributes. In each iteration, it updates numerical attributes by their means while updates categorical attributes by their modes. In k-prototype, the distance between a vertex v_i and a prototype v_p is defined by

$$d(v_i, v_p) = \sum_{j=1}^{m_r} (v_{ij}^r - v_{pj}^r)^2 + \gamma \sum_{j=1}^{m_c} \delta(v_{ij}^c, v_{pj}^c) \tag{3.8}$$

where m_r is the number of numerical attributes, v_{ij}^r and v_{pj}^r are values of a numeric attribute of v_i and v_p respectively. m_c is the number of categorical attributes and v_{ij}^c and v_{pj}^c are values of a categorical attribute. γ is a weight balancing the two types of attributes. $\delta(v_{ij}^c, v_{pj}^c) = 0$ if $v_{ij}^c = v_{pj}^c$, and $\delta(v_{ij}^c, v_{pj}^c) = 1$ otherwise.

Modified distance measure. Simply uses 1 to indicate the difference between two different categorical values (*i.e.,* $\delta(v_{ij}^c, v_{pj}^c)$) fails to reflect the attribute's value distribution. Further, this equation is inadequate to handle many-value and multi-value categorical attributes.

Example 3.6.1. *Suppose a categorical attribute have 3 possible values: A, B, and C. Among 10 vertices, 8 vertices have value A while 1 vertex has value B and another has value C. Intuitively, the difference between value A and value C (or A and C) should larger than the difference between value B and value C if we consider their distributions.*

To retain categorical value distribution, and to handle multi-value and many-value categorical attributes, we normalize attribute values as follows

in this case study.[3]

- Numerical attributes are normalized by z-score normalization.
- Categorical attributes are encoded by one hot encoder and normalized by z-score normalisation (Normalization may lose interpretability but depends on application).
- For multi-value and many-value categorical attributes, we normalise each singleton value by z-score normalisation, and store each value as a ⟨categorical value, z-score⟩ pair. A multi-value attribute is stored as a set of key-value pairs.

The distance between a vertex v_i and a prototype v_p is redefined as:

$$\begin{aligned} d(v_i, v_p) = &\sum_{j=1}^{m_r} \|(\hat{v_{ij}^r} - \hat{v_{pj}^r})\| + \sum_{j=1}^{m_c} \|(\hat{v_{ij}^c} - \hat{v_{pj}^c})\| \delta(v_{ij}^c, v_{pj}^c) \\ &+ \sum_{j=1}^{m_u} \mathbb{J}(\hat{v_i^u}, \hat{v_p^u}) + \sum_{j=1}^{m_a} \|(\hat{v_{ij}^a} - \hat{v_{pj}^a})\| \delta(v_{ij}^a, v_{pj}^a) \end{aligned} \tag{3.9}$$

In above equation, $\hat{\cdot}$ denotes normalized values. v^u is a value of multi-value attribute and v^a represents a value of many-value attribute. With respect to the original distance, we use the difference of normalized values between two categorical values to represent their distance, instead of 1.

For multi-value attributes, each value is a set of key-value pairs. We calculate their distances using weighted Jaccard distance $\mathbb{J}$.

$$\mathbb{J}(\hat{v_i}, \hat{v_p}) = 1 - \frac{\sum_{x \in \hat{v_i} \cap \hat{v_p}} w(x)}{\sum_{y \in \hat{v_i} \cup \hat{v_p}} w(y)} \tag{3.10}$$

Here $w(x)$ is the normalized value of x. Weighted Jaccard distance, values in the range of [0,1], measures the dissimilarity between two multi-value attributes.

[3]Other forms of normalization than z-score normalization can be applied as well. Whether to normalize a categorical attribute depends on the needs from application.

Prototype Updating for Categorical Variable The original k-prototype algorithm updates a categorical attribute of a prototype in two steps: (i) calculate the frequency for all categories, and (ii) assign the prototype the category with the highest frequency.

This updating scheme can be directly extended to many-value and multi-value attributes. For multi-value attribute, the value of a prototype is a set of singleton values. It means that we need to choose multiple singleton values as a set, to update it in an iteration. For example, given 4 attributes, each has its 4, 5, 4, 3 singleton values respectively, listed in a column.

$$\begin{bmatrix} c_{1,1} & c_{2,1} & c_{3,1} & c_{4,1} \\ c_{1,2} & c_{2,2} & c_{3,2} & c_{4,2} \\ c_{1,3} & c_{2,3} & c_{3,3} & c_{4,3} \\ c_{1,4} & c_{2,4} & c_{3,4} & \\ & c_{2,5} & & \end{bmatrix}$$

If k is 3, we assign 3 prototypes with 4 multi-value attributes as:[4]

$p_1 = \{\{c_{1,1}, c_{1,3}\}, \{c_{2,1}, c_{2,2}\}, \{c_{3,2}\}, \{c_{4,1}, c_{4,3}\}\}$,
$p_2 = \{\{c_{1,2}\}, \{c_{2,3}, c_{2,4}\}, \{c_{3,2}\}, \{c_{4,2}\}\}$,
$p_3 = \{\{c_{1,4}\}, \{c_{2,2}\}, \{c_{3,3}, c_{3,4}\}, \{c_{4,3}\}\}$

A singleton value is considered frequent if it is shared by majority vertices in a cluster. Based on this intuition, multi-value attribute can be updated in two steps: (i) calculate frequencies for all singleton values of one multi-value attribute, and (ii) assign to the prototype the set of singleton values where each value is shared by more than half vertices in the cluster.

In other words, when a value is shared by more than half of the vertices in a cluster, it will be updated to the prototype because it is considered a common feature to that cluster. One concern is that whether we can avoid updating two mutually exclusive singleton values to a single prototype at the same time. The answer is positive and is stated below.

[4]The values in this example are randomly assigned for purpose of illustration.

Lemma 1. *Two frequent values updated to prototype will never be mutually exclusive.*

It can be proved by contradiction. Two singleton values are mutually exclusive if they are unable to appear in the same multi-value attribute instance. Let us assume two singleton values v_1 and v_2 are mutually exclusive and both of them have been chosen to update to a prototype in a cluster. So there are more than half vertices containing value v_1 and more than half vertices containing value v_2. To satisfy both conditions, at least one vertex shall contain both v_1 and v_2, which contradicts with our assumption. As a result, our model does not take mutually exclusive values when updating a multi-value attribute for a prototype.

3.6.3 Analysis of Clustering Results

There are many challenges when we run community detection algorithms on real PayPal network compared to running on open data sets. The attributes in PayPal's network provide very important information for clustering. However, not all sellers have all attributes available and at the same time there are aforementioned special attribute types. Those methods that solely rely on attributes and methods require all vertices to have the same set of attributes are not directly applicable on this network.

In our case study, we compare AGGMMR with Louvain and CODICIL. Louvain is known for its scalability and it only uses connectivity information for partition. So it can be directly applied to our network. For CODICIL, instead of using union edges to replace all original edges, we calculate union edge for vertices who have this set of attributes and left other edges with their original weights. With this minor modification, CODICIL is able to run on PayPal's network without any additional assumptions. However, due to the large complexity for calculating nearest neighbours for each vertex in CODICIL, we did not manage to run it successfully on our APAC data set, the data with 100 million vertices.

The Purity and FScore results plotted in 3.4 show that AGGMMR yields much better results than the two baselines. As the Louvain algorithm neglects attributes information, it only detects community with dense connections. The CODICIL algorithm does not perform as well as expected even it considers both types of information. There might be two possible reasons. The first is that the CODICIL algorithm prunes information on the graph to reduce computation cost. As the result, a lot of information is pruned if we force it to run on a very large graph. The second possible reason is that it does not well handle the noise from the unconformity between attributes and connectivity information in our network. Simply combining both types of information may not always give better performance. AGGMMR tends to cluster communities that conform to both types of information; it is robust to handle these scenarios. Our internal evaluation of communities detected by AGGMMR shows a strong goodness of fit with existing professional knowledge.

3.6.4 Analysis of Weight Learning

Figures 3.5(a) and 3.5(b) report weight changing in the weight learning phase on PayPal network. Since belongingness edge weights are changed based on each attribute center, we selected 10 attributes center's average weights (δw_i^t in Equation 3.4) to analyze their updating by iterations. Half of them are attribute centers that have the largest positive weight changes and others are the centers that have the largest negative weight changes. Those centers with the largest positive weight changes are those centers whose members are distributed in a very small number of communities by modularity maximisation. This is consistent with our expectations. If an attribute center contains a large number of members, and most of them concentrate on a small number of communities, it indicates the attributes relationship for this attribute center provides a very positive contribution in clustering. The attribute centers with the largest negative weight changes are those centers

whose members are distributed into different communities. The connections from those attribute centers provide trivial information or even noise. These attribute centers would be adjusted to much smaller weights automatically in our weight learning phase.

3.6.5 Analysis of Modularity Refine

Figures 3.5(c) and 3.5(d) show the trend of modularity refinement when running AGGMMR on PayPal network. The modularity score from the Louvain method is also plotted as a baseline for comparison. We observe that the modularity score is continuously increasing through the whole process of our refinement, and there always exists a significant modularity increase at the first iteration of refinement. It illustrates the fact that in most cases, a small number of false assignments caused by poor processing ordering would result in a hugely different result. Such cluster assignments can be effectively refined through our greedy refinement process in a small number of iterations.

3.7 Conclusion

In this chapter, we propose a practical framework for community detection based on attributes and connectivity information for large network with mixed types of attributes. The attributes information can be effectively captured and transformed into an augmented graph through attribute clustering. After that, a weight learning process is designed to automatically adjust weights from those attributes information according to their contributions to clustering. A greedy-based modularity maximisation algorithm is adopted to partition the graph based on both types of information and finally a modularity refinement process is introduced to optimise the result effectively. Experiments on different open data networks and a study on PayPal's large payment networks demonstrate that our proposed framework is effective and practical.

Chapter 4

Seller Community Detection on Payment Network: An Incremental Approach [1]

4.1 Introduction

In the last chapter, we propose the AGGMMR framework to detect communities on payment network. Network data in industry is seldom static but grows with new vertices or edges being added over time. Thus, it is necessary to detect communities for previous unseen vertices. However, it is not viable to perform community detection from scratch every time when new vertices/edges get added, especially when a network is large. As graph keeps evolving, it is also difficult to map new vertices to the previously detected communities. All aforementioned challenges limit many existing algorithms to be applicable to industry applications.

In this chapter, we extend the AGGMMR framework to perform incremental community detection on large, attributed, and growing networks. By studying the weight learning algorithm in AGGMMR, we observe that the weights of belongingness edges for all vertices that belong to the same attribute center can be easily approximated, without the need of re-running

[1] The work in this chapter has been published in ACM Transactions on Knowledge Discovery from Data (TKDD), 15(6), 1-20.

weight learning. Based on this observation, we propose an incremental community detection framework, named inc-AGGMMR, to detect communities in complex attributed network. Our proposed incremental framework addresses all challenges for community detection on a complex attributed and growing network.

We summaries the main contributions of the work as belows:

- We proposed a novel incremental community detection framework to detects communities in complex attributed network. Compare with existing algorithms, it has the strength in working with more complex problem settings and real applications.
- Complexity analysis is provided to quantitatively estimated the time complexity cost of the incremental framework. The analysis demonstrates the effectiveness of the framework that running on large scale network.
- We perform evaluation of the incremental framework on open datasets, synthetic dataset and the large scale PayPal network by comparing to both groundtruth and non incremental algorithm. It demonstrate our framework inc-AGGMMR is able to incrementally detect communities for unseen vertices and achieves the same detection quality with AGGMMR on different applications.

4.2 The Incremental Algorithm

The main bottleneck for our incremental approach comes from the attribute center assignment and the iterative weight learning. Therefore, we have to find an alternative way to assign a vertex to its attribute center and approximate the weight on its belongingness edge at a lower cost. We observe that the weight on belongingness edge after weight learning is only related to its initial weight and the contribution score of the attribute center that vertex

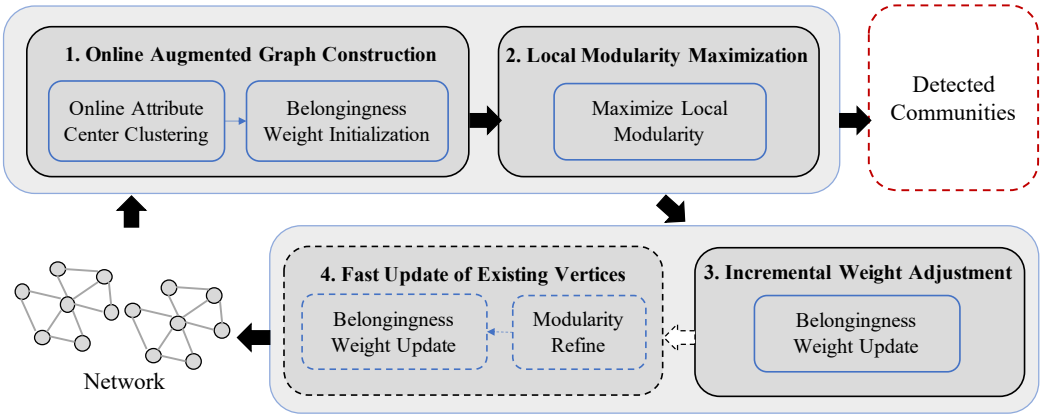


Figure 4.1: The overall Process of inc-AGGMMR. The input to the algorithm is the network with new vertices and edges. The communities are detected by step 1 and 2. The network is updated by step 3 and 4.

belongs to. This property reveals the possibility of designing an incremental approach of AGGMMR.

In this section, we present our incremental algorithm inc-AGGMMR. The framework is shown in Figure 4.1. When a new vertex comes, in the first step (named **Online Augmented Graph Construction**), it will be assigned to an attribute center to retain its attribute relationship through online augmented graph construction. In the second step (named **Local Modularity Maximization**), after all attribute relationships are transformed into edges, the vertex will be assigned to a community with the local modularity maximization model. In the last step (named **Incremental Weight Adjustment**), after all new vertices get their community assignments, the weights on belongingness edges are updated through an incremental weight adjustment algorithm. Additionally, we provide an optional step (named **Fast Update of Existing Vertices**) to update the community assignments for all existing vertices in an efficient manner. Next, we detail each step.

4.2.1 Online Augmented Graph Construction

Clustering vertex to an attribute center can be easily extended by an incremental approach with threshold. This idea has been widely adopted by many

incremental algorithms [46] [22]. Follow this framework, we can assign a new coming vertex to an exist or new attribute center with 2 steps:

- Find the nearest attribute center for new vertex with shortest attribute distance.
- Assign vertex to the nearest attribute center if attribute distance smaller than threshold T. Otherwise, creates a new attribute center for that vertex.

The threshold T can be set with the largest distance between a vertex to its attribute center.

After attribute-based clustering, the vertices have been assigned to their nearest attribute centers. Thus we create belongingness edges between those vertices to their attribute centers. At this stage, the attribute relationships for new vertices have been fully retained on the graph by belongingness edges. Next, we initialize the weight of a belongingness edge as (i) it connects to a new attribute center, we initialize the weight with equation 3.2 described in previous chapter. Since we don't have any information about the new attribute center in the current stage, we initialize the weight of belongingness edges to balance its connectivity information. Later the weight will be adjusted incrementally through our online weight adjustment algorithm. (ii) it connects to an exist attribute center, we initialize the weight same with other belongingness edges connecting to that attribute center. The detail of this scheme will be discussed in Section 4.2.3.

4.2.2 Local Modularity Maximization

In the augmented graph, both attributes and connectivity relationships are represented by edges. We therefore use Louvain modularity maximization model to detect community for new vertices. In specific, it assigns vertex v_i

to one of its neighbours community by maximize the local modularity gain through equation:

$$\Delta Q = [\frac{\sum_{in} + 2k_{i,in}}{2m} - (\frac{\sum_{tot} + k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2] \tag{4.1}$$

Where $\sum_{in}$ is the total weight within the community that v_i is going to join. $\sum_{tot}$ is the total weight to all vertices within that community. k_i is the weighted degree of the v_i and $k_{i,in}$ is the sum of weights v_i connect to other vertices within that community. Now we get the community memberships for all new vertices.

4.2.3 Incremental Weight Adjustment

After detecting communities for new vertices, weight are adjusted for attribute centers according to their contributions. In AGGMMR, the weight of w_i for a belongingness edge is learned from iterative algorithm showed in Chapter 3. We summarize the process of learning after n iterations as follow

$$\begin{aligned}
w_i^{t+1} &= \frac{1}{2}(w_i^t + \frac{\Theta_a}{\sum \Theta}\frac{W}{|V_a|}) \\
&= \frac{1}{2}(\frac{1}{2}(\dots(\frac{1}{2}(w_i^1 + \frac{\Theta_{a1}}{\sum \Theta}\frac{W}{|V_a|}) + \frac{\Theta_{a2}}{\sum \Theta}\frac{W}{|V_a|})\dots) + \frac{\Theta_{an}}{\sum \Theta}\frac{W}{|V_a|}) \\
&= \frac{1}{2^n}w_i + \frac{\Theta_{a1}}{2^{n-1}\sum \Theta}\frac{W}{|V_a|} + \frac{\Theta_{a2}}{2^{n-2}\sum \Theta}\frac{W}{|V_a|} + \dots + \frac{\Theta_{an}}{2^1\sum \Theta}\frac{W}{|V_a|} \\
&= \frac{1}{2^n}w_i + \Delta \\
\Delta &= \frac{\Theta_{a1}}{2^{n-1}\sum \Theta}\frac{W}{|V_a|} + \frac{\Theta_{a2}}{2^{n-2}\sum \Theta}\frac{W}{|V_a|} + \dots + \frac{\Theta_{an}}{2^1\sum \Theta}\frac{W}{|V_a|} \\
&= (\frac{\Theta_{a1}}{2^{n-1}\sum \Theta} + \frac{\Theta_{a2}}{2^{n-2}\sum \Theta} + \dots + \frac{\Theta_{an}}{2^1\sum \Theta})\frac{W}{|V_a|}
\end{aligned} \tag{4.2}$$

It is not hard to verify that the weight is only affected by contribution score Θ and initial weight w_i. The change of weight Δ is shared by all members within the same attribute center. When n gets larger, the impact of initial weight w_i is minimised. The final weight is proportional to the

contribution score of each attribute center. Thus, we can approximate the weight update on belongingness edge through

$$w_i^{t+1} = \frac{\Theta_a}{\Sigma\Theta} \times \frac{W}{|V_a|} \qquad W = \sum w^t \tag{4.3}$$

Using this approximation has two advantages. Firstly, the contribution of each attribute center is well captured. The goal of weight learning is to adjust the weight between different attribute centers. The weight is adjusted according to the contribution of an attribute center towards the final objective. This new weight adjustment scheme fit the objective of weight learning very well. Secondly, it enables incremental update. The weight for belongingness edge can be adjusted by recomputing the contribution score of each attribute center after new vertices are added.

4.2.4 Fast Update of Existing Vertices

Usually most of incremental algorithm did not provide means to update community memberships for all exist vertices. They only update vertices community membership that involved in the new edges. When the network keeps growing, the structure of the network may changes significantly after a certain period. In real application, it is necessary to update the community membership for all exist vertices after a significant change of network structure. However, updating community membership of all exist vertices is inevitable to re-evaluate the communities for all vertices. Thus most of existing incremental algorithms did not provide this functionality due to high computational complexity, especially for algorithms that consider both attributes and connectivity information. Thanks to the efficiency of greedy modularity maximisation framework, it enables a fast update of community membership for all vertices on the network. In inc-AGGMMR, we perform a fast update elegantly with following two steps.

- Performs a modularity refine that described in Chapter 3 to adjust all vertices' membership.

- Update weight of belongingness edges by evaluating the contribution score of each attribute center with equation 4.3.

Modularity refine adjust all vertices' community membership based on both attribute and connectivity information within $O(n \log n)$ time complexity. After that, weight of all attribute center's belongingness edges get updated immediately. Note that, all belongingness edges to the same attribute center share the same weight. Thus it takes at most $O(k)$ to compute the new weights and $O(n)$ to update those weights, where k is the number of attribute center and n is number of vertices have belongingness edge. Now we have an updated augmented graph reflects the latest network structure.

4.2.5 Complexity Analysis for Incremental Algorithm

The whole framework is efficient and scalable. For online augmented graph construction, it takes $O(kn_{new})$ to assign a vertex to the nearest attribute center and $O(n_{new})$ to initialize the weight on new belongingness edges, where k is the number of attribute center and n_{new} is number of new vertices. For local modularity maximization, it detects vertices' communities with complexity $O(n_{new} \log n_{new})$. For the phase of weight adjustment, the time complexity is bounded by $O(n)$ as contribution score of an attribute center only depends on its member vertices' community membership. The total number of vertices connect to an attribute center is less or equal to total number of vertices n in network. Overall, the total time complexity for the framework is bounded by $O(n + n_{new} \log n_{new})$.

4.3 Experiment

In this section, we validate our incremental framework on both open network dataset and large scale PayPal networks.

Table 4.1: Statistics of the four open networks

Dataset	Vertices	Edges	Attributes
Citeseer	3,312	4,732	3,702 binary attributes
Cora	2,708	5,429	1,433 binary attributes
Terrorist	1,293	3,172	106 attributes
SinaNet	3,490	30,282	LDA topics
Synthetic	512	3,242	synthetic attributes

4.3.1 Datasets and Evaluation Metrics

The evaluations are performed on the same four open network data sets with Chapter 3: **CiteSeer**, **Cora**, **Terrorist** and **SinaNet**. Please refer to Chapter 3 for detail descriptions of the datasets.

Synthetic is a synthetic attributed network generated by acMark. We use fifty as the maximum degree to construct the network. Each vertex is described by four synthetic attributes and labeled into one of six categories [80].

To evaluate the capability of incremental community detection, we run inc-AGGMMR in an incremental manner. In specific, we split each data set into two parts. The first part represents the exist data while the second part represents new coming data. The ratio of splitting varies from 0.2 to 0.8(*e.g.,* 0.2 represents 20% exist data and 80% new data). We run AGGMMR to detect communities for exist data, then run inc-AGGMMR to incrementally detect communities for all new data.

FScore and Purity. The evaluation metrics used in this paper are Fscore (or F1 measure) and Purity, which is the same with Chapter 3. Again, please refer to Chapter 3 for detail descriptions of Fscore and Purity.

4.3.2 Comparison Methods

Similar as reported in the Chapter 3, the methods we compared with inc-AGGMMR are: **Louvain** [18,65], **Distance Dynamics** [115], **NAGC** [79], **CODICIL** [108], **K-means**, **GCN** [61], **GraphSage** [47] and **AGGMMR**.

Please refer to last chapter for description of those algorithms that compared in previous chapter. AGGMMR is the community detection algorithm we proposed in last chapter that detect communities based on both attributes and connectivity information.

To summarize, Louvain and Distance Dynamics only utilize topological information for detection while K-means only uses attribute information for detection. NAGC, CODICIL, GCN, GraphSage, and AGGMMR detect communities by utilizing both types of information.

Same with previous chapter, since K-means algorithm requires K as input, we perform parameter search for it. In specific, we test different values of K which includes but not limited to the number of communities described in ground truth & the number of communities detected by algorithms that do not need K as input. We do the same process for all algorithms that adapts K-means algorithm. For CODICIL algorithm, we set parameter N as 50 which is the value suggested by authors, and we perform parameter search for α from 0.1 to 0.9 to balance the weights from two types of information. For GCN and GraphSage, we perform parameter search for hyper parameters (*e.g.,* number of hidden dimensions, number of layers) and adapt the optimal parameters for different datasets correspondingly. We use K-means algorithm to extract communities from vertex representations.

4.3.3 Results

The baseline methods that are non-incremental detect communities on the entire dataset. For GraphSage and inc-AGGMMR, to simulate the growing of network, we split each dataset into two parts. The first part represents the existing data while the second part represents new coming data. The ratio of the splitting varies from 0.2 to 0.8 with the step of 0.2 (*e.g.,* 0.2 means that 20% of the dataset is considered as existing data and the remaining 80% is new data). For GraphSage, we train it on the existing data then perform representation inference on new data. For inc-AGGMMR, we run AGGMMR

Table 4.2: Evaluations on the four open network datasets and one synthetic dataset. Best results are in bold and second best underlined, on each dataset.

	Purity				
Dataset	Citeseer	Cora	Terrorist	SinaNet	Synthetic
Louvain	0.7355	0.7651	0.7318	0.2506	<u>0.7225</u>
DDynamic	0.5041	0.5194	0.7318	0.3265	0.5395
K-means	0.5018	0.5255	<u>0.8118</u>	<u>0.7684</u>	0.5949
NAGC	0.7271	0.7858	0.7069	0.5484	0.4234
CODICIL	0.6760	0.7991	0.7349	0.6550	0.6243
GCN	0.6111	0.7323	0.7380	0.4947	0.6250
GraphSage	0.3898	0.4446	0.7001	0.3068	0.6358
AGGMMR	<u>0.7656</u>	<u>0.8342</u>	0.7504	**0.8131**	0.7207
inc-AGGMMR	**0.7888**	**0.8731**	**0.8160**	0.7664	**0.7289**

	Fscore				
Dataset	Citeseer	Cora	Terrorist	SinaNet	Synthetic
Louvain	0.7250	0.7567	0.5498	0.1924	<u>0.7188</u>
DDynamic	0.5896	0.4878	0.5534	0.3380	0.4590
K-means	0.4953	0.5060	0.5706	0.7462	0.5949
NAGC	0.7001	0.7736	0.5265	0.4610	0.4199
CODICIL	0.6595	0.7800	0.5660	0.6051	0.6250
GCN	0.5848	0.7251	0.5005	0.3965	0.6206
GraphSage	0.3883	0.4446	0.5836	0.2666	0.6314
AGGMMR	<u>0.7501</u>	<u>0.8238</u>	<u>0.5892</u>	**0.7653**	**0.7207**
inc-AGGMMR	**0.7860**	**0.8432**	**0.6122**	<u>0.7573</u>	0.7101

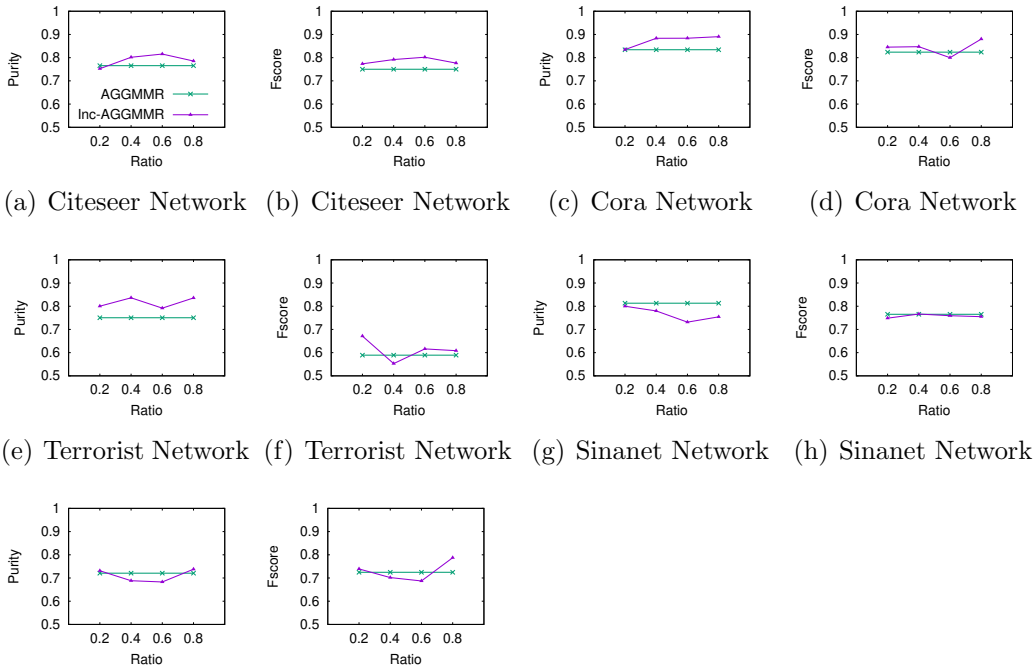
on existing data, then run inc-AGGMMR to incrementally detect communities with new data. For easy comparison with non-incremental methods, we calculate the average performance of GraphSage and inc-AGGMMR over different ratios of split.

Table 4.2 reports the average performance of inc-AGGMMR and all baseline methods. Both inc-AGGMMR and AGGMMR outperform all other methods on almost all datasets. The inc-AGGMMR provides very stable performance, and on three out of four real-world datasets, it outperforms

its non-incremental counterpart, AGGMMR. One possible reason for inc-AGGMMR achieving better results than AGGMMR could be multiple rounds of updating during the processing of new vertices. As we discussed earlier, the last step in AGGMMR is modularity refinement, for the purpose of reducing the impact of the vertex processing order to the final results. In the process of inc-AGGMMR, the vertices are updated multiple times with the processing of new coming vertices (see Section 4.2.4). This updating process may further refine the modularity for better final community detection.

Due to the unconformity between attribute and topological information on the datasets, those methods that only take one type of information show very biased performances. For example, K-means algorithm performs quite well on Terrorist and Sinanet datasets but performs poorly on the Citeseer and Cora datasets. GCN and GraphSage take both types of information for community detection, but their performance is unstable. They do not handle the unconformity of information and the parameters of GraphSage are not updated along with the grow of the network. AGGMMR and inc-AGGMMR automatically adjust the weight to reduce the noise from unconformity of information. Thus both show very stable performance on all the datasets. However, an incremental approach is usually more sensitive to noise. On the Sinanet dataset, inc-AGGMMR achieves good performance on Fscore, but its purity is slightly lower than K-means.

Figures 4.2 plots the performance comparison between inc-AGGMMR's and AGGMMR, by Purity and Fscore respectively. Here, the performance of AGGMMR are obtained on the entire dataset, and that of inc-AGGMMR are obtained by using a ratio of dataset as existing data and the remaining as new data, as stated earlier. There is no clear pattern when different ratios of data are used as new data. On the other hand, on all four datasets, the performance of inc-AGGMMR is very similar to, and most time is slightly better than AGGMMR. The results are consistent with what we observed in Table 4.2.



(a) Citeseer Network (b) Citeseer Network (c) Cora Network (d) Cora Network

(e) Terrorist Network (f) Terrorist Network (g) Sinanet Network (h) Sinanet Network

(i) Synthetic Network (j) Synthetic Network

Figure 4.2: Purity & Fscore comparison between AGGMMR and inc-AGGMMR on open datasets

To balance efficiency and performance, most attribute-based incremental algorithms focus only on predicting community memberships of new vertices instead of re-evaluating community memberships of all existing vertices. This trade-off affects the purity of communities that detected by an incremental approach, especially when the ratio of new vertices gets very large. Overall speaking, the inc-AGGMMR algorithm shows a very outstanding performance and its result even better than most of the non-incremental algorithms on all datasets.

4.3.4 Stability Analysis

Besides evaluation by ground-truth labels, we also report the similarity between the results of inc-AGGMMR, by considering the results of AGGMMR

Table 4.3: Result similarity comparison on open network datasets

	Similarity by Purity				Similarity by Fscore			
Dataset	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
`Citeseer`	0.7703	0.9918	0.9745	0.9137	0.9892	0.9807	0.9766	0.8859
`Cora`	0.8983	0.9584	0.9735	0.9718	0.9575	0.9127	0.9395	0.9576
`Terrorist`	0.7364	0.9089	0.9019	0.8598	0.6605	0.8737	0.8827	0.8399
`SinaNet`	0.7805	0.7262	0.7243	0.7395	0.7614	0.7283	0.7731	0.7370
`Synthetic`	0.7017	0.6786	0.7024	0.7767	0.7595	0.7437	0.7343	0.8457

as the ground-truth labels. That is, we run AGGMMR on the entire dataset and consider the community membership detected by AGGMMR as ground-truth labels. Then we run inc-AGGMMR by using different ratios of the dataset as existing data and the remaining as new data, and evaluate the results of inc-AGGMMR, based on the labels obtained by AGGMMR.

Because inc-AGGMMR is built on top of AGGMMR with the same objective function, it is expected that both algorithms deliver similar results. We use Fscore and Purity to quantify the quality of inc-AGGMMR against the labels generated by AGGMMR, by using different ratios of data as existing data, and the remaining as new data. Table 4.3 reports the results. On three out of the four datasets, inc-AGGMMR gives very similar results as AGGMMR. When 40% of data are used as existing data, inc-AGGMMR achieves high Purity and Fscore on CiteSeer, Cora, and Terrorist datasets. For SinaNet and Synthetic dataset, the vertex degree is significanlty more than that in other datasets (see Table 4.1). The large degree number could be one possible reason of less similar results between inc-AGGMMR and AGGMMR. Overall, we consider that inc-AGGMMR is stable for incremental community detection.

4.4 Incremental Seller Community Detection on PayPal APAC Network

The goal of our study is to uncover seller communities in PayPal's APAC dynamic network using both attributes and connectivity information.

4.4.1 PayPal Networks

We use the same PayPal datasets with Chapter 3 for the case study. The **Philippine Network** consists of $1.5 million$ users with $2 million$ transactions between those users. All sellers in this network are located in Philippine while the consumers are from all over the world. It is a subset of the APAC network which contains a large number of freelancer businesses. The **APAC Network** consists of $100 million$ users with $1.5 billion$ transactions between those users. This network includes all sellers located in the APAC region while the consumers are from all over the world.

The detail of the preprocessing that handling with missing attributes and values are detailed described in Chapter 3. In specific, we adapt a modified K-prototype algorithm with a modified distance measurement to handle complex data and perform attribute-based clustering on those vertices that contains attributes.

Again, we split both network datasets into two parts. The first part represents existing data while the second part represents new data. AGGMMR firstly perform community detection on all existing data, then inc-AGGMMR incrementally predicts community memberships for new data.

4.4.2 Analysis of Results

Performing community detection on PayPal's network is challenging due to large data size and complex, incomplete attributes. Most of existing algorithms are not directly applicable on such a complex network. Follows the

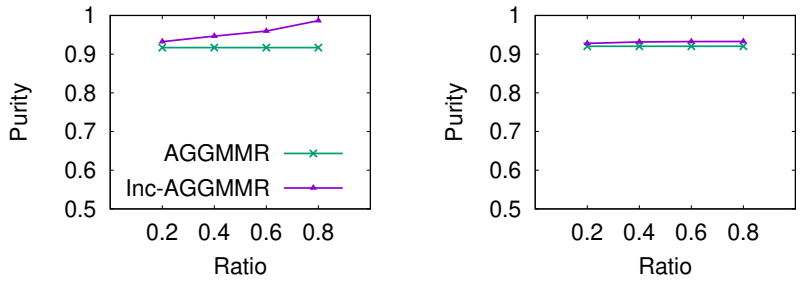
Table 4.4: Evaluation on PayPal Network

	Purity			
Dataset	Louvain	CODICIL	AGGMMR	inc-AGGMMR
Philippines	0.8665	0.8505	0.9170	0.9563
APAC	0.8944	-	0.9203	0.9311

Table 4.5: Evaluation on PayPal Network

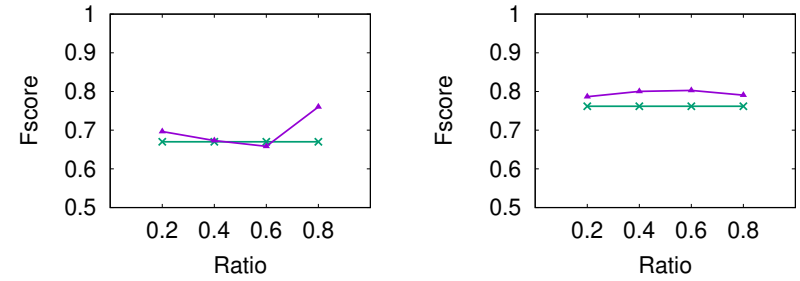
	Fscore			
Dataset	Louvain	CODICIL	AGGMMR	inc-AGGMMR
Philippines	0.5745	0.5340	0.6700	0.6970
APAC	0.5556	-	0.7618	0.7950

experiment settings from Chapter 3, we compare inc-AGGMMR with AGGMMR, Louvain and CODICIL. Louvain is known for its scalability and it only uses connectivity information for partition. So it can be directly applied to our network. For CODICIL, instead of using union edges to replace all original edges, we calculate union edge for vertices who have this set of attributes and left other edges with their original weights. With this minor modification, CODICIL is able to run on PayPal's network without any additional assumptions. However, due to the large complexity for calculating nearest neighbours for each vertex in CODICIL, we did not manage to run it successfully on our APAC data set, the data with 100 million vertices. The results are shown in Tables 4.4, 4.5 and Figure 4.3. From the results, both AGGMMR and inc-AGGMMR outperform other methods with a very large margin. Since the communities in real industry network usually not based on any individual type of information, an algorithm considers more information usually gives better result. However, simply equally combine different types of information may also not helpful. For example, some freelancer sellers form a community due to their connections with some similar freelancer platforms. In this case, attributes information may not help to find those communities but only make noise. Fortunately, the weight learning phase



(a) Purity on Philippines Network

(b) Purity on APAC Network



(c) Fscore on Philippines Network

(d) Fscore on APAC Network

Figure 4.3: Purity and Fscore Comparisons with AGGMMR on PayPal network

in inc-AGGMMR helps to reduce those noise when performing community detection based on both types of information. As a result, inc-AGGMMRR detects communities based on stronger side of relationships, either attributes, connectivity or both. Thus it achieves the best performance on both data sets. The result for stability analysis is shown in Figures 4.6 and 4.7. From the result, we observed that even on such large networks, the inc-AGGMMR is still produce high similarity result which is consistent with AGGMMR. Addition to our evaluation, the major communities detected from AGGMMR and inc-AGGMMR are presented to internal PayPal analysts and the results show strong consistency with their existing knowledge.

Table 4.6: Similarity Comparisons by Purity on PayPal network

	Similarity by Purity			
Dataset	0.2	0.4	0.6	0.8
Phillipine	0.9976	0.9970	0.9965	0.9954
APAC	1.0000	0.9100	0.9305	0.9475

Table 4.7: Similarity Comparisons by Fscore on PayPal network

	Similarity by Fscore			
Dataset	0.2	0.4	0.6	0.8
Phillipine	0.8151	0.7674	0.7707	0.7274
APAC	0.9259	0.9256	0.7547	0.7998

4.5 Conclusion

In this chapter, we proposed a novel incremental community detection algorithm inc-AGGMMR to overcome the challenges from complex attribute network. It decouples the computation of attributes and connectivity relationships, thus can be easily adapted to many challenging scenarios. The weight learning phase helps to reduce the noise from the inconformity between attributes and connectivity information, thus provides better quality communities. Further more, the low time complexity enables our algorithm to run on very large scale network. The experiment on both open datasets and PayPal network demonstrates the effectiveness and practicality of our algorithm.

Chapter 5

Risky Seller Detection on Payment Network

5.1 Introduction

The task of risky seller detection is to detect risky sellers, *i.e.,* the sellers who result in revenue loss due to various reasons (*e.g.,* fraud, bankruptcy, bad suppliers) on payment network. Identifying risky seller in PayPal payment services helps to cover millions of revenue loss every year. In recent years, graph-based methods especially graph convolution networks, have shown great success in payment network risk detection [77, 87].

One limitation of many existing graph convolution networks is that they only model the connectivity relationship between vertices, and are unable to model other types of vertex properties (*e.g.,* vertices' local topology structures). This is because the convolution process in graph convolution networks only propagates information across the edges of the graph. However, many real-world applications require to take vertices' local topology structure into consideration. A large payment network typically contains millions of sellers and transactions. In order to detect such risky sellers in advance, we need to consider sellers' connectivity relationships (*e.g.,* interactions with other sellers, buyers, and suppliers), as well as their topology structural similarities (*e.g.,* similarities on business models as supplier, drop-shipper, or retailer).

We argue that existing graph network architectures are very limited in modeling both properties.

Motivated by this need, in this chapter, we propose a **D**ual-**P**ath **G**raph **C**onvolution Attention **N**etwork architecture, named **DP-GCN**, for vertex classification. DP-GCN combines both connectivity and topology structure properties, in a very efficient and effective manner. Specifically, DP-GCN consists of a C-GCN module, a T-GCN module, and a multi-head self-attention module. The C-GCN module captures the connectivity relationships between vertices; the T-GCN module captures the topology structure similarities between vertices; and the multi-head self-attention module aligns both properties. To model topology structure more efficiently, we propose *role-based convolution*, which extracts vertex topology structure into high-level *latent roles* for convolution. This strategy significantly reduces the computation required for each vertex but well retains all convolution paths between vertices that are similar in topology structure.

In our experiments, we evaluate DP-GCN on four open network datasets. Compared to state-of-the-art baselines, we observe DP-GCN achieves significant performance improvements on all datasets. We also conduct a careful ablation study and an embedding analysis to show that all components of DP-GCN are important for the best performance. These studies also show that our model's inner working is interpretable. Further, we evaluate DP-GCN on three large scale PayPal payment networks to detect real risky sellers. This study is challenging due to the large-size of the networks, the diversity of risky seller types, as well as the high imbalance ratio. As expected, only a very small number of sellers are risky sellers.

In summary, we have made the following contributions in this work:

- We propose DP-GCN, a dual-path graph convolution attention network, to effectively perform vertex classification, based on both vertex connectivity relationship and topology structure similarity.

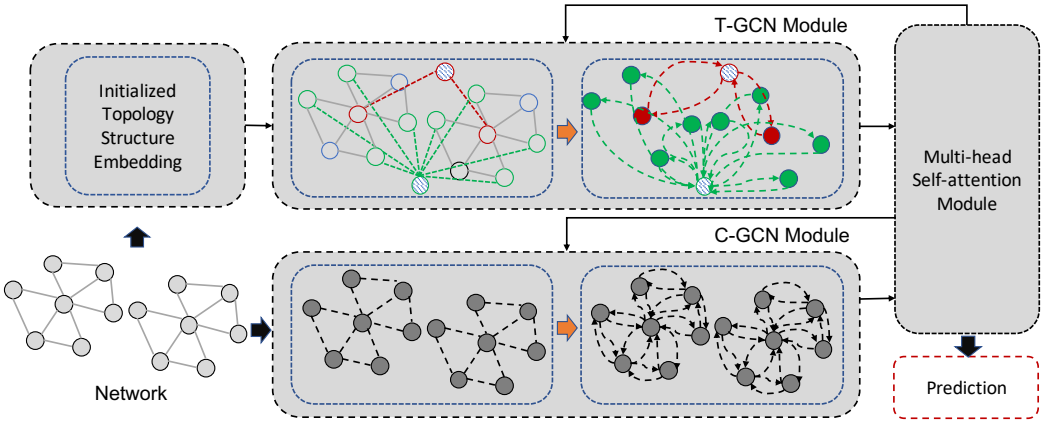


Figure 5.1: Dual-path convolution with T-GCN and G-GCN (Best viewed in color). The shaded vertices in T-GCN are latent topology roles, constructed on top of existing vertices. The unified embedding produced by multi-head attention are shared as input for the next layer C-GCN and T-GCN. Thus both convolutions are interactively optimized towards the common classification objective.

- We propose role-based convolution, a novel and generalizable model to enable the convolution process to be applied on additional vertex properties aside connectivity.
- We evaluate our DP-GCN on six benchmark datasets, and provide a case study on three large-scale PayPal payment network datasets. Together with ablation study and embedding analysis, we show DP-GCN's effectiveness and practicality.

5.2 DP-GCN Architecture Overview

In our problem setting, we take both the vertex features F and the graph G as input, and predict the label associated with each vertex. Vertex features hence are not related to the graph connectivity. In many related graph convolution network researches, they also support a non-feature option. In this case, an Identity matrix or Random matrix is used as feature matrix F [96].

The overall architecture of the DP-GCN is illustrated in Figure 5.1. We build our DP-GCN on top of the graph convolution framework and its classification process is summarized as follows. vertex representations are first initialized with input features. We model vertex connectivity by the C-GCN module and model topology structure similarity by the T-GCN module, through a dual-path convolution process. The outputs from both the C-GCN and T-GCN modules are aligned by the multi-head self-attention module. This multi-head self-attention module automatically adjusts the importance of outputs from C-GCN and T-GCN, and learns the unified embedding representations of vertices. The learned unified embeddings are then fed to the next layer of dual-path convolution, and eventually fed into the classification module for the final prediction of class labels. In the following sections, we detail all the modules in DP-GCN.

5.3 Dual-Path Convolution

The dual-path convolution layer aims to learn the connectivity as well as the topology structure embeddings of vertices simultaneously, with graph convolution in an end-to-end fashion.

As discussed in chapter 2, related works [51, 149, 154] already show that dual-path convolution architecture is superior in learning unified vertex embedding. Our dual-path convolution consists of two interactive convolution modules: the C-GCN module and the T-GCN module. In our dual-path architecture, both modules take the unified embedding as input, thus the two modules will mutually reinforce each other, towards the common classification objective.

5.3.1 The C-GCN Module

The C-GCN module employs the standard graph convolution, for modeling vertex connectivity relationships, which has been demonstrated to be

effective in vertex representation learning. The input to the connectivity convolution layer is the vertex feature matrix F, which represents the initial vertices' embeddings. All vertices connectivity information are stored in the adjacency matrix A_c, thus we directly apply convolution based on A_c. Specifically, we use the symmetric normalizing trick described in [61] to normalize A_c, to avoid changing the scale of feature vector:

$$\hat{A}_c = \hat{D}^{-\frac{1}{2}}(A_c + I)\hat{D}^{-\frac{1}{2}} \tag{5.1}$$

where $\hat{D}$ is the diagonal degree matrix and I is the Identity matrix. The Identity matrix is added to include each vertex's self-feature into convolution. Each convolution layer takes the input from previous layers and generates vertex level embedding $F^{(l+1)}$. The computation of the convolution is defined as:

$$F_c^{l+1} = \sigma(\hat{A}_c H^l W) \tag{5.2}$$

In this equation, W is the weight matrix, l denotes the $l-th$ convolution layer. H denotes the unified embedding received from the multi-head self-attention module (see Figure 5.1), and the only exception is that $H^1 = F$. The output of the C-GCN module F_c^{l+1}, will be fed into the multi-head self-attention module together with the output from the T-GCN module to produce the updated unified embedding H^{l+1}.

Multi-layer convolutions enable the vertex to receive messages from a further neighbourhood [134]. As detailed above, our design is different from a typical convolution layer. The input to the successive convolution layers (*i.e.,* when $l > 1$) in the C-GCN module is the unified vertex embedding H^l, rather than F^l as in typical convolution. This design allows the two modules C-GCN and T-GCN to mutually reinforce each other while modeling different vertex properties.

5.3.2 The T-GCN Module

The goal of the T-GCN module is to model the topology structure similarity of vertices through graph convolution. The topology structure property is fine-tuned during the convolution, and finally integrated into the vertex embeddings.

Threshold-based Convolution Model. The simplest way to model topology structure by graph convolution, is to generate "augmented paths" between vertices that share similar topology structures. Those augmented paths create a new kind of neighborhoods for vertices that are similar in terms of local topology structures. In this way, graph convolution can aggregate vertices along the augmented paths, which naturally learn the vertex embedding based on topology structure similarity.

Based on this strategy, a simple threshold-based convolution model can be designed with the following steps: (i) leverage local topology structure embedding algorithms to generate vertex topology structure features, (ii) calculate pair-wise similarity between vertices based on the topology structure features, (iii) create augmented paths between vertices if their topology structure similarity is larger than a predefined threshold, and (iv) perform graph convolution on augmented paths to fine-tune the vertex topology structure embedding for classification.

With this model, we can derive a topology structure adjacency matrix A_t to capture the augmented paths between vertices. Then the convolution in Equation 5.1 can be directly adopted on the augmented paths, *i.e.,* by replacing A_c with A_t.

This threshold-based model is straightforward and easy to implement. However, it requires $O(n^2)$ time complexity to evaluate the topology structure similarity between all pairs of n vertices. The high computational cost restricts this model from being applied to enterprise-scale applications. When a graph becomes large, it is impractical to evaluate and generate the augmented paths between every pair of vertices.

To address this limitation, we propose a novel **topology role convolution model** to perform topology structure convolution. The topology role convolution consists of two steps: topology role construction, and topology role convolution. Next, we detail the two steps.

5.3.3 Topology Role Construction

The main idea of topology role convolution is to summarize and model vertex topology structure with latent topology roles. We define **topology role** as a high-level representation of underlying topology structure identities of vertices, similar to the centroid of a cluster of vertices.

To initialize topology roles, we first obtain topology structure features for each vertex. Topology structure features of vertices can be generated by using vertex topology structure embedding models (see Section 2). Then we perform clustering or classification on topology structure features to discover the latent topology roles. Topology roles discovered by clustering preserve the original graph properties, while topology roles discovered by classification can be directly mapped into the task-related space (*i.e.,* vertex classification). Dummy vertices are created on the graph to represent the discovered topology roles. Then, **belongingness edges** between topology roles and their member vertices are added to retain their topology structure similarities. If two vertices are similar based on their topology structure features, then the two vertices connect to the same topology role through belongingness edges.

In our implementation, we evaluated a few ways to generate the topology structure features, *e.g.,* Struc2vec [104], Graphwave [33] and Role2vec [8]. Based on the features, we discover the initial latent topology roles with K means or regression classifier.

5.3.4 Topology Role Convolution

Convolution with topology roles simultaneously extracts the topology roles' representations and fine-tune their members' embeddings through the con-

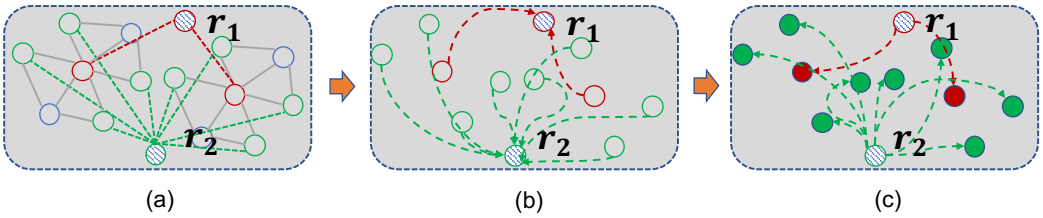


Figure 5.2: (a) Topology roles are represented by shaded vertices (r_1, r_2). Member vertices connect to topology roles through belongingness edges (edges in green and red for two topology roles). (b) First stage information propagation. Information is propagated from member vertices to topology roles. (c) Second stage information propagation. Information is propagated from topology roles to member vertices.

volution process. The convolution consists of information propagation in two stages, as illustrated in Figure 5.2.

In the first stage, the information is propagated from member vertices to the topology roles through belongingness edges. This process learns/updates the embedding representations for topology roles. The convolution on each topology role $r_i \in R$ is expressed by:

$$r_i^{l+1} = \sigma\left(\frac{1}{|m_r(i)|} \cdot W \sum_{i \in m_r(i)} h_i^l\right) \tag{5.3}$$

where l denotes the $l-th$ convolution layer, W is the weight parameter, and $m_r(i)$ denotes the member vertices of role i. Similarly, the input to the successive topology role convolution layers ($l > 1$) are the unified vertex embedding $h \in H$.

In the second stage, the information is back-propagated from topology roles to their member vertices. This process fine-tunes the member vertices' embeddings from their topology roles. Here, we assume that the underlying topology roles in the graph are non-overlapping. Thus we use a sharing scheme for this back-propagation. That is, vertices inherit the topology structure embeddings directly from their connected roles.

In implementation, the overall topology role convolution can be transformed into corresponding matrix multiplications as follows:

$$R^{l+1} = \sigma(\hat{A}_t H^l W) \tag{5.4}$$

where R^{l+1} is the embedding matrix of topology roles and $\hat{A}_t$ is the $|R| \times |N|$ normalized membership matrix for topology roles. Then, we compute the topology embedding for each member vertex by

$$F_t^{l+1} = \hat{A_t^T} R^{l+1} \tag{5.5}$$

where $(\cdot)^T$ denotes the transpose function. Note that, we share the topology role embedding among all its members, and we assume topology roles are non-overlapping, thus $\hat{A_t^T} = A_t^T$.

The topology role convolution is an effective way to model topology structure similarity between vertices. The strategy of using the topology role preserves the information propagation paths for all vertices that are similar in topology structures. As the convolution follows the belongingness edges and each vertex has only one belongingness edge to its corresponding topology role, the complexity for the convolution is $O(m + n)$ for a graph with n vertices and m topology roles. Note that, the total number of topology roles is bounded by $O(n)$. In the worst case, each vertex is modeled by an individual topology role. But in this case, the topology structure information becomes trivial as there is no pair of vertices having similar topology structure.

With the T-GCN module, the proposed dual-path convolution framework is generalizable to model other types of vertex properties besides topology structure. It is also extendable to a multi-path convolution framework when there are more properties to be modeled.

5.4 Multi-Head Graph Self-Attention

Based on previous works [123,124], we use a modified multi-head self-attention mechanism to enhance the correlation, and to adjust the importance, between

connectivity relationship and topology structure similarities, in each layer of dual-path convolution. For simplicity, we denote vertex connectivity property by c and topology structure similarity property by t. We obtain the attention coefficients for each property $j \in \{c, t\}$ of vertex i as follows:

$$e_{ij}^{l+1} = LeakyReLU\left(\alpha \left[Wh_i^l \| Wf_{ij}^{l+1}\right]\right), j \in \{c, t\} \tag{5.6}$$

Again, l denotes the $l-th$ convolution layer; h_i is the unified embedding and f_{ij} is either the output from C-GCN or T-GCN, *i.e.,* $j \in \{c, t\}$. α is the weight of the transformation network while $LeakyReLU$ is the nonlinearity function. $\|$ denotes concatenation. Following [124], a shared linear transformation W is applied to the input features h_i and f_{ij} to enhance the expressive power. As e_{ij}^{l+1} indicates the importance of the information to vertex i, the attention weight can be naturally calculated with the SoftMax function:

$$a_{ij}^{l+1} = SoftMax_{ij}\left(e_{ij}^{l+1}\right) = \frac{\exp\left(e_{ij}^{l+1}\right)}{\sum_{k \in c,t} \exp\left(e_{ik}^{l+1}\right)} \tag{5.7}$$

Thus the unified embedding is obtained by:

$$h_i^{l+1} = \sigma\left(\sum_{j \in \{c,t\}} a_{ij}^{l+1} W f_{ij}^{l+1}\right) \tag{5.8}$$

Further, we adapt multi-head attention mechanism to jointly extract different types of information from multiple representation subspaces [68, 125]. Similar to [124], the K head attentions are concatenated and we obtain the unified embedding as:

$$h_i^{l+1} = \|_{k=1}^{K} \sigma\left(\sum_{j \in \{c,t\}} a_{ij}^{l+1,k} W f_{ij}^{l+1,k}\right) \tag{5.9}$$

Different convolution layers contain different hops of neighborhood information, thus we employ this attention on each layer of dual-path convolution. The attention mechanism on each convolution layer is illustrated in

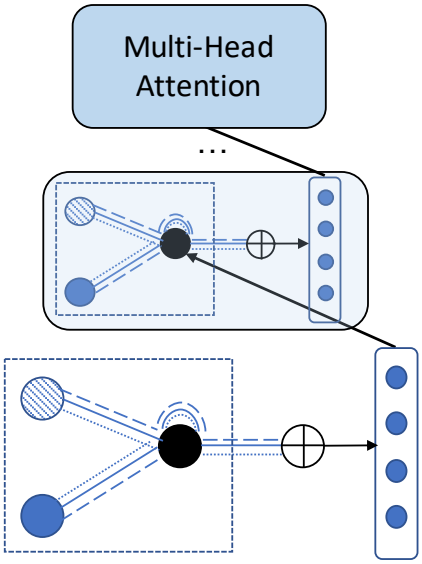


Figure 5.3: Multi-head attentions in the different layers of convolution (Best viewed in color). The dark vertex denotes the unified embedding from the previous layer. The blue and the shaded vertices denote output embedding from C-GCN and T-GCN respectively.

Figure 5.3. Following Equation 5.9, the attention on each layer takes the previously unified embedding into consideration, thus preserves the dependency information through layers. In our experiments, we employ single-head attention in the last dual-path convolution layer to generate unified embedding with fixed dimensions.

5.5 Classification and Optimization

Finally, we pass the unified embeddings into a classification network to do prediction. The prediction layer of our model is computed as follows:

$$\begin{aligned} h'_i &= Elu(h_i) \\ \hat{y}_i &= LogSoftMax(h'_i) \end{aligned} \tag{5.10}$$

According to [151] and also our empirical evaluations, it is beneficial to add an additional normalization layer on top of the unified embedding for

classification. With normalization, the predictions are expressed as:

$$\begin{aligned} h'_i &= h'_i / \|h'_i\|_2 \\ \hat{y}_i &= log\left(\frac{\exp(h'_i)}{\sum \exp(h'_i)}\right) \end{aligned} \tag{5.11}$$

The network is trained by using standard multi-class negative log-likelihood loss in tandem with the log-SoftMax in Equation 5.11 [57]. The negative log-likelihood loss is expressed as

$$L = -\sum_{i=1}^{n} \left[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)\right] \tag{5.12}$$

where y_i is the ground-truth label, and $\hat{y}_i$ is the output of the network.

5.6 Experiments on Open Datasets

In our experiments on open networks, we evaluate the DP-GCN classification model against state-of-the-art baselines. More importantly, we aim to find out through experiments whether the T-GCN module can effectively utilize the topology structure information for classification, and whether the multi-head self-attention contributes to model performance. Lastly, we study whether the embedding from dual-path convolution is interpretable. In Section 5.7, we conduct the study on PayPal payment network.

5.6.1 Experiment Setting and Results

We first evaluate our model on the task of vertex classification on four publicly available network datasets and two synthetic datasets, against strong baselines. All the four open network datasets come with ground-truth labels for validation.

Datasets. We utilize four network datasets that take both connectivity and topology structure into consideration, for vertex classification. Among the four datasets, **Euro**, **Brazil** and **USA** are airline networks. Those networks

Table 5.1: Statistics of the four open networks

Dataset	Vertices	Edges	Classes
Euro	399	5,995	3
Brazil	131	1,074	3
USA	1,190	13,599	3
BA	804	46,410	5
Acmark1	1,024	22,008	6
Acmark2	4,096	116,012	12

encode direct flights between different airports. The label to each airport is its level of activity, which expects the topology structure equivalence due to hub-and-spoke structure existing in the datasets [33, 104]. The **BA** dataset is a labeled network for vertex classification research from [105]. The two synthetic networks **AcMark1** and **AcMark2** are generated by [81], and are included in our experiments for comprehensiveness. Data statistics are reported in Table 5.1.

Table 5.2: Evaluation on four open networks and two synthetic datasets; best results are in boldface and second best highlighted.

	Accuracy					
Dataset	Euro	Brazil	USA	BA	Acmark1	Acmark2
Struc2vec	0.5875	0.6296	0.5336	0.2298	0.2633	0.1121
Graphwave	0.4375	0.4444	0.5756	0.1677	0.1361	–
Role2vec	0.3625	0.5185	0.5336	0.2174	0.4704	0.5296
RiWalk	0.5750	**0.8148**	0.6008	0.1988	0.2146	0.1146
Node2vec	0.2625	0.3333	0.4790	0.2360	0.6124	0.5170
Deepwalk	0.2750	0.4074	0.4958	0.2484	0.6479	0.5202
GCN	0.4000	0.4814	0.4621	0.2236	0.6745	0.5000
GAT	0.4242	0.5681	0.5038	0.2236	0.5088	0.3934
DEMO-Net	0.5250	0.5925	0.5882	0.2733	0.5024	0.5451
AM-GCN	0.3750	0.4815	0.3319	0.2174	<u>0.7610</u>	0.4183
DP-GCN-SV	0.6125	<u>0.7407</u>	0.5756	**0.2857**	**0.7658**	<u>0.5707</u>
DP-GCN-GW	<u>0.6250</u>	0.7037	<u>0.6050</u>	0.2670	0.7463	–
DP-GCN-RV	0.5125	0.6667	0.5546	<u>0.2795</u>	0.7365	**0.5731**
DP-GCN-RiW	**0.6500**	0.7037	**0.6134**	0.2546	0.7560	0.5597

Table 5.3: Evaluation on four open networks and two synthetic datasets; best results are in boldface and second best highlighted.

	Fscore					
Dataset	Euro	Brazil	USA	BA	Acmark1	Acmark2
Struc2vec	0.5499	0.4071	0.5081	0.2235	0.2628	0.0930
Graphwave	0.1522	0.3117	0.5734	0.1611	0.1371	–
Role2vec	0.2063	0.3436	0.4917	0.2175	0.4536	0.4901
RiWalk	0.4817	0.5583	0.3646	0.1845	0.1870	0.1001
Node2vec	0.2210	0.1750	0.4767	0.2317	0.5669	0.5059
Deepwalk	0.1847	0.2241	0.4931	0.2474	0.6134	0.5134
GCN	0.3677	0.3000	0.4535	0.0731	0.6666	0.4663
GAT	0.3707	0.4399	0.4769	0.1416	0.4736	0.3443
DEMO-Net	0.5454	0.5289	0.5573	<u>0.2498</u>	0.4918	<u>0.5494</u>
AM-GCN	0.1873	0.1625	0.2275	0.1141	0.7324	0.3447
DP-GCN-SV	0.5818	**0.6617**	0.5623	**0.2531**	**0.7491**	0.5269
DP-GCN-GW	<u>0.5858</u>	0.4875	<u>0.5888</u>	0.2032	0.7176	–
DP-GCN-RV	0.4779	<u>0.6324</u>	0.5398	0.2346	0.7158	**0.5518**
DP-GCN-RiW	**0.5981**	0.5943	**0.6029**	0.1733	<u>0.7389</u>	0.4994

Comparison Methods. We consider both traditional and state-of-the-art graph-based methods as baselines. As different baselines use different strategies to study the graph, we ensure the baselines in our experiments demonstrate sufficient diversity. We include the following 10 baselines in our evaluation. **Struc2vec** [104] is a widely used vertex topology structure embedding method. It encodes topology structure explored by random walk. **Graphwave** [33] learns vertex topology structure representation by diffusion patterns. **Role2vec** [8] captures vertex topology structure similarity by analyzing vertex motif patterns. **Riwalk** [78] models vertex topology structure by degree and relative position. **Deepwalk** [43] is one of the most popular vertex connectivity embedding method that is based on the skip-gram model. **Node2vec** [97] extends the skip-gram model by capturing breadth-first and depth-first search-based neighborhood relationship. **GCN** [61] is a classic graph convolution neural network. **GAT** [124] adapts self-attention to measure the neighborhood importance when performing vertex convolu-

tion. **DEMO-Net** models vertex degree specifically and **AM-GCN** models vertex feature with KNN graph to better fusing its features and connectivity.

To summarize, Struc2vec, Graphwave, Role2vec and Riwalk are vertex topology structure-based methods, Deepwalk and Node2vec are connectivity-based methods. GCN, GAT, DEMO-Net and AM-GCN are graph neural networks.

In our experiments, we split data for evaluation. Specifically, 67% of vertices are used for training and the remaining 33% of vertices for testing. For GCN, GAT, DEMO-Net and DP-GCN, we use 2-layers of convolution and 120 hidden dimensions. For AM-GCN, we use 1-layer of convolution and perform parameter search for k (from 2 to 7) to build KNN graph. We use the Identity matrix as the initial feature input. For graph neural networks, we follow the standard strategy for training [61, 124]. In specific, we build the entire graph for convolution, but only back-propagate the loss from the training vertices. For traditional methods, we implement the same classifier from DP-GCN and add additional dense layers for classification. We perform parameter search for hyper parameters (*e.g.,* number of hidden dimensions, number of layers) and adapt the optimal parameters for different datasets correspondingly.

Recall that the T-GCN module requires initialization of topology roles. We evaluate four different ways of topology roles initialization. Specifically, we initialize the topology roles in DP-GCN from Struc2vec (SV), Graphwave (GW), Role2vec (RV) and Riwalk (RiW) by k-means clustering ($k = 100$) respectively. Accordingly, we have four variants of DP-GCN and we denote them by **DP-GCN-SV**, **DP-GCN-GW**, **DP-GCN-RV** and **DP-GCN-RiW** respectively.

Experimental Results. We report both classification accuracy and Fscore for quantitative evaluation in Table 5.2 and Table 5.3 [1].

[1]We did not manage to get results for Graphwave on Acmark2, due to disconnectivity of the network.

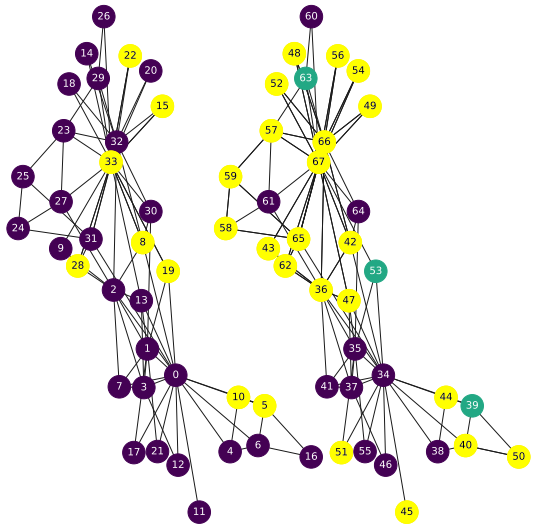
Table 5.4: Ablation Analysis

Module change	Fscore					
Dataset	Euro	Brazil	USA	BA	Acmark1	Acmark2
`DP-GCN (Avg)`	**0.5609**	0.5939	**0.5734**	**0.2160**	**0.7303**	**0.5260**
`Remove C-GCN`	0.5150	0.4993	0.5121	0.1179	0.1506	0.2222
`Remove T-GCN`	0.3969	0.4460	0.4852	0.1376	0.6676	0.4349
`Remove Attention`	0.4903	0.4575	0.5541	0.1603	0.6981	0.4947
`1-Head Attention`	0.5395	**0.6030**	0.5684	0.1333	0.6931	0.5225
`Single-Layer GCN`	0.5074	0.5749	0.5548	0.1853	**0.7303**	0.4565

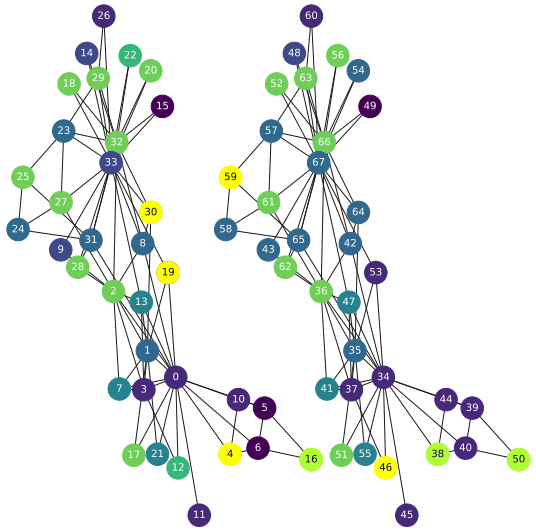
All four DP-GCN variants clearly outperform existing state-of-the-art models over a large margin. Specifically, the improvements are about 5% to 11%. Especially on the Brazil and Acmark1 networks, our methods have an average $\approx 8\%$ improvement in accuracy and Fscore over all other baselines. This result indicates the importance of combining both connectivity and topology structure information in vertex classification. We also observe that on Euro, Brazil, and USA datasets, the topology structure information is more important to the task, as the baselines focusing on topology structure information tend to have better performance. For BA and Acmark1 networks, we observe the opposite. However, DP-GCN achieves consistently best results as our model utilizes both sides of information and uses self-attention to align them properly. Overall, results of all variants DP-GCN-SV, DP-GCN-GW, DP-GCN-RV and DP-GCN-RiW are comparable.

5.6.2 Ablation Analysis

We now conduct ablation studies to evaluate the effectiveness of each module in the proposed DP-GCN model. Table 5.4 reports the results with different settings. As there are different ways to initialize the topology roles, we use the average Fscore of DP-GCN-SV/GW/RV/RiW when applicable. For instance, the first row in Table 5.4 is the average results of the four variants of DP-GCN.



(a) vertex representation from GCN



(b) vertex representation from DP-GCN

Figure 5.4: vertices representations from GCN, DP-GCN without training (Best viewed in color)

To evaluate the effectiveness of the component modules, we start with removing each individual module at a time, *i.e.,* C-GCN, T-GCN, and the multi-head attention module. When the multi-head attention module is removed, we combine the outputs from C-GCN and T-GCN unweighted. Alternatively, we also evaluate the setting of replacing the multi-head attention module by a single-head setting, or by using a single convolution layer.

We make four observations from Table 5.4. First, both T-GCN and C-GCN modules play important roles in DP-GCN as the performance decreases significantly after removing either of two modules. Second, the attention mechanism is also an important component in DP-GCN. It is intuitive that different types of vertex information take different levels of importance in the prediction task. Combining multiple types of vertex information without carefully analyzing their importance may not be effective in the classification task, and sometimes may even introduce noise to the model. Third, tuning the number of heads for attention is not trivial. It helps the model to better learn the information alignment towards the task-specific objective from multiple representation subspaces. Finally, we observe that the C-GCN module does not contribute much to the Euro, Brazil, and USA datasets. This is due to the characteristics of the datasets. In those three datasets, the topology structure information is more important for the corresponding classification objective. However, on BA, Acmark1, and Acmark2 datasets, the C-GCN module contributes more than the T-GCN module. Overall, this study suggests that all modules in the architecture are crucial for achieving the best classification results. If we combine the information modeled by both C-GCN and T-GCN modules with a proper attention mechanism, we are able to achieve the best classification results over all datasets.

5.6.3 Analysis on the Embeddings

We now focus on the interpretability of the dual-path convolution process in DP-GCN, which is important in providing insights into the inner workings of

Table 5.5: Risky seller detection on PayPal network datasets. Best results are in bold and second best underlined.

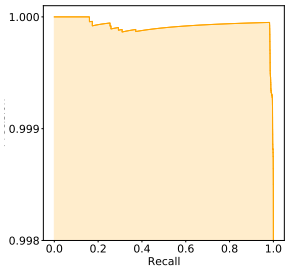
Dataset	SG-Jan		
Model	Precision	Recall	Fscore
Role2vec	0.5230	0.9623	0.6777
Riwalk	0.5263	0.9555	0.6787
Deepwalk	0.5100	0.8954	0.6499
GCN	0.5183	0.8536	0.6450
DP-GCN-RV	<u>0.5509</u>	<u>0.9793</u>	<u>0.7052</u>
DP-GCN-RiW	**0.5667**	**0.9866**	**0.7199**

Dataset	SG-Apr		
Model	Precision	Recall	Fscore
Role2vec	0.5453	0.9684	0.6977
Riwalk	0.5307	<u>0.9688</u>	0.6858
Deepwalk	0.5100	0.8857	0.6473
GCN	0.5206	0.8843	0.6554
DP-GCN-RV	**0.6538**	0.9015	**0.7579**
DP-GCN-RiW	<u>0.5580</u>	**0.9850**	<u>0.7124</u>

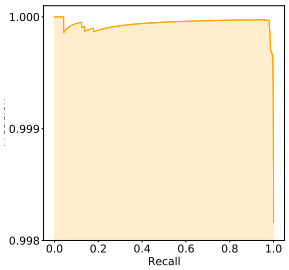
Dataset	SG-Jul		
Model	Precision	Recall	Fscore
Role2vec	0.5228	<u>0.9608</u>	0.6771
Riwalk	0.5244	0.9480	0.6752
Deepwalk	0.5083	0.8543	0.6374
GCN	0.5203	0.9036	0.6603
DP-GCN-RV	**0.5606**	0.9606	**0.7080**
DP-GCN-RiW	<u>0.5260</u>	**0.9751**	<u>0.6834</u>

our model. In specific, we analyze the vertex representations generated from the DP-GCN on the well-known Zachary's Karate Club network [142].

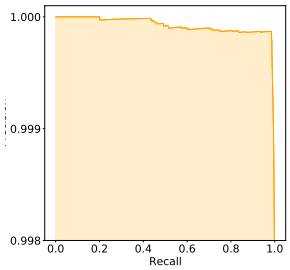
We follow [104] to construct a mirrored network of the same club network. Then we obtain a unified network which consists of the original and the mirrored networks (*i.e.,* two copies of the same Karate Club network). In this unified network, each vertex is assigned a unique vertex id; so corresponding vertices in the original and mirrored networks will be assigned different vertex ids, as shown in Figure 5.4. We build DP-GCN with randomly initialized weights on this unified network. To enable role convolution, we made a simple assumption that the same corresponding vertices in the original and the mirrored networks have the same topology structure (*i.e.,* modeled by the same topology role). Without any training, we input the adjacency matrix of the unified Zachary's karate club and the Identity matrix $X = I$ (*i.e.,* without any additional features) to the model. Then DP-GCN performs both connectivity and topology role convolution on all vertices in the unified



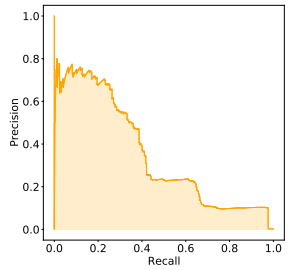
(a) SG-Jan for non-risky sellers



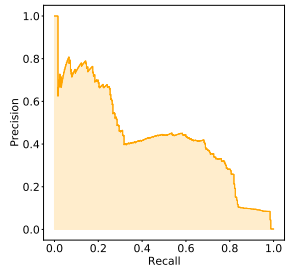
(b) SG-Apr for non-risky sellers



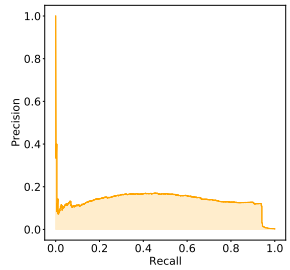
(c) SG-Jul for non-risky sellers



(d) SG-Jan for risky sellers



(e) SG-Apr for risky sellers



(f) SG-Jul for risky sellers

Figure 5.5: DP-GCN precision-recall curve on PayPal network datasets

network. The final vertex representations are obtained through a random weighted attention which combines both sides of information (as we do not have the task specific loss function here to guide the attention). As a comparison, we generate vertex representations from the standard GCN through the same process. The vertex representations are shown in Figure 5.4, where different labels are in different colors.

We observe that GCN tends to group vertices based on connectivity relationships, thus vertices within the same local community have very close representations. The convolution from GCN does not take topology structure similarity into consideration. The representations generated from DP-GCN, on the other hand, consider both vertex connectivity and vertex topology structure similarity. As an example, the corresponding vertices {18,29,32} and {52,63,66} are densely connected in each of their own local communities.

They have the same topology structure as they reside at the same positions in original and mirrored club networks. Thus they have been learned to have a close representation by DP-GCN, but not GCN. Results in Figure 5.4 verifies the effectiveness of using dual-path convolution to model these two types of information simultaneously in a network.

5.7 Risky Seller Detection on PayPal Payment Network

In this section, we evaluate our model on payment networks from PayPal. The task is to detect risky sellers, *i.e.,* the sellers who result in revenue loss due to various reasons (*e.g.,* fraud, bankruptcy, bad suppliers).

5.7.1 PayPal Internal Datasets

We conduct experiments on three network datasets with ground-truth labels from PayPal: **SG-Jan**, **SG-Apr** and **SG-Jul**. Each network contains a subset of one-week payment activities of anonymous sellers (seller ids are encrypted) in Singapore in January, April, and July 2020 respectively. The sizes of the three datasets are comparable and each contains about half a million vertices and half a million edges.[2] The risky seller labels are provided by an internal department. The data is imbalanced, due to the fact that only a very small number of sellers are risky sellers.

Due to memory constraints, we manage to compare our model DP-GCN with Role2vec, Riwalk, Deepwalk, and GCN. Again, Role2vec, Riwalk are vertex topology structure-based methods, while Deepwalk is vertex connectivity-based method. GCN is the standard graph convolution network. We use the same number of layers and hidden layer dimensions for DP-GCN and GCN. We perform parameter search on the over- and under-sampling ratio for the imbalanced labels and use the same ratio for all methods. For DP-GCN,

[2]Due to enterprise privacy, we mask the exact numbers.

we use topology roles generated from Role2vec and Riwalk embeddings, and use 1000-dimensional binary vectors as vertex initial features. For GCN, we evaluate the performance by using either 1000-dimensional binary vectors or topology structure embeddings as vertex initial features, and then select the best one to represent its final performance. As the labels are much imbalanced, we use macro-averaged precision, recall, and Fscore for performance evaluation, instead of accuracy. Macro-averaged metrics are known to be more appropriate for imbalanced data as it avoids dominance of majority classes [26, 148].

5.7.2 Result Analysis

Results of risky seller detection are reported in Table 5.5. DP-GCN is consistently at the top on all the three datasets by Fscore. Role2vec and Riwalk are the best performing models among baselines. The results indicate that using topology structure information is effective in the task of risky seller detection. We also observe that simply feeding topology structure embeddings as the initial features to graph convolution network leads to very marginal improvement. This further emphasizes the importance of using dual-path convolution to model the different types of information in the graph neural network.

To better analyze our solution, Figure 5.5 shows the precision-recall curve for each class by DP-GCN. The curve is produced by using the outputs after log-SoftMax and the outputs give us a sense on how the model makes predictions. As expected, on a typical imbalanced prediction task, our model achieves very high and stable performances on the predictions of non-risky sellers, on all datasets. Predicting risky sellers is a more challenging task, because of data imbalance. Our model is able to discover most of the risky sellers with acceptable precision. That is, our solution is able to effectively detect risky sellers while minimizing the number of false positives on non-risky sellers. Our results on enterprise-scaled data suggests that combining

both connectivity and topology structure information for risky seller detection is a promising direction on payment network, and is well aligned with the real-world enterprise needs.

5.8 Conclusion

In this chapter, we propose an interactive dual-path graph convolution network to explore both vertex connectivity and topology structure information for vertex classification tasks. We propose a generalizable role-based convolution to explicitly model the topology structure similarity of vertices with latent topology roles. Furthermore, we align vertex connectivity and topology structure information with multi-head graph self-attentions to reduce the gaps between the two types of information. Via visualization and qualitative analysis, we show that our proposed model achieves highly competitive results on both open datasets and PayPal risky seller detection application.

Chapter 6

Cyber Anomaly Detection on CSP Network[1]

6.1 Introduction

The Content Security Policy(CSP) is the most accepted and efficient cyber-security phenomena against cross-site scripting (XSS), clickjacking, and code injection [1]. In specific, CSP is used to prevent data requests from unknown locations to be loaded. The violations are blocked by rules in real-time and millions of violations are generated on daily basis, results in relatively high false positives rates.

To detect real anomaly activities on the CSP network, a bipartite graph [32] can be built from CSP logs. Each **external domain** URL is a source vertex, and each **internal service server** is a target vertex. A request from an external domain to a service server is represented by an edge. Many current studies on graph anomaly detection focus on connectivity inconsistencies, *i.e.,* anomaly vertices having very different connectivity patterns from the majority of vertices on graph. From empirical studies, the malicious on the CSP log graph usually has the following characteristics: (i) A new or existing external domain has a sudden burst of traffics, (ii) a new or existing external domain has uncommon traffic patterns with internal servers, and (iii) a

[1]The work in this chapter has been published in IEEE International Conference on Data Mining, 2020.

sudden burst in traffics for an internal server. Thus, anomalies in the CSP network may arise from not only connectivity patterns but also **burstiness** in network activities. We note that burstiness does not adhere to the common definition of graph anomaly, thus existing anomaly detection methods mostly do not consider the impact of burstiness in graph. Hence, existing solutions are not directly applicable to our application. In our problem setting, we consider both burstiness and connectivity patterns.

Burstiness: Burstiness refers to the intermittent increases and decreases in activities or frequency of an event. Burstiness detected on both sides of a bipartite graph is important for the anomaly detection task. For example, a sudden burst of traffics sent from a single external domain may indicate that this computer is suspicious. However, attackers may hide their traces and perform attacks with multiple domains. In this case, burstiness from a server enables us to identify those attacks.

Connectivity Pattern: Connectivity pattern refers to the connectivity behavior between source vertices U and target vertices V. Specifically, it measures how U interacts with V. Again, in our CSP network, the connectivity pattern indicates an external domain's preferences over servers' services. Connectivity pattern helps us to identify anomalies whose connectivity behavior deviates from the underlying majority in the network.

In this chapter, we propose a framework named **BEA**, which first detects **B**urstiness (in a given time period), then models dynamic connectivity patterns through bipartite graph **E**mbedding, and finally detects **A**nomaly on the graph. Given source vertices U and their connected targets V, our framework computes anomaly scores for all $u \in U$ and identifies the anomaly ones who have high anomaly scores. The key difference between our solution and existing ones is that BEA handles both dynamics and burstiness in bipartite graph.

We evaluate BEA on three public open graph data sets. Further, we evaluate BEA on PayPal CSP network to detect the real anomaly. Through

experiments, we demonstrate the effectiveness of BEA and its superiority over diverse state-of-the-art baselines.

In summary, we have made the following contributions in this research:

- We propose BEA, a novel unsupervised learning framework for anomaly detection on dynamic bipartite graphs. It detects anomaly based on two important assumptions: burstiness and connectivity pattern. Experiments on three open data sets and PayPal CSP log network demonstrate the effectiveness of BEA.

- We define two-step pattern embedding on bipartite graph, which enables graph convolution process on different types of vertices in bipartite graphs.

- We introduce a strategy to model vertices from real-world applications into source vertices and target vertices by their characteristics. This strategy motivates us to detect burstiness on sources and targets, which is more reasonable and effective.

6.2 Bipartite Graph Definition

A **bipartite graph** is represented by $G = (U, V, E, W)$. U and V denote the two sets of vertices, which we call **source** and **target** vertices, respectively. Here, we consider the relatively dynamic side on a bipartite graph as the source vertices *e.g.,* external domains in the CSP network. Accordingly, the target vertices are the relatively static side of the bipartite graph, which are typically managed by enterprises *e.g.,* internal service servers in CSP network. The connections between U and V are represented by edges $E \subset U \times V$. The connectivity of this graph can be stored in a $|U|$ by $|V|$ adjacency matrix A where $A[u_i][v_j] = w$; $w \in W$ is the weight on the edge.

To detect burstiness, we need to observe the changes in the graph connections for a time period. For this purpose, we define a time window Δ_t.

Given a snapshot of a bipartite graph G at time t_0, and the connection changes between source and target vertices from t_0 to $t_0 + \Delta_t$, we aim to detect the anomalies among the source vertices U. Target vertices V are assumed managed by enterprises, and hence not anomalies.

6.3 The BEA Framework

In this section, we present our burst-driven bipartite graph anomaly detection framework BEA. Our framework consists of three main components: *burstiness detection*, *dynamic pattern embedding*, and *anomaly detection*. Within a time window Δ_t, the burstiness detection component detects burst activities among both source and target vertices. As a result, a **burst graph** is constructed at the end of the time window. The edges in the burst graph, model the connections and burstiness between source and target vertices. Finally, the vertices' representations are passed to our anomaly detection component to detect anomalies. Next, we detail each component.

6.3.1 Burstiness Detection

We now introduce the burstiness detection mechanism, which plays an essential role in our framework. Different from existing methods which simply treat source and target vertices equally, our framework leverages the fact that source and target vertices usually have significantly different burstiness behaviours. Therefore, we calculate burstiness scores for source and target vertices in a different manner.

6.3.1.1 Burstiness of Source Vertex u

Recall that source vertices refer to the relatively dynamic side of a bipartite graph. In different time windows (*i.e.,* Δt's), the set of source vertices could be very different. Burstiness of a source vertex, therefore, is related to how active this vertex is, comparing to expectation. Here, we measure how active

a vertex is by the number of target vertices it connects to, or the frequency of its connections.

Specifically, we model the probability of a source vertex with frequency $f_{u,\Delta t}$ within the time window Δt by a Gaussian distribution [29].

$$\begin{aligned} P(f_{u,\Delta_t}) &\sim \mathcal{N}_{\Delta_t}(E[U_{\Delta_t}],\, \sigma^2[U_{\Delta_t}]) \\ E[U_{\Delta_t}] &= \frac{N_{\Delta_t}}{|U_{\Delta_t}|} \\ \sigma^2[U_{\Delta_t}] &= \frac{\sum_{u \in U_{\Delta_t}} (u - E[U_{\Delta_t}])^2}{|U_{\Delta_t}|} \end{aligned} \tag{6.1}$$

In Equation 6.1, U_{Δ_t} denote all source vertices in the current time window Δ_t. The expected frequency and variance for each source vertices are denoted by $E[U_{\Delta_t}]$ and $\sigma^2[U_{\Delta_t}]$ respectively.

Definition 6.3.1 (Source Vertex Burstiness)**.** *A source vertex u is burst in a time window Δ_t, if its frequency $f_{u,\Delta_t} > E[U_{\Delta_t}] + \alpha\sigma[U_{\Delta_t}]$, where $\sigma[U_{\Delta_t}]$ is the standard deviation.*

From empirical rule of Gaussian distribution, 95% data instances fall within two standard deviations.

$$Burst(u, \Delta t) = S(10 \times \frac{f_{u,\Delta_t} - E[U_{\Delta_t}]}{2 \times \sigma[U_{\Delta_t}]}) \tag{6.2}$$

where $S(\cdot)$ denotes Sigmoid function and 10 is a scalar. The value 10 is chosen here because Sigmoid function smooths very well for a value in range of $[-10, 10]$ [66].

6.3.1.2 Burstiness of Target Vertex v

Target vertices defined in this work refer to the relatively static side on a bipartite graph, *e.g.,* servers in the computer network. Usually, the vertex set of targets do not change frequently between different time windows. Moreover, the burstiness threshold for each target vertex should relate to its usual activity level. For example, in a computer network, the burstiness threshold

for the busiest server should be much higher than a server which is idle most time.

Thus, burst detection on a target vertex is cast to the problem of finding abnormal aggregates, based on sliding time windows over its own connection streams. We need to monitor many time windows to detect those windows with significant differences from other time periods.

As each target can be either burst or non-burst, we model its burstiness with a two-mode automaton [62, 67]. Each target vertex is modeled with a mode M which takes one of the two values $\{0, 1\}$, where 0 stands for base mode and 1 stands for burst mode.

Within each time window, a target vertex's mode is not directly observed. It is computed based on its connections. As arrival time is exponentially distributed [58, 121, 132], the connections arriving for each target vertex can be modelled by exponential distribution as follows.

$$f(\delta; \lambda) = \lambda e^{-\lambda \delta} \tag{6.3}$$

$$P(\delta_v) = \begin{cases} f(\delta_v, \lambda_0), & \text{if } M = 0 \\ f(\delta_v, \lambda_1), & \text{if } M = 1 \end{cases} \tag{6.4}$$

In the above equations, δ_v is the time interval between the arriving of two adjacent connections to target vertex v. λ_0 and λ_1 are the connection arriving rates for v in its base mode and burst mode respectively. They can be easily obtained by sampling on the data. In real implementation, we may not directly observe the rate for v in its burst mode. Instead, we can practically set $\lambda_1 = \alpha\lambda_0$, where $\alpha > 0$. We define $P(\delta_v)$ as the probability of target v obtaining two adjacent connections within time interval δ. It can be drawn from Equation 6.4 with respect to its mode M.

Let N_{Δ_t} denote the total number of connections within time window Δ_t. f_{v,Δ_t} be the number of connections received by target vertex v. The probability of observing target v with frequency f_{v,Δ_t} can be modelled by binomial distribution [62].

$$P(f_{v,\Delta_t}) = \binom{N_{\Delta_t}}{f_{v,\Delta_t}} P_{v,\Delta_t}^{f_{v,\Delta_t}} \left(1 - P_{v,\Delta_t}\right)^{N_{\Delta_t} - f_{v,\Delta_t}} \tag{6.5}$$

Given N_{Δ_t} is typically very large in terms of network connections, it can be approximated reasonably with a Gaussian distribution [66].

$$P(f_{v,\Delta_t}) \sim \mathcal{N}\left(N_{\Delta_t} P_{v,\Delta_t},\ N_{\Delta_t} P_{v,\Delta_t}\left(1 - P_{v,\Delta_t}\right)\right). \tag{6.6}$$

We assign the cost for a target v with mode M in time window Δ_t as

$$\phi(v, M_{\Delta_t}) = -\ln P(f_{v,\Delta_t}) \tag{6.7}$$

The state transition cost is defined as follows, where γ is a hyperparameter for cost.

$$\tau(v, \Delta_t) = \begin{cases} \gamma \ln f_{v,\Delta_t}, & \text{from } M_{\Delta_{t-1}} = 0 \text{ to } M_{\Delta_t} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

The mode transition cost for a target v from base mode to burst mode is proportional to the number of connections it has. There is no cost for a target v to stay in its current mode or to transit from burst mode to its base mode.

Given a sequence of time windows $\{\Delta_1, \Delta_2, \cdots, \Delta_n\}$, the goal is to find the sequence of modes $\{M_{\Delta_1}, M_{\Delta_2}, \cdots, M_{\Delta_n}\}$ for each target v that minimizes the cost

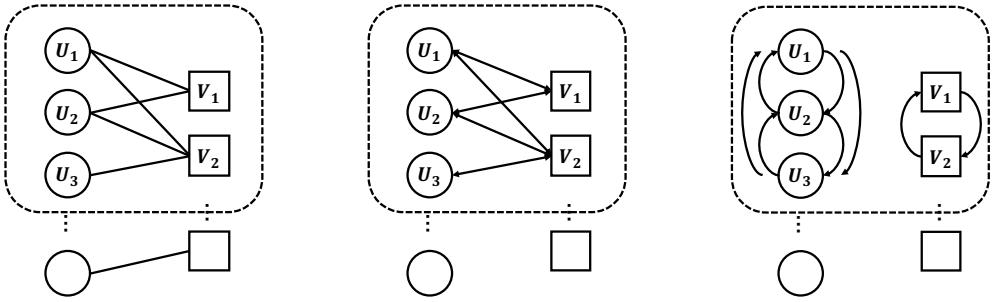
$$c(v, \Delta_t) = \sum_{t=0}^{n} \phi(v, M_{\Delta_t}) + \sum_{t=0}^{n} \tau(v, \Delta_t) \tag{6.9}$$

This optimization can be efficiently solved with a greedy approach by assuming M_{Δ_t} only depends on $M_{\Delta_{t-1}}$ and is independent of previous states. This assumption has been proven reasonable in many applications as it efficiently captures the short term behaviour [67]. With this assumption, mode M_{Δ_t} can be predicted by

$$\arg\min_{M_{\Delta_t}} c(v, \Delta_t) = \phi(v, M_{\Delta_t}) + \tau(v, \Delta_t) \tag{6.10}$$

The burst score for a target vertex v in time window Δ_t is given by:

$$\begin{aligned} Cost(v, \Delta_t) &= \begin{cases} 0, & M_{\Delta_t} = 0 \\ \phi(v, 0) - \phi(v, 1), & M_{\Delta_t} = 1 \end{cases} \\ Burst(v, \Delta_t) &= \frac{e^{Cost(v,\Delta_t)}}{\sum_{v' \in V_{\Delta_t}} e^{Cost(v',\Delta_t)}} \times |V_{\Delta_t}| \end{aligned} \tag{6.11}$$



(a) Bipartite graph (b) 1st step propagation (c) 2nd step propagation

Figure 6.1: An illustration of the proposed two-step propagation on bipartite graph. The three example source vertices have similar connectivity patterns.

In above equations, $\phi(v,0) - \phi(v,1)$ calculates the difference in cost by adapting burst mode instead of base mode. Then the cost is mapping to burstiness score by feeding the cost into a $softmax$ function.

6.3.1.3 Burst Graph Construction

Now we construct a burst bipartite graph with U_{Δ_t} and V_{Δ_t} by initializing the weight on each edge as follows.

$$w_b(u,v) = w(u,v) \times Burst(v, \Delta_t) \tag{6.12}$$

Here, $w(u,v)$ is the original weight on an edge from u to v. The constructed burst graph retains all connection information while emphasising the connections on burst target vertices.

6.3.2 Dynamic Pattern Embedding

Given a dynamic bipartite graph, we now try to efficiently encode each source vertex into a vector representation based on its connectivity pattern and burstiness scores. To this end, we propose **two-step pattern embedding** on bipartite graphs basing GraphSage [47]. We initialize the embeddings of the two kinds of vertices as follows:

Source Vertices Embedding Initialization. The goal of the embedding is to ensure source vertices with similar connectivity patterns to have close embeddings. We initialize the embedding h_u^0 for each source vertex $u \in U$ with its weighted connection relationships in time window Δ_t. The embedding for vertex u can be initialized as any transformation of $A[u]$ from adjacency matrix A.

Target Vertices Embedding Initialization. We define the embedding of a target vertex as the expected connectivity patterns from all source vertices connected to it. In this way, the information propagation between sources and targets is meaningful as they interchange information for local connectivity patterns. Thus it is initialized by

$$h_v^0 \leftarrow MEAN\left(h_u^0, \forall u \in N_+(v)\right); \tag{6.13}$$

Here, $N_+(v)$ denotes all source vertices who connect to v, $MEAN(\cdot)$ is the mean aggregation function that takes element-wise mean of the embedding of $h_u^0, \forall u \in N_+(v)$.

After we initialize the embeddings for both vertices in U and V, we apply a two-step training process. In each of the two steps, we do the following (note $z_u^0 = h_u^0$ and $z_v^0 = h_v^0$).

$$\begin{aligned}
h_{N(u)}^k &= AGG\left(w_b(u,v) \times z_v^{k-1}, \forall v \in N_+(u)\right); \\
h_u^k &= \sigma(W^k \cdot Concat(h_{N(u)}^k, z_u^{k-1})); \\
z_u^k &= (h_u^k / \left\|h_u^k\right\|_2) \\
h_{N(v)}^k &= AGG\left(w_b(u,v) \times z_u^{k-1}, \forall u \in N_+(v)\right); \\
h_v^k &= \sigma(W^k \cdot Concat(h_{N(v)}^k, z_v^{k-1})); \\
z_v^k &= (h_v^k / \left\|h_v^k\right\|_2)
\end{aligned} \tag{6.14}$$

In the above equations, k is the step. $w_b(u,v)$ is the weight of edge between u, v on the burst graph. z_u and z_v are the embeddings of vertices u and v respectively, after normalization from h_u and h_v derived above. AGG is the aggregation function.

The propagation takes two steps as illustrated in Figure 6.1. In the first step, the information is propagated between vertices in U and their connected vertices in V. The propagation process adjusts the embeddings of vertices in U and V according to their local expected connectivity patterns. In the second step, the information is propagated to their second-order neighbourhood, which leads to vertices in U connected to a similar set of vertices in V, to have close embeddings; the same applies to vertices in V.

To train the parameters, we adopt the loss function from graph embedding with negative sampling

$$\begin{aligned} Loss(u) = &- \sum_{u} \sum_{v \in N_+(u)} \log \sigma(z_u^T z_v) \\ &- \sum_{u} \sum_{v \in N_-(u)} k \log \sigma(-z_u^T z_v) \\ L = &Loss(u) + Loss(v) \end{aligned} \tag{6.15}$$

where k is the number of negative samples. This loss uses the inner products of embeddings between vertices in U and V to capture the likelihood of existence of each connection [39].

In our burst graph constructed, the connections between a source vertex and target vertices have been intensified by target vertices' burst scores as we showed in Section 6.3.1.3. As a result, the information propagated between source to burst targets will take more weights in the convolution process. Thus this process works as an attention mechanism, that allows the embeddings to focus on capturing the connectivity patterns from those burst target vertices.

The raw embedding for a source vertex does not contain its own burstiness information. To capture burst information, we multiply each embedding z_u with its own burstiness score which we derived from Section 6.3.1.1.

$$z_u = Burstiness(u, \Delta t) \times z_u \tag{6.16}$$

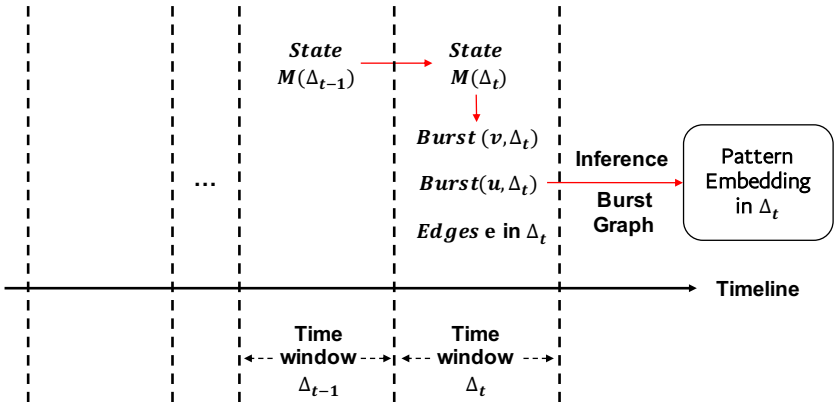


Figure 6.2: Embedding generation on time window Δt

Note that, when any previous unseen source vertex arrives in a new time window, its embedding z_u can be immediately inferred based on its connectivity with targets.

6.3.3 Anomaly Detection

Anomaly detection is a reactive process. The anomaly detection component should be able to adapt to new vertices/connections and new anomalies. It is therefore reasonable to model general patterns with multiple Gaussian mixtures for anomaly detection.

Inspired by the effectiveness and computational benefit of the deep Gaussian mixture model (DAGMM), we feed the learned embeddings into an estimation network [155] to detect the anomaly.

6.3.4 Time Complexity Analysis

Figure 6.2 shows the information propagation between time windows. The running time of the burstiness detection component mainly depends on counting connection frequency for each source vertex, and calculating the burstiness score for each target vertex. With our greedy approach, the calculation of burstiness score in time window Δ_t only depends on information propagated from Δ_{t-1}. Thus the time complexity is $O(|V|)$ which is linear to the number of vertices.

For dynamic pattern embedding, the time complexity of embedding inference is the same as GraphSage. The complexity for target vertices embedding initialization is linear to the number of source vertices connected to each target vertex. Instead of using all the connected source vertices, it is reasonable to uniformly sample a fixed-size set of source vertices. With this sampling strategy, the complexity for embedding initialization is $O(|V| \times |S|)$ where $|S|$ is a constant sample size. The anomaly detection component only contains a feedforward propagation, which is linear to the number of source vertices when the size of parameters is fixed.

6.4 Experiment on Open Networks

6.4.1 Datasets and Evaluation Metrics

We conduct experiments on three open bipartite graph datasets from different domains. Quantitative evaluation of anomaly detection system is challenging because it is difficult to construct ground-truth labels for the given settings [116]. We follow the related studies [10, 141, 147] and adopt anomaly injection in experiments. To prevent affecting the structural properties of the original graphs, a total of 5% anomalies and related edges are injected into each graph in the three datasets. **MovieLens** [2, 48] graph is constructed from user-movie ratings. The source vertices represent users and target vertices represent movies. Each edge indicates a rating from a user to a movie. **UCI Forum** [3, 94] graph is constructed based on an online community at University of California, Irvine. It captures message interactions of students with forums. **Unicode Language** [4] graph captures the relationship between languages and countries. The source and target vertices represent languages and countries respectively. Each edge indicates a language being spoken in a country.

6.4.2 Comparison Methods

On the open datasets, we compare our BEA framework with the following baselines. **Isolation-Forest** [71] and **One-Class SVM** [84] are the most widely used anomaly detection algorithms. **DAGMM** [155] is the state-of-the-art deep learning anomaly detection algorithm. **Netwalk** [141] and **GraphSage** [47] are both graph-based methods. They represent the state-of-the-art graph anomaly detection algorithm and graph embedding algorithm, respectively. To summarize, Isolation-Forest and One-Class SVM are traditional anomaly detection methods. DAGMM is a deep learning method, and both Netwalk and GraphSage are graph-based methods. For Isolation Forest, we have done a parameter search and finally run it with 100 estimators. For One-Class SVM, we adopt default "RBF" kernel and set the fraction of errors as 5% which conforms to our dataset. For Netwalk, we use the parameters recommended by the authors in the original paper. For DAGMM and GraphSage, we have done parameter searches and use the same set of parameters with BEA.

6.4.3 Results

As evaluation based on precision and recall becomes tricky for experiments with anomaly injection [10], we evaluate all methods by AUC scores and report the results in Table 6.1. Observe that the proposed BEA outperforms all baselines on all three graphs. DAGMM is the second-best performer which outperforms the traditional models (*i.e.,* One-Class SVM and Isolation Forest) by a small margin. From additional experiments (results not reported here), we observe that those non-graph based methods are more sensitive to burstiness in the graph. Because our proposed model not only considers graph connectivity patterns but also burstiness, it achieves better and more stable performance than all baselines.

Table 6.1: AUC results on the three open datasets. Best results are in bold face and the second best are underlined.

Dataset	MovieLens	UCI Forum	Unicode
`Isolation-Forest`	0.683	0.819	0.763
`One Class SVM`	0.696	0.788	0.753
`DAGMM`	<u>0.706</u>	<u>0.829</u>	<u>0.776</u>
`Netwalk`	0.523	0.496	0.606
`GraphSage`	0.607	0.466	0.641
`BEA` (ours)	**0.852**	**0.909**	**0.800**

6.5 Cyber Anomaly Detection on PayPal CSP Logs Network

After showing good results on open datasets, we now conduct experiments on real CSP log network from PayPal.

A bipartite graph can be built from CSP logs. Each **external domain** URL is a source vertex, and each **internal service server** is a target vertex. A request from an external domain to a service server is represented by an edge. From empirical studies, the malicious on CSP log graph usually has the following characteristics: (i) a new or existing external domain has a sudden burst of traffics, (ii) a new or existing external domain has uncommon traffic patterns with internal servers, and (iii) a sudden burst in traffics observed on an internal server.

DataSpii Attack. We evaluate our framework on a recent real attack called *DataSpii*.[2] For validation, we have a set of ground truth labels: a known list containing all identified domain URLs that are involved in DataSpii attack. The CSP logs used in this evaluation contain $120,012$ traffics between hundreds of external domains and internal servers.[3] The data is segmented into different time windows based on the timestamp on each traffic.

[2]https://dataspii.com/

[3]We mask the exact numbers because of data protection policy.

Table 6.2: Evaluation of BEA and baselines on PayPal CSP Logs for DataSpii attacks.

(a) Time Window = 30m

Method	Acc	Pre	Rec	Fscore	AUC	MAP
Iso-Forest	0.260	0.037	0.471	0.068	0.185	0.180
OC-SVM	0.287	0.015	0.177	0.028	0.164	0.177
DAGMM	0.409	0.024	0.235	0.044	0.288	0.205
Netwalk	0.568	0.076	0.588	0.135	0.571	0.453
GraphSage	0.581	0.065	0.471	0.114	0.513	0.462
Source Burst	0.669	0.000	0.000	0.000	0.506	0.247
Target Burst	0.193	0.041	**0.588**	0.077	–	–
BEA	**0.693**	**0.106**	**0.588**	**0.180**	**0.609**	**0.521**

(b) Time Window = 1h

Method	Acc	Pre	Rec	Fscore	AUC	MAP
Iso-Forest	0.264	0.060	0.500	0.107	0.205	0.156
OC-SVM	0.308	0.034	0.250	0.060	0.180	0.154
DAGMM	0.407	0.031	0.188	0.053	0.261	0.174
Netwalk	0.560	0.128	0.688	0.218	0.595	0.357
GraphSage	0.610	0.123	0.563	0.202	0.480	0.425
Source Burst	0.648	0.000	0.000	0.000	0.482	0.215
Target Burst	0.253	0.046	0.375	0.081	–	–
BEA	**0.681**	**0.182**	**0.750**	**0.293**	**0.728**	**0.532**

(c) Time Window = 2h

Method	Acc	Pre	Rec	Fscore	AUC	MAP
Iso-Forest	0.319	0.064	0.455	0.112	0.283	0.149
OC-SVM	0.302	0.027	0.182	0.047	0.222	0.139
DAGMM	0.388	0.016	0.273	0.091	0.174	0.133
Netwalk	0.466	0.119	0.727	0.205	0.729	0.508
GraphSage	0.552	0.127	0.636	0.212	0.782	0.619
Source Burst	**0.698**	0.000	0.000	0.000	0.537	0.205
Target Burst	0.345	0.099	0.727	0.174	–	–
BEA	0.560	**0.155**	**0.818**	**0.261**	**0.921**	**0.682**

6.5.1 Analysis of Results

The evaluation metrics used in this study include Accuracy, Precision, Recall, Fscore (F_1), Area under the ROC Curve (AUC) and Mean Average Precision (MAP). MAP is the mean of average precisions (AP). It is more interpretable than AUC as MAP not only considers the detection precision but also considers the ranks of retrieval [34].

Again, we evaluate BEA framework against all baselines that we have evaluated earlier. For the completeness of evaluation, we also implemented two more methods: **Target Burst** detects the burstiness scores for all target vertices by the burstiness detection component in BEA. Then it catches all source vertices who connect to those targets in burst mode as anomalies. **Source Burst** captures all source vertices that have a number of connections deviates more than one standard deviation from majorities as anomalies.

The results are reported in Table 6.2. From the table, we observe that BEA outperforms all baselines on anomaly detection over DataSpii attack, on almost all measures. Our results also show that graph-based methods on real computer networks provide better and consistent performance, comparing to other baselines. Purely based on statistical measures, burstiness scores on either source vertices or target vertices do not give stable results. The two traditional methods Isolation-Forest and One-Class SVM do not perform well on this dynamic graph, and DAGMM delivers comparable results as traditional methods.

6.5.2 Experiment on Dynamic Detection

We also analyze the performance of our model in a dynamic environment as described in Section 6.3.2. We perform experiments for all three time windows. In each experiment, we only train the embeddings in the first time window, then infer all source vertices' embeddings dynamically in the following time windows. We present our results in Table 6.3. From the results,

Table 6.3: Evaluation with Dynamic Setting

(a) Time Window = 30m

Method	Acc	Pre	Rec	Fscore	AUC	MAP
GraphSage-Batch	0.581	0.065	0.471	0.114	0.513	0.462
GraphSage-Dynamic	0.517	0.044	0.353	0.077	0.364	0.281
BEA-Batch	**0.693**	**0.106**	**0.588**	**0.180**	**0.609**	**0.521**
BEA-Dynamic	0.666	0.082	0.471	0.139	0.602	0.475

(b) Time Window = 1h

Method	Acc	Pre	Rec	Fscore	AUC	MAP
GraphSage-Batch	0.610	0.123	0.563	0.202	0.480	0.425
GraphSage-Dynamic	0.632	0.141	0.625	0.230	0.497	0.383
BEA-Batch	**0.681**	**0.182**	**0.750**	**0.293**	**0.728**	**0.532**
BEA-Dynamic	0.670	0.177	0.750	0.286	0.700	0.436

(c) Time Window = 2h

Method	Acc	Pre	Rec	Fscore	AUC	MAP
GraphSage-Batch	0.552	0.127	0.636	0.212	0.782	0.619
GraphSage-Dynamic	0.560	0.143	0.727	0.239	0.651	0.403
BEA-Batch	0.560	0.155	0.818	0.261	**0.921**	**0.682**
BEA-Dynamic	**0.603**	**0.193**	**1.000**	**0.324**	0.876	0.508

we observe that the performance of anomaly detection based on inferred embeddings, has only small differences compared to the ones learned in static manner. For 2-hour time window, the anomaly detection on inferred embedding achieves even higher precision and recall. It suggests that performing dynamic embedding based on an appropriate time window size may provide even better anomaly detection capability as it removes the noise generated from training within each individual time window. From this experiment, we show that our model is capable of achieving consistent performance in a dynamic environment.

6.6 Conclusion

In this chapter, we propose an unsupervised anomaly detection framework for dynamic bipartite graph with burstiness, called BEA. In this framework, we map the two sets of vertices on a bipartite graph as source and target vertices according to their characteristics. One of the key concepts of BEA is to detect anomalies based on both burstiness and connectivity patterns. Extensive experiments on both open datasets and the study on the PayPal CSP log anomaly detection application demonstrates that BEA is effective and practical. It suggests a promising direction to detect anomalies with dynamic bipartite graph data.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have studied three important risk management applications on PayPal networks, including seller community detection, risky seller detection, and cyber anomaly detection. Due to the nature of network data, we cast each of the applications into a specific graph mining task, *i.e.,*community detection on attributed graph, vertex classification with graph convolution, anomaly detection on graph with burstiness. The main contributions of the dissertation are the novel frameworks we proposed and analysis for each of the applications. We evaluate our frameworks on both open datasets and PayPal networks to demonstrate their effectiveness and practicality. The summarisations of the contributions for each of the application are as follows:

In Chapters 3 and 4, we present AGGMMR framework to detect seller communities on PayPal payment network. It addresses the challenges come from scalability, complex data type, incomplete data, and multiple information sources. AGGMMR detects community based on both seller connectivities and business attributes. We propose a weight learning step to automatically adjust the importance between attributes and connectivities and a modularity refine step to refine the community memberships assigned by greedy modularity maximisation. Moreover, as PayPal's payment network

keeps evolving, we have extended our AGGMMR algorithm to continuously detect seller community on dynamic graph. The sellers' communities membership is able to be updated whenever the payment network grows.

In Chapter 5, we study a risky seller detection application on payment network. The objective is to classify risky seller based on both transaction connectivities and business topologies. We propose an intuitive graph convolution network DP-GCN for the detection. It has a dual-path structure to model both transaction connectivities and business topologies with graph convolution process. The two sides of information are modeled in an interactive way and finally combined by a multi-head attention module.

In Chapter 6, we research on an anomaly detection application on PayPal CSP log cyber network. The objective is to detect external domains that have malicious activities. Due to the characteristic of the anomaly on the CSP log network, we proposed an anomaly detection framework BEA. BEA detects anomalies based on traffic connectivity patterns and burstiness from both external domains and internal servers.

Our proposed techniques such as role-based convolution, dual-path convolution can be easily extended to model other/multiple network graph properties for network applications with corresponding needs.

7.2 Limitations and Future Work

There are still many issues in our proposed models that can be explored in the future. In this section, we discuss the limitations in this dissertation and outline directions for future works.

In Chapters 3 and 4, we study the community detection application on PayPal payment network and proposed frameworks AGGMMR and Inc-AGGMMR. These two frameworks are based on the traditional modularity maximization approach, which requires loading the whole graph into memory for processing. It potentially limits the framework been used when there

is no sufficient memory space. Recently, there are many works trying to optimize the modularity with deep learning models [31, 137]. Deep learning models allow the learning phase on large graph been distributed into batches, thus greatly address the challenges from memory limitation. Hence, in future work, we can transform the logic of AGGMMR and Inc-AGGMMR into a deep learning framework. This allows our framework to benefit from sampling or batch learning, thus is easier to be applied when there is limited memory space. Another limitation may come from the Inc-AGGMMR framework when it detects communities on dynamic graph. The current Inc-AGGMMR framework adopts an effective incremental approach to update all vertices' community memberships when there is a new vertex comes. But this approach does not consider the case when vertices have been removed from the network. In many scenarios, we can leave those removed vertices on the network. For example, there are accounts been canceled from the PayPal network, but the historical transactions from those accounts may still help to reveal the community structures. Despite the usefulness of those removed vertices for community detection tasks, we can still design a process to update vertices community memberships in the case when those vertices have been removed from the network. It potentially enables our framework been adapted to more scenarios.

In Chapter 5, we study the risky seller detection application on PayPal network. We propose role-based convolution to model vertex topology information. The role-based convolution preserves the convolution path for all vertices that are similar in topology while significantly reduce the computation time of the convolutions. However, it does not support multi-hop convolution for topology roles. Multi-hop convolution allows the convolution been performed on different levels, thus adjust the distance a vertex signal could be propagated. In the current design, we use a sharing scheme to update the embeddings on member vertices from their connected topology roles. This scheme requires the member vertices from different topology roles to be

non-overlapped. In many use cases, a vertex could belong to multiple latent topology roles. In future work, we could use an additional aggregation layer instead of using the current sharing scheme to allow the member vertices' embeddings been updated from multiple topology roles. With this design, the vertices signals are allowed to be propagated between different topology roles, thus enables the convolution of topology roles through multi-hop neighbours.

In Chapter 6, we research on the cyber anomaly detection on PayPal CSP log network. We propose framework BEA to detect anomalies from both network traffic connectivity patterns and domain/server burstiness. Due to the characteristic of CSP log anomaly, we designed bipartite graph convolution to model the connectivity pattern. For other graph anomaly detection applications, the sequence of connectivity may also useful to detect the corresponding anomaly. The sequence of connectivity captures the order of interactions, thus can be used to differentiate specific behaviours. In future work, we can extend the current pattern embedding module to model the sequence information, making it to be applicable to other anomaly detection tasks.

Finally, more theoretical analysis can be conducted for each of the proposed frameworks. Theoretical guarantees ensure our frameworks achieve consistent and interpretable performance on risk management applications.

Publications

Accepted

1. **Zhe Chen**, Aixin Sun, Xiaokui Xiao, "Community detection on large complex attribute network", In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2041-2049), 2019.

2. **Zhe Chen**, Aixin Sun, "Anomaly detection on dynamic bipartite graph with burstiness", In *2020 IEEE International Conference on Data Mining (ICDM)* (pp. 966-971). IEEE, 2020.

3. **Zhe Chen**, Aixin Sun, Xiaokui Xiao, "Incremental Community Detection on Large Complex Attributed Network", *Transactions on Knowledge Discovery from Data (TKDD), 15(6), 1-20*, 2021.

Under Review

1. **Zhe Chen**, Aixin Sun, "DP-GCN: Node Classification Based on Both Connectivity and Topology Structure Convolutions", Submitted to *WSDM 2022*.

References

[1] "Content Security Policy: A Rising Step Towards CyberSecurity." [Online]. Available: https://www.3esofttech.com/content-security-policy-step-towards-cybersecurity

[2] "MovieLens 100k network dataset – KONECT," April 2017. [Online]. Available: http://konect.uni-koblenz.de/networks/movielens-100k_rating

[3] "UC Irvine forum network dataset – KONECT," April 2017. [Online]. Available: http://konect.uni-koblenz.de/networks/opsahl-ucforum

[4] "Unicode languages network dataset – KONECT," Apr. 2017. [Online]. Available: http://konect.uni-koblenz.de/networks/unicodelang

[5] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "Outlier detection in graph streams," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 399–409.

[6] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[7] N. K. Ahmed, R. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry, "Learning role-based graph embeddings," *arXiv preprint arXiv:1802.02896*, 2018.

[8] N. K. Ahmed, R. A. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry, "role2vec: Role-based network embeddings," in *Proc. DLG KDD*, 2019.

[9] L. M. Aiello, C. Cherifi, H. Cherifi, R. Lambiotte, P. Lió, and L. M. Rocha, *Complex Networks and Their Applications VII: Volume 1 Proceedings The 7th International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2018*. Springer, 2018, vol. 812.

[10] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[11] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, "Dynamic self-organizing maps with controlled growth for knowledge discovery," *IEEE Transactions on neural networks*, vol. 11, no. 3, pp. 601–614, 2000.

[12] A. Aljubairy, M. Zaib, Q. Z. Sheng, W. E. Zhang, N. H. Tran, K. L. Nguyen *et al.*, "Hetegraph: a convolutional framework for graph learning in recommender systems," in *2020 International Joint Conference on Neural Networks, IJCNN 2020*. Institute of Electrical and Electronics Engineers (IEEE), 2020.

[13] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[14] N. Barbieri, F. Bonchi, and G. Manco, "Influence-based network-oblivious community detection," in *ICDM*, 2013, pp. 955–960.

[15] V. Batagelj, A. Mrvar, and M. Zaveršnik, "Partitioning approach to visualization of large graphs," in *International Symposium on Graph Drawing*. Springer, 1999, pp. 90–97.

[16] Y. Bian, J. Ni, W. Cheng, and X. Zhang, "Many heads are better than one: Local community detection by the multi-walker chain," in *ICDM*, 2017, pp. 21–30.

[17] P. Bindu, R. Mishra, and P. S. Thilagam, "Discovering spammer communities in twitter," *Journal of Intelligent Information Systems*, vol. 51, no. 3, pp. 503–527, 2018.

[18] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. of statistical mechanics: theory and experiment*, vol. 2008, no. 10, 2008.

[19] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[20] X. Cai, F. Nie, and H. Huang, "Multi-view k-means clustering on big data," in *IJCAI*, 2013, pp. 2598–2604.

[21] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 377–386.

[22] S. Chakraborty and N. Nagwani, "Analysis and study of incremental k-means clustering algorithm," in *International Conference on High Performance Architecture and Grid Computing*, 2011, pp. 338–341.

[23] A. Chaudhari and P. Mulay, "A bibliometric survey on incremental clustering algorithm for electricity smart meter data analysis," *Iran Journal of Computer Science*, vol. 2, p. 197–206, 2019.

[24] W. Cheng, C. Greaves, and M. Warren, "From n-gram to skipgram to concgram," *International journal of corpus linguistics*, vol. 11, no. 4, pp. 411–433, 2006.

[25] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.

[26] T. Choeikiwong and P. Vateekul, "Improve accuracy of defect severity categorization using semi-supervised approach on imbalanced data sets," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2016.

[27] T. Dang and E. Viennet, "Community detection based on structural and attribute similarities," in *Int'l conf. on digital society*, 2012, pp. 7–12.

[28] Deloitte, "The future of digital payments," 2019. [Online]. Available: https://www2.deloitte.com/sg/en/pages/financial-services/articles/future-of-digital-payments.html

[29] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[30] H. Dev, "A user interaction based community detection algorithm for online social networks," in *SIGMOD*. ACM, 2014, pp. 1607–1608.

[31] J. Ding, Y. Sun, P. Tan, and Y. Ning, "Modularity maximization for community detection in networks using competitive hopfield neural network," *International Journal of Innovative Computing, Information and Control*, vol. 15, no. 4, pp. 1455–1467, 2019.

[32] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (anti) social communication: characterization and detection," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 886–894.

[33] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1320–1329.

[34] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 424–432.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, no. 34, 1996, pp. 226–231.

[36] I. Falih, N. Grozavu, R. Kanawati, and Y. Bennani, "Community detection in attributed network," in *WWW*, 2018, pp. 1299–1306.

[37] R. Francis, "Payments in the digital era: Who bears the risks of high-speed transactions?" 2020. [Online]. Available: https://www.jdsupra.com/legalnews/payments-in-the-digital-era-who-bears-93521/

[38] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416–1424.

[39] Y. Gao, C. Zhang, J. Peng, and A. Parameswaran, "The importance of norm regularization in linear graph embedding: Theoretical analysis and empirical demonstration," *arXiv preprint arXiv:1802.03560*, 2018.

[40] A. Geyer-Schulz and M. Ovelgönne, "The randomized greedy modularity clustering algorithm and the core groups graph clustering scheme," in *German-Japanese Interchange of Data Analysis Results*. Springer, 2014, pp. 17–36.

[41] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[42] P. Giudici, "Fintech risk management: A research challenge for artificial intelligence in finance," *Frontiers in Artificial Intelligence*, vol. 1, p. 1, 2018.

[43] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2016, pp. 855–864.

[44] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *ACM Sigmod record*, vol. 27, no. 2, pp. 73–84, 1998.

[45] ——, "Rock: A robust clustering algorithm for categorical attributes," *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[46] N. Gupta and R. Ujjwal, "An efficient incremental clustering algorithm," *World Comput. Sci. Inf. Technol. J*, vol. 3, no. 5, pp. 97–99, 2013.

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

[48] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[49] N. A. Heard, D. J. Weston, K. Platanioti, D. J. Hand *et al.*, "Bayesian anomaly detection methods for social networks," *The Annals of Applied Statistics*, vol. 4, no. 2, pp. 645–662, 2010.

[50] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1231–1239.

[51] Q. Huang, J. Wei, Y. Cai, C. Zheng, J. Chen, H.-f. Leung, and Q. Li, "Aligned dual channel graph convolutional network for visual question answering," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7166–7176.

[52] Y. Huang and H. Wang, "Consensus and multiplex approach for community detection in attributed networks," in *IEEE GlobalSIP*, 2016, pp. 425–429.

[53] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *PAKDD*, 1997, pp. 21–34.

[54] ——, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *DMKD*, vol. 2, no. 3, pp. 283–304, 1998.

[55] C. Jia, Y. Li, M. B. Carson, X. Wang, and J. Yu, "Node attribute-enhanced community detection in complex networks," *Scientific Reports*, vol. 7, no. 1, p. 2626, 2017.

[56] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Catchsync: catching synchronized behavior in large directed graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 941–950.

[57] S. Kanai, Y. Fujiwara, Y. Yamanaka, and S. Adachi, "Sigsoftmax: Reanalysis of the softmax bottleneck," in *Advances in Neural Information Processing Systems*, 2018, pp. 286–296.

[58] S. Karlin, J. McGregor *et al.*, "Many server queueing processes with poisson input and exponential service times." *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 87–118, 1958.

[59] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[60] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

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

[62] J. Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[63] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[64] W. Kudo, M. Nishiguchi, and F. Toriumi, "Gcnext: graph convolutional network with expanded balance theory for fraudulent user detection," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–12, 2020.

[65] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.

[66] C. Li, A. Sun, and A. Datta, "Twevent: segment-based event detection from tweets," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 155–164.

[67] H. Li, G. Fei, S. Wang, B. Liu, W. Shao, A. Mukherjee, and J. Shao, "Bimodal distribution and co-bursting in review spam detection," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1063–1072.

[68] J. Li, Z. Tu, B. Yang, M. R. Lyu, and T. Zhang, "Multi-head attention with disagreement regularization," *arXiv preprint arXiv:1810.10183*, 2018.

[69] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *AAAI*, 2018.

[70] S. Lim, J. Kim, and J.-G. Lee, "Blackhole: Robust community detection inspired by graph drawing," in *ICDE*, 2016, pp. 25–36.

[71] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[72] ——, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[73] L. Liu, L. Xu, Z. Wangy, and E. C., "Community detection based on structure and content: a content propagation perspective," in *ICDM*, 2015, pp. 271–280.

[74] X. Liu, Q. Yang, and L. He, "A novel dbscan with entropy and probability for mixed data," *Cluster Computing*, vol. 20, no. 2, pp. 1313–1323, 2017.

[75] Y. Liu, Y. Ouyang, and Z. Xiong, "Incremental clustering using information bottleneck theory," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 05, pp. 695–712, 2011.

[76] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 2077–2085.

[77] L. Lv, J. Cheng, N. Peng, M. Fan, D. Zhao, and J. Zhang, "Auto-encoder based graph convolutional networks for online financial anti-fraud," in *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE, 2019, pp. 1–6.

[78] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, "Riwalk: fast structural node embedding via role identification," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 478–487.

[79] S. Maekawa, K. Takeuch, and M. Onizuka, "Non-linear attributed graph clustering by symmetric nmf with pu learning," *arXiv preprint arXiv:1810.00946*, 2018.

[80] S. Maekawa, J. Zhang, G. Fletcher, and M. Onizuka, "General generator for attributed graphs with community structure."

[81] ——, "General generator for attributed graphs with community structure," in *proceeding of the ECML/PKDD Graph Embedding and Mining Workshop*, 2019, pp. 1–5.

[82] A. Mahmood and M. Small, "Subspace based network community detection using sparse linear coding," in *ICDE*, 2016, pp. 1502–1503.

[83] S. T. Mai, I. Assent, and M. Storgaard, "Anydbc: An efficient anytime density-based clustering algorithm for very large complex datasets," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1025–1034.

[84] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[85] H. Manjunatha and R. Mohanasundaram, "Brnads: Big data real-time node anomaly detection in social networks," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2018, pp. 929–932.

[86] R. Márquez, "Overlapping community detection in static and dynamic networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 925–926.

[87] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos, "Snare: a link analytic system for graph labeling and risk detection," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1265–1274.

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

[89] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[90] M. E. Newman, "Detecting community structure in networks," *The European Physics Journal B*, vol. 38, no. 2, pp. 321–330, 2004.

[91] ——, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.

[92] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[93] N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, "Dynamic social community detection and its applications," *PloS one*, vol. 9, no. 4, 2014.

[94] T. Opsahl and P. Panzarasa, "Triadic closure in two-mode networks: Redefining the global and local clustering coefficients," *Social Networks*, vol. 34, 2011.

[95] G. Pan, W. Zhang, Z. Wu, and S. Li, "Online community detection for large complex networks," *PloS one*, vol. 9, no. 7, p. e102799, 2014.

[96] H. Park and J. Neville, "Exploiting interaction links for node classification with deep graph neural networks." in *IJCAI*, 2019, pp. 3223–3230.

[97] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[98] D. T. Pham, S. S. Dimov, and C. Nguyen, "An incremental k-means algorithm," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 218, no. 7, pp. 783–795, 2004.

[99] J. M. Phillips and S. Venkatasubramanian, "A gentle introduction to the kernel distance," *arXiv preprint arXiv:1103.1625*, 2011.

[100] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo, "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," *Decision Support Systems*, vol. 133, p. 113303, 2020.

[101] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *WWW*, 2014, pp. 225–236.

[102] T. Puschmann, "Fintech," *Business & Information Systems Engineering*, vol. 59, no. 1, pp. 69–76, 2017.

[103] M. J. Rattigan, M. Maier, and D. Jensen, "Graph clustering with network structure indices," in *ICML*, 2007.

[104] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 385–394.

[105] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: http://networkrepository.com

[106] R. A. Rossi, N. K. Ahmed, E. Koh, S. Kim, A. Rao, and Y. A. Yadkori, "Hone: higher-order network embeddings," *arXiv preprint arXiv:1801.09303*, 2018.

[107] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive network representation learning," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 953–960.

[108] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *WWW*, 2013, pp. 1089–1098.

[109] M. Salehi and L. Rashidi, "A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]," *SIGKDD Explor. Newsl.*, vol. 20, no. 1, pp. 13–23, May 2018. [Online]. Available: http://doi.acm.org/10.1145/3229329.3229332

[110] N. F. Samatova, W. Hendrix, J. Jenkins, K. Padmanabhan, and A. Chakraborty, *Practical graph mining with R.* CRC Press, 2013.

[111] P. I. Sánchez, E. Müller, U. L. Korn, K. Böhm, A. Kappes, T. Hartmann, and D. Wagner, "Efficient algorithms for a robust modularity-driven clustering of attributed graphs," in *SDM*, 2015, pp. 100–108.

[112] 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.* Springer, 2018, pp. 593–607.

[113] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[114] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.

[115] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *KDD*, 2015, pp. 1075–1084.

[116] O. Shchur, A. Bojchevski, M. Farghal, S. Günnemann, and Y. Saber, "Anomaly detection in car-booking graphs," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW).* IEEE, 2018, pp. 604–607.

[117] X.-B. Shen, W. Liu, I. W. Tsang, F. Shen, and Q.-S. Sun, "Compressed k-means for large-scale clustering." in *AAAI*, 2017, pp. 2527–2533.

[118] H. Shiokawa, Y. Fujiwara, and M. Onizuka, "Fast algorithm for modularity-based graph clustering," in *AAAI*, 2013, pp. 1170–1176.

[119] P. L. Szczepanski, A. S. Barcz, T. P. Michalak, and T. Rahwan, "The game-theoretic interaction index on social networks with applications to link prediction and community detection," in *IJCAI*, 2015, pp. 638–644.

[120] W.-S. Tai and C.-C. Hsu, "Growing self-organizing map with cross insert for mixed-type data clustering," *Applied Soft Computing*, vol. 12, no. 9, pp. 2856–2866, 2012.

[121] P. Todorovic and D. Woolhiser, "On the time when the extreme flood occurs," *Water Resources Research*, vol. 8, no. 6, pp. 1433–1438, 1972.

[122] Q.-A. Tran, H. Duan, and X. Li, "One-class support vector machine for anomaly network traffic detection," *China Education and Research Network (CERNET), Tsinghua University, Main Building*, vol. 310, 2004.

[123] 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*, 2017, pp. 5998–6008.

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

[125] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[126] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[127] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 2017, pp. 327–335.

[128] X. Wang, J. Song, K. Lu, and X. Wang, "Community detection in attributed networks based on heterogeneous vertex interactions," *Applied Intelligence*, vol. 47, no. 4, pp. 1270–1281, 2017.

[129] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1243–1253.

[130] Z. Wang, S. Gu, X. Zhao, and X. Xu, "Graph-based review spammer group detection," *Knowledge and Information Systems*, vol. 55, no. 3, pp. 571–597, 2018.

[131] H. Weng, Z. Li, S. Ji, C. Chu, H. Lu, T. Du, and Q. He, "Online e-commerce fraud: A large-scale detection and analysis," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1435–1440.

[132] H. Wu, M. Zhou, and J. Gong, "Investigation on the ip flow inter-arrival time in large-scale network," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2007, pp. 1925–1928.

[133] J. Wu, J. He, and J. Xu, "Net: Degree-specific graph neural networks for node and graph classification," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 406–415.

[134] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[135] Y. Yamaguchi and K. Hayashi, "When does label propagation fail? a view from a network generative model," in *IJCAI*, 2017, pp. 3224–3230.

[136] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *ICDM*, 2013, pp. 1151–1156.

[137] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning." in *IJCAI*, vol. 16, 2016, pp. 2252–2258.

[138] Z. Yang, T. Hao, O. Dikmen, X. Chen, and E. Oja, "Clustering by nonnegative matrix factorization using graph random walk," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1079–1087, 2012.

[139] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, 2018, pp. 4800–4810.

[140] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 647–657.

[141] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2672–2681.

[142] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[143] N. Zarayeneh and A. Kalyanaraman, "A fast and efficient incremental approach toward dynamic community detection," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 9–16.

[144] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[145] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[146] Y. Zhao, X. Wang, H. Yang, L. Song, and J. Tang, "Large scale evolving graphs with burst detection," in *28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[147] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn."

[148] Z. Zheng, X. Wu, and R. Srihari, "Feature selection for text categorization on imbalanced data," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 80–89, 2004.

[149] Z. Zheng, L. Zheng, M. Garrett, Y. Yang, M. Xu, and Y.-D. Shen, "Dual-path convolutional image-text embeddings with instance loss," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2, pp. 1–23, 2020.

[150] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *PVLDB*, vol. 2, no. 1, pp. 718–729, 2009.

[151] Q. Zhu, Z. He, T. Zhang, and W. Cui, "Improving classification performance of softmax loss function based on scalable batch-normalization," *Applied Sciences*, vol. 10, no. 8, p. 2950, 2020.

[152] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Tech. Rep. CMU-CALD-02-107, 2002.

[153] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 336–345.

[154] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 499–508.

[155] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.