Utrecht University

Synthetic Financial Transaction Data Generation: a Graph-Based Approach

Supervisors:
Internal:
Dr. I.R. Karnstedt-Hulpus
Dr. E.J. van Leeuwen
External:
T. Pelle

Author:
Karen Schutte
6403514

MSc Artificial Intelligence

Master's Thesis

Date:
July, 2024

Abstract

This thesis presents a novel approach to generating synthetic transaction networks. The research focuses on developing a graph-based generative model capable of replicating characteristics observed in real-world financial networks. The motivation of this model is to preserve data privacy, and it generates networks that exhibit power-law degree distributions, no assortativity or disassortativity, exponential weight distributions, and community structures similar to those found in actual financial transaction data.

The methodology involves a clustering analysis of a real transaction dataset to identify node-types, which are then integrated into the generative model. Parameters for node generation, edge densification, and a probability matrix governing type-based connections are established to control the network's structural properties. The model is validated against this real network dataset from Rabobank, by comparing the metrics and structural properties.

Experimental results show that the model can produce stable synthetic networks over 200,000 iterations, with generated networks exhibiting comparable degree distributions, edge densities, and community structures to the real dataset. However, limitations include the use of a sampled and aggregated dataset for validation, which restricts the model's ability to capture the full complexity of real financial networks, and the model's exponential weight distribution diverging from the real dataset's power-law weight distribution.

This research contributes a publicly available tool, which can be used as a starting point for generating synthetic financial transaction networks, facilitating applications in machine learning model training for detecting criminal financial activity. Future research directions include improving weight distribution modeling, exploring algorithms for power-law distributions, and extending the model to include interbank networks and temporal dynamics.

Contents

1 Introduction

AI is expected to grow exponentially in the upcoming years [32], and is applied in many fields such as healthcare, social media and finance. It is known that AI models are data hungry [1]. They require large sample datasets to be trained on, which ideally leads to improved decision-making. There is no predetermined required sample size, but many researchers follow the Widrow-Hoff learning rule [34] which states that for every parameter at least ten datapoints are needed [12]. More datapoints will result in more generalized predictions, and less overfitting. In most cases, data is publicly available and can be downloaded online. However, in some domains, privacy concerns arise, and data cannot be shared publicly. This is the case with financial data, which banks are not allowed to share due to privacy laws. Still, it is crucial to investigate this transactional data to generate a model that can detect patterns of criminal financial activity such as fraud, terrorism financing or money laundering.

A solution to this problem could be to generate synthetic data, which is data that is generated using some properties of the real data, such that privacy is not invaded. Synthetic data can be widely used, e.g. to train models to detect previously mentioned criminal patterns, or to solve the problem of data imbalance in transaction datasets, as only a small percentage is fraudulent [49]. Also, investigating synthetic data generation might be interesting for other domains that are dealing with private data.

Transaction data can be effectively modelled as a network [29] where the nodes represent financial entities, and the edges indicate the occurrence of transactions between them. The weights on these edges can represent the transaction volume, frequency, or some other transaction characteristic. These edges can be directed to indicate the direction in which the money was transferred. Transaction networks can be dynamic to match real-world transaction networks. These type of networks can be effectively used for detecting criminal behavior [28], which is why generating synthetic transaction networks is a promising research area.

Current approaches of generating synthetic data are based on graph generative models [51], intelligent agents [55, 4], machine learning models [11, 42], or other techniques [22, 40, 21]. However, some of them do not take network structures into account [30], are not reproducible [51], or only provide the datasets, and not the generators of the dataset [4]. Others are based on data that is not publicly available or require domain knowledge [40, 21, 22], making validation a hard task. Further, models that are based on real data can be susceptible to attackers, as they will try to acquire the real data from the generated data [17].

Graph generative models are a promising research avenue for generating synthetic data, due to their stability [4] and privacy preserving nature, as they do not require real data directly during generation. Furthermore, using graph generative models for synthetic transaction data might provide interesting insights into the evolution of transaction networks, if the results align with real data. However, most graph generative models suffer from limitations: they typically produce unweighted and undirected networks, are domain-agnostic, and do not account for nodes with different behaviors or types. While some models use a notion of 'fitness' [9] to distinguish between nodes, or divide the network evolution in two classes [8], no models offer clear, controllable types during network evolution. Further, in financial data, scale-free properties are observed for degree and weight distributions, they are neither assortative, nor disassortative, show bow-tie structures and community structures [47]. No graph generative models are known that can generate all of these properties, as most only generate scale-free degree distributions, and/or community structures.

In this research, we provide a stable graph generative model that preserves privacy and can generate weighted, directed networks with distinct node-types. The networks show negative to slightly positive assortativity, scale-free degree distributions and a bow-tie structure similar to a real transaction network. The generator is publicly available, and is validated on real (aggregated) transaction data that is publicly available, making this research reproducible.

Background information will be provided on network metrics for weighted directed networks, as well as an overview of research conducted in the fields of synthetic data and financial data. In Section 4, the model is introduced. This section will also focus on analyzing the degree distribution, mixing behavior, and weight distribution of the graphs generated by the model. In Section 5, simulations will be conducted to investigate the effect of the different parameters on the metric values. Section 6 will describe a use case where our model is applied to real data. Finally, in Sections 7 and 8, the model is concluded and discussed by giving future research directions and limitations.

1.1 Problem statement

We aim to create a model of network generation and growth that can generate and evolve graphs based on a predetermined set of large-scale properties. These properties include degree distribution, clustering coefficient, assortativity, and density. Our aim is to create a model that can be calibrated to cover a broad range of the targeted values, within the limits of the broad network characteristics that are most frequent in real-world networks. We aim to replicate features such as power-law degree distributions and community structures.

One of the application domains we target is the financial domain. In financial networks, the edge weights as well as the edge directions are critical. We therefore put particular emphasis on reproducing the in-degree and out-degree distributions, the bow-tie structure of directed networks and the edge-weight distribution. These characteristics are essential for understanding the flow of transactions and identifying patterns within financial data.

Furthermore, an important aspect of financial networks is the consistent behavior of many nodes over time, which exhibit clear patterns of connections. For example, an account of a subscription app regularly receives numerous incoming transactions of low amounts. With this intuition, we aim for our model to also be able to reproduce networks with multiple types of nodes and with various node-type interaction patterns. This capability is crucial for capturing the diversity of real-world financial networks and ensuring that the synthetic data generated is realistic and useful for further analysis.

Summarizing, our research aims to develop a stable and flexible graph generative model that can match a wide range of network properties seen in real-world scenarios. Graphs are generated and evolved based on a predetermined set of large-scale properties. We focus on the financial domain, intending to create synthetic networks that keep data private and mimic the structure and behavior of real financial transaction networks. We do this by using multiple types of nodes with various node-type interaction patterns, to ensure that the synthetic data is realistic.

1.2 Contributions

- This research provides a publicly available synthetic graph generator which can be used to generate synthetic financial networks, given a set of input parameters.
- The model produces stable, weighted, directed networks, whose weights represent the counts of transactions between two nodes, and other weights can be added to represent the volume of transactions by adding an extra weight parameter.
- This model is validated on publicly available data, making the research reproducible.
- In the generated graphs we measured clustering coefficients ranging from 0.0098 to 0.42, assortativity coefficients ranging from -0.64 to 0.086 and density ranging from 0.0002 to 0.023 by adjusting the parameters. However, these metrics are interdependent. More specific: in almost all parameter combinations that result in an increased clustering, density increases as well, but assortativity decreases.
- The model is capable of accurately replicating the bow-tie structure of a given network.
- The model produces stable graphs.
- The model might be used to generate other types of networks by using different parameter settings.

2 Network metrics in weighted directed networks

To generate synthetic networks it is necessary to understand the structure and characteristics of these networks. In transaction networks, the nodes usually represent accounts of customers, companies ATMs, or other financial entities. The edges indicate whether there exist transactions between them. These networks are often weighted and directed, where the weights represent the amount or volume of the transaction, and the direction corresponds to the direction in which the money is transferred, as explained in Section 3. It is suitable to represent financial networks as weighted, directed networks [29], which is why an overview of metrics is provided to analyze networks which can also be used for weighted, directed networks. Some of the given metrics are only suitable for unweighted networks, in the analysis we then only use the unweighted, undirected version of the generated network.

We define a graph as a pair $G = (V, E)$, where V is the set of vertices or nodes, and E the set of edges. For node i, we define the degree of the node as k_i and the set of neighbors as V_i. The weight between nodes i and j is defined as w_{ij}. Lastly, a_{ij} represents the elements of the adjacency matrix A, where, for unweighted networks, each element takes the value 1 if there is an edge between i and j, and 0 otherwise. In weighted networks, these elements can have different values to represent the weights. If the network is undirected, this matrix will be symmetric. If it is directed, the matrix will be asymmetric. Most networks do not contain self-loops, meaning a_{ii} will always be 0, resulting in a zero-diagonal matrix.

2.1 Strength of a node

For the strength of a node i we calculate the in-strength $s_{in}(i)$, out-strength $s_{out}(i)$ and add these together to obtain the strength $s(i)$. The in- and out-strength are calculated by calculating the total incoming and outgoing weights of a node.

2.2 Clustering coefficient

The clustering coefficient is a measure of cohesion in a network. It measures 'how tightly knit the neighbors are' [46]. There are several ways to measure the clustering coefficient. In unweighted networks, the local clustering coefficient is defined as:

$$CC(i) = \frac{1}{k_i \cdot (k_i - 1)} \sum_{j,h} a_{ij} a_{ih} a_{jh}$$

This clustering coefficient is calculated for every node, and then averaged to find the average clustering coefficient.

Onnela et al. [39] also proposed a clustering coefficient measure. This measure was based on a concept called *subgraph intensity*, which is defined as the geometric average of the subgraph edge weights [45].

$$CC(i)_O = \frac{1}{k_i \cdot (k_i - 1)} \sum_{j,h} \hat{w}_{ij} \hat{w}_{ik} \hat{w}_{jk}$$

Here, the edge weights are normalized by the maximum weight in the network, $\hat{w}_{ij} = w_{ij} / \max_{a,b}(w_{ab})$. This definition illustrates the proportion of triangle weights in relation to the maximum value within the network.

More definitions of clustering coefficients exist; for a more detailed overview we refer to the paper of Saramaki [45]. Our research will utilize the unweighted clustering coefficient proposed by Onnela et al. [39] and the unweighted version, as these are the most commonly used metrics for (un)weighted networks.

2.3 Assortativity

Assortativity is the tendency of a vertex to connect with another vertex based on their similarity [44]. This similarity could be in vertex degree, weight or some other characteristic. The assortativity coefficient was originally proposed for undirected, unweighted networks [38], but can also be used for directed [36] and weighted [25] networks. In this research we limit ourselves to undirected degree and weight assortativity, as these are the most commonly used metrics in literature.

Calculating the assortativity coefficient can be done by calculating the Pearson correlation coefficient, or the Spearman correlation coefficient [16] of the degrees of two adjacent nodes. In this research, the definition of Newman et al. [37] is used, which uses matrix e_{ij}, which is the fraction of edges in a network that connect a vertex of type i to one of type j, to define assortativity. In the definition, a_x and b_y are the fractions of edges that start at a node with x and end at a node with value y. This assortativity is then calculated by the Pearson correlation coefficient:

$$r = \frac{\sum_{xy} xy(e_{xy} - a_x b_y)}{\sigma_a \sigma_b}$$

This definition is used, because it is the one used in the NetworkX package we use.

2.4 Degree distribution

Each node has a degree. The degree distribution represents the probability distribution of these degrees across the entire network. In weighted, directed networks, we can look at the degree distribution by looking at the in- and out-degree distribution. For unweighted, undirected networks, Barabasi and Albert found that many networks follow a power-law degree distribution [7]. The power-law can be defined as $(P(k = x) \sim k^{-\gamma})$, meaning the probability that a node has degree k follows the given probability.

2.5 Community structure

A community is a group of nodes that are connected together. In various real-world networks these structures emerge. Finding these communities is an active research topic, and many different algorithms exist [33]. Communities can be hard to detect, because a formal definition is absent. However, an attempt to define it was done by Condon and Karp [15]. In their definition, a network contains communities if the probability that two nodes in a community are connected is higher than the probability that two nodes in different communities are connected.

A measure to detect communities is modularity [14], which compares the difference between the actual number of edges in a community and the expected number of edges in a community. Modularity based algorithms try to maximize this modularity, to identify meaningful groupings of nodes. Modularity is formally defined as:

$$Q = \sum_{c_i \in \mathcal{C}} \left[\frac{|E_{c_i}^{\text{in}}|}{|E|} - \left(\frac{2|E_{c_i}^{\text{in}}| + |E_{c_i}^{\text{out}}|}{2|E|} \right)^2 \right]$$

Where C is the set containing all communities, c_i is a community in this set, $|E_{c_i}^{\text{in}}|$ is the amount of edges in the community, $E_{c_i}^{\text{out}}|$ the number of edges from the community to outside the community, and $|E|$ is the total number of edges in the network.

Two similar algorithms that use modularity for community detection are the Leiden algorithm [52], and the Louvain algorithm [10]. The authors of the Leiden algorithm argue that the Louvain algorithm may return arbitrarily poorly connected communities, and claim to have found a solution to this problem. The Leiden algorithm is more complex than the Louvain algorithm, but produces communities that are ensured to be connected, and is faster than the Louvain algorithm. Both Leiden and Louvain are suitable for weighted, undirected networks. In this research, Louvain is used.

2.6 Bow-tie structure

The bow-tie structure of a network decomposes it into seven groups of nodes, as defined by Yang et al. [56]. These groups are as follows:

1. SCC: The core. Consists of nodes that are strongly connected.
2. IN: Nodes from which the core component S is reachable.

3. OUT: Nodes that are reachable from the core component S.
4. TUBES: Nodes that are reachable from IN and can reach OUT, but are not part of S.
5. INTENDRILS: Nodes that are reachable from IN, but cannot be reached by S, and cannot reach OUT
6. OUTTENDRILS: Nodes that can reach OUT, but cannot be reached by S or IN.
7. OTHER: The remaining nodes that do not fit into the above categories.

The bow-tie structure can be seen in Figure 1.

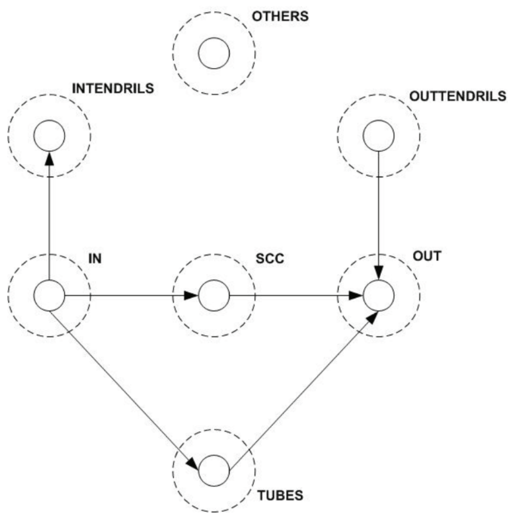


Figure 1: Image of bow-tie structure, taken from [56].

2.7 Summary

The previously described metrics can be used to understand weighted, directed networks, and to compare two or more networks with each other. These metrics will be used to understand the available networks, and to generate networks that are similar to real-world networks. Nevertheless, it is important to note that some metrics are highly correlated, and there is no research on how they hierarchically build upon each other [48].

3 Background

In this section the characteristics that are found in financial networks will be discussed, and an overview of the proposed solutions to generate synthetic networks is provided. According to Lim et al. [29] the process of generating realistic synthetic graphs can be divided into three parts: i) graph generative model selection, ii) synthetic graph generation, and iii) graph generative model validation. Different graph generative models are discussed, but also generative AI models, and finally agent-based modelling, as these are the three most popular processes used for generating graphs.

3.1 Financial Network data

We found four works in the literature that investigate real-world financial network data and discuss them in turn.

Saxena et al. [47] constructed a network from banking transactions of Rabobank users, which they made publicly available. They sampled two directed, weighted networks where the edges contain as weights the aggregated amount of transactions, and the total number of transactions. The networks contain 1.6 million nodes and 3.8 million edges. Their data was collected from 2010 to 2020 and is, to the authors' knowledge, the first of its kind that is publicly shared.

They give an extensive analysis of their networks, showing that it has clear similarities with scale-free networks, but also some differences. The networks both showed a power-law degree, strength and edge-weight distribution, similar to other real-world networks [35]. However, it was shown that there was no correlation between in- vs. out-degree and in- vs. out-strength. The clustering coefficient is analyzed for the unweighted and weighted versions of the graph and showed that when the degree increases, the clustering coefficient decreases. This means that if one entity makes transactions with multiple different entities, these entities are less likely to be connected to each other. The network is neither assortative, nor disassortative, has a community (detected using Leiden algorithm) and hierarchy structure, and has no correlation between weak-ties (connections between communities) and edge-weights. Correlations were measured using Spearman correlation. They mention finally that it is still an open question how transaction networks evolve.

Another financial network was analyzed by Kyriakopoulos et al. [24]. They analyzed a dataset that contains all financial transactions between the accounts of practically all major financial players within Austria over one year. Their dataset contains 423 accounts of financial players, 4.187.943 transactions of a total volume of 11.07 trillion Euro's. The data was aggregated temporally using three scales: yearly, monthly and daily. The distribution of the transaction volume shows an exponential increase for small amounts (between 10^2 and 10^3), and most transactions had a volume of 1000, 10^5 or 1 million.

The authors first assessed the network topology by looking at the unweighted, directed versions and the unweighted, undirected versions of the network. The unweighted, directed networks were obtained by changing all values in the adjacency matrices to 1 if they had a value greater than 0. Next, these matrices were symmetrized to obtain the unweighted, undirected networks.

Power laws were observed in the degree distributions of the unweighted, undirected networks, indicating the presence of hubs. A hub is a node with a neighbor count that substantially exceeds the network's average. This means most accounts only transfer money with a small group of other nodes, and a small number of nodes transfers money with a large part of the network. The authors found strong dependencies between network characteristics on aggregation time, especially for the unweighted networks. Clustering coefficients, which show power law dependencies based on degree, become less pronounced with longer time periods. The authors suggest that hierarchical structures which are noticeable on a daily basis, become invisible over monthly and yearly periods. In contrast, the power law dependencies in the nearest neighbor degree as a function of degree become more evident over longer periods. Further, they mention that degree correlations, indicated by both the Pearson coefficient and nearest neighbor degree versus degree, suggest assortative behavior where high-degree nodes are connected to other high-degree nodes. This is likely influenced by the presence of links with small weights in unweighted networks.

Power-laws were also observed in the strength of the weighted networks, which are less dependent on aggregation. A remarkable thing that the authors found was that the weighted clustering coefficients versus the unweighted ones had similar behavior, while the nearest neighbor degrees differed strongly; the weighted case indicated disassortative behavior for large degrees.

Zhang et al. [57] studied a dataset from Kaggle containing credit card transactions from a European bank over a two-day period in September 2013. They created a weighted support vector machine to detect fraud, and used the Kaggle data for training. There are 284,807 transactions in the dataset, of which 0,17 percent was fraudulent. Most transaction amounts were between 0 and 100 dollars, 79.76% for non-fraudulent and 73.60% for fraudulent transactions. No network metrics were analyzed in this research, because the dataset does not provide node id's, meaning no network structure can be constructed. Still, it is useful to look at the transaction amounts and time distributions in the data.

Kondor et al. [23] conducted an Empirical Analysis of the Bitcoin Transaction Network. Bitcoin, and other cryptocurrency data is interesting to analyze, because transactions are publicly available. The authors created a bitcoin dataset containing the complete list of transactions (on May 7th, 2013) to construct a network containing transaction times and amounts. This data has 17.3 million transactions, 13.1 million addresses (appearing in at least one transaction), of which 1.6 million were active in the last month. Each node in the network represents a Bitcoin address and an edge represents whether there has been at least one transaction between two nodes.

The evolution of the network was studied and two distinct phases of growth were observed. First, the *initial phase*, where the network had large fluctuations and little activity, and second the *trading phase*, where the network did not change significantly anymore and converged to the typical value. The latter phase resembles a real currency more closely, and was characterized by disassortative degree correlations, and powerlaw degree distributions. Also, the growth of the network was driven by linear preferential attachment.

Summarizing, we know that financial networks contain power laws for the degree, strength and edge-weight. Assortativity of financial networks differed in the literature. The paper from Saxena et al. about the Rabobank network mentioned their network was neither assortative nor disassortative, while the paper from Kyriakopoulos et al. with the data from Austria mentioned the unweighted networks were assortative in contrast to the weighted. Finally, the paper from Kondor et al. containing the bitcoin network showed disassortative behavior. The reason for this might be that the used metric for assortativity differed per paper. The bitcoin and Austria paper both used Pearson correlation to measure assortativity, but the Rabobank paper used Spearman correlation. Finally, it is important to mention that these networks have major differences, as the Rabobank network contains aggregated data over a 10-year period, the bitcoin network contains all transactions separately and the Austria network had three different aggregations of their data, daily, monthly and yearly. These aggregations might influence the assortativity, which can explain these differences. Lastly, financial networks show community structures. The bitcoin dataset is the only dataset that shows how the model evolves, because it is the only one that can share this kind of data. The growth was driven by linear preferential attachment.

3.2 Graph Generative Models

In this section we discuss different graph generative models, especially for generating weighted directed networks. We focus on those models whose properties make them suitable for the financial domain. For an extensive survey on weighted network models we refer you to Saxena et al. [46].

The Barabasi-Albert (BA) model [2] is a model proposed by Barabasi and Albert which generates scale-free networks using preferential attachment. A preferential attachment process, also called 'rich get richer', is a process where a quantity is distributed among individuals or objects based on what they already have. In simpler terms, the more someone or something has, the more it gets. In networks this means that new nodes have a higher probability to connect to nodes that have a high degree than to nodes that have a low degree. The BA model generates undirected, unweighted graphs, and power-law degree distributions with $\gamma = 3$. The BA model can be an interesting base model for the financial domain, as we have seen that financial networks have scale-free properties and might be driven by preferential attachment, such as in the Bitcoin network. However, it does not take weights or directions into account.

In 2004, Chakrabarti et al. proposed R-MAT [13], a recursive model for graph mining. They created their model with three goals for the generated graph. The graph should:

1. match the degree distributions (power-law or no power-law)
2. exhibit a community structure
3. have a small diameter

Their model works with an $N \times N$ adjacency matrix A containing only zeroes, which is divided into four quarters, a, b, c, d, which are all assigned a probability. All added together, this probability is equal to 1. The idea of R-MAT is to choose one quarter of the adjacency matrix with the given probabilities, subdivide this quarter again in quarters, and keep choosing until one element a_{ij} is reached. A weight of 1 will be added to this cell, representing a directed, weighted edge from node i to node j. This way, a directed, weighted graph is generated, and the values of a, b, c, d can be tweaked to achieve the desired network characteristics. Additionally, this model can be extended to generate unweighted and undirected networks by restricting the maximum value to be 1 and symmetrizing the matrix, respectively. R-MAT can be promising for modelling transaction networks, as it can generate power-laws and community structures, which are properties of transaction networks. R-MAT is also able to generate weighted directed, networks, which is desirable, as discussed in Section 2.

Other models that can generate power-law distributions are the power-law minimal models, which are simple models that are specifically designed to generate networks with power-law characteristics. These models require minimal effort, making them interesting to investigate.

Antal et al. [5] proposed an evolving model that generalizes BA where the network starts with some seed nodes and grows through a simple strength-driven rule: a new node connects to an existing node with a probability proportional to the strength of that node. The weight of the edge between the nodes is chosen from a given weight distribution, and remains fixed afterwards.

Bianconi [8] divided the evolution of networks in two classes based on the ratio of strength and the rate of making new connections. In, for example, traffic networks, a new edge is added more frequently between two nodes that are not yet connected than increasing the strength between two already connected nodes. This type of network is referred to as class 1. These networks have linear relationships between the strength and the degree of a node. Other networks, such as co-authorship networks, do not have this probability; two existing collaborators tend to publish more papers together than with new collaborators. These are referred to as class 2 networks, and have nonlinear relationships between strength and degree. The network model contains two steps, adding a new node and connecting it using preferential attachment, and choosing already existing edges of which the weight is increased. The parameters of the model can be tweaked in order to get class 1 or class 2 models. We expect that, for transaction networks, class 2 models will be more suitable, as nodes that already have transactions with each other might be more likely to have more transactions with each other than with other or new nodes.

In the power-law minimal models, edges are generated based on the strength of a node. In 2001, Bianconi and Barabasi [9] introduced the notion of fitness for unweighted networks, where each node possesses a fitness value (η), and nodes with higher fitness attract more links, so preferential attachment does not depend anymore on degree only. This phenomenon is also referred to as 'fitter gets richer'.

Zheng et al. [59] generated a stochastic weighted model based on this fitness model. Each node receives a fitness parameter η, which is a random number between 0 and 1. With probability p the edge weight is assigned using weighted scale-free (proportional to the strength of its endpoint), and with probability $(p - 1)$ we use the fitness parameter to assign the weight (proportional to the fitness of its endpoint).

Two types of growth are often used in graph generative models: topological growth, and self-growth i.e. densification [54]. Topological growth means a new node joins the network and connects with existing nodes. Self-growth refers to existing nodes that make new connections, or increase the weight between them.

Wang et al. [54] introduced such an incremental self-growing model for weighted networks. The network starts with n nodes that are fully connected with edge weight w_0. The network has two growth mechanisms: topological growth, following strength preferential attachment, and mutual selection growth, following a given probability function. The total strength of nodes increases uniformly with the network size. Further, the authors demonstrated that the model accurately represents the disassortative nature found in real-world networks.

As mentioned in Section 3, financial networks have community structures. In community-structured networks, two different types of edges exist: intra-community links, when the nodes of the edge are in the same community , and inter-community links, when the nodes of the edge are in a different community.

Li and Chen [26] proposed a model which generates community-structured networks, based on inner- and inter-community preferential attachment, and inner- and inter-community strengthening. The model starts with M seed communities, which have m_0 fully connected nodes. These communities are linked to each other through $M(M-1)/2$ inter-community connections, with each community being connected to $(M-1)$ other communities. For growth there are two options. First, with probability α, a new node is added, and connects to a community using preferential attachment. With probability β, this node makes an inter-community link. All the added edges will have weight 1. Second, with probability $(1-\alpha)$, only an inter-community link will be added. With probability η, also an inter-community link will be added. These added edges will also have weight 1, and will be added using preferential strengthening. The value of β and η can be small when a small amount of inter-community links is desired.

All these graph generative models capture some characteristics that are observed in real transaction networks, such as disassortativity, community structure, and preferential attachment. This makes them interesting to use for generating a transaction network. More generative graph models exist, but these are left out because they cannot capture the characteristics of transaction networks. However, the previously mentioned models are all domain-agnostic, meaning adjustments may need to be done to be able to model transaction networks.

3.3 Generative Adversarial Networks

Another way to generate synthetic networks is by using Generative Adversarial Networks (GANs), or other machine learning (ML) techniques. A GAN is a model proposed by Goodfellow et al. in 2014 [18] which can generate data similar to an input dataset. It uses two neural networks, a generator and a discriminator, which are trained using adversarial training. The generator tries to generate data that is similar to the input data. The discriminator tries to discriminate whether the generated data is real or not. This process continues until the generated data cannot be distinguished anymore from real data. In this section two of these models are covered, which are able to generate unweighted, undirected networks, but can be extended to generate weighted, directed networks. More of these models exist; for an overview of models we refer to the survey of Guo and Zhao [20].

Bojchevski et al. [11] created NetGAN, which they claim to be 'the first implicit generative model for graphs able to mimic real-world networks'. This model learns characteristics of the network by learning the distribution of random walks over the graph. The main goal is to generate a realistic graph, which is a generalization of the input graph. The model is claimed to be an implicit model and to be able to consistently capture all the important graph characteristics. However, NetGAN is computationally expensive, which limits the maximum number of nodes that can be generated; generating more than a few thousand nodes gets time-consuming [31].

The CELL model [42] is inspired by NetGAN, but computationally more efficient. The authors found that the generalizability of the NetGAN model does not lie in the GAN architecture, making it possible to remove this part of the model, and making it less complex. They generate a highly simplified version, which is still able to obtain similar results as NetGAN, but much faster, and easier to adapt.

In a thesis written by Erik Lundin [31], the CELL model and NetGAN model are evaluated. Both of the models are created to only generate undirected and unweighted networks. In the thesis the CELL model is chosen to be extended to be compatible with weighted and directed networks.

As Lundin mentions, GANs and other generative models that use machine learning techniques can be useful for generating synthetic data, and evaluating the used model. In for example image generation, these are very convenient, as humans can tell whether an image looks natural.

However, in graphs this is not the case, making it a hard task to evaluate a model. If a model is built to mimic the input graph, and tested whether it indeed mimics the input graph, one might ask the question whether we are still generating a synthetic graph.

Further, synthetic data generated by GANs and ML models can be more susceptible to attackers. Because this data is generated using real data, criminals may attempt to find the real data back from the synthetic data. Fu et al. [17] describe two types of attackers in their review of methods for generating privacy-preserving graph data with graph-based AI models: active and passive attackers. These attackers have two main goals according to [6]: they want to know whether edges exist between two target nodes, and they want to find the true identities of the users. Active attackers embed structures in the graph before publication to identify victims, by locating the embedded structures in the synthetic graph. These active attackers need access to the original graph and their operations are usually computationally costly. Passive attackers on the other hand use the assumption that entities in graphs belong to a unique, small identifiable graph. They observe the published graph to identify victims, and only use minimal external information about the nodes in the network. With this little information they are still able to identify victims, according to [6]. Methods exist to preserve privacy in ML models [17], but when not using real data at all, e.g. in graph-based models, this risk will be minimized.

3.4 Synthetic Data Generation in Finance

In finance, researchers have worked on generating synthetic financial networks. Most of this research was done for AML purposes. As previously mentioned, these networks are often weighted and directed, where the weights represent the amount or volume of the transaction, and the direction corresponds to the direction in which the money is transferred. In this section an overview of this research is given.

3.4.1 Agent-Based Synthetic Data

Generating synthetic data can be done by generating intelligent agents that represent financial entities. These agents can generate transactions following a certain behavior pattern that matches the real-world. In this section an overview of the most popular agent-based simulators for generating transaction data is given.

The payment simulator (Paysim) [30] is a simulator that generates mobile money transactions based on an original dataset which contains logs of a mobile money service from an African country. The model contains different entities: clients, which can make transactions, and merchants, which can accept payments from clients. Clients can make transactions with other clients or with merchants. Different transaction types are taken into account. The simulator was able to generate synthetic data that is similar to real data; however no network properties are considered in this model.

AMLSim [55, 50] is a data simulator that simulates data for anti money laundering purposes. This data is generated in two steps. First, the model generates a graph via NetworkX based on a given degree distribution. Then, the model generates transactions via PaySim, based on the given transaction distributions and dynamics which are observed in real data. Finally, money laundering patterns are injected in the network, resulting in a labelled dataset that can be used to train or test money laundering detection models.

Altman et al. [4] generated synthetic data based on AMLSim. They created a synthetic financial transaction generator that builds a multi-agent virtual world. In this world, some agents are criminals, which have illegitimate income to launder. When comparing AMLSim to AML-World, AML-World shows to model a much richer set of characteristics. AML-world models for example multiple currencies, where AMLSim does not. Also it models not only transfers, but also payments, credits, etc. This all makes that the model is able to generate very realistic datasets, containing perfectly labelled data.

However, as the authors of AML-World mention in the Future Work section, agent-based models can be instable. Making minor parameter tweaks can result in significant changes in the synthetic data. They mention that graph-based approaches can therefore be an interesting research avenue.

3.4.2 Graph-Based Synthetic Networks

Grishin et al. created a model that is able to generate large synthetic payment graphs [19]. Their graph contains transactions between three different client types: companies, clients and ATMs. Their generator first creates the entities, based on a hardcoded distribution, which contains averaged parameters. After that, the edges are created, which represents whether two nodes have had a transaction between them. This was done by taking a probability distribution of the node type connections; client-client, client-company, etc. These probabilities can be configured using generator parameters. This method was chosen to speed up the generating process. Finally, the transactions were added to the network, which were random.

The researchers injected three money laundering patterns in their graph, and made their project publicly available. The size of the graph can be customized. However, no validation was done to evaluate the quality of this generator.

Tian et al. [51] introduce a synthetic data generation method that utilizes transaction metadata, statistics, and domain expertise to produce extensive payment network data. They consider the application of their synthetic payment generator in the context of real-time payment (RTP) networks. The generator is claimed to be highly adaptable, offering configuration options, and claimed to go beyond simulating regular transactions by incorporating diverse and realistic fraudulent activities and money laundering patterns.

They generate three transaction graphs: a graph for legitimate transactions, a graph for fraudulent transactions, and graph for money laundering transactions. These three are then combined to obtain the final synthetic graph. The transaction graphs are generated by first creating a weighted undirected base graph with fixed number of nodes and average node degree, and after that the dynamic legitimate graph is generated by sampling randomly from the base-graph. Multiple legitimate Barabasi-Albert (BA) graphs are generated, but other graphs can be used as well if the user desires. In this way, the desired clustering coefficient can be reached. Finally, fraud and money laundering patterns are generated and injected in the legitimate network to obtain the RTP network. Unfortunately, it is not known what transactional data is used, nor what the weights in the base graph represent, making the research irreproducible.

Validation of the RTP network generator is challenging, as no real dataset can be used for comparison. For this reason, the authors came up with two creative ways of validating their generator using node embeddings. They first visualized the node embeddings using the t-Distributed Stochastic Neighbor Embedding (t-SNE) method [53] and showed that similar accounts are close to each other in 2D space. Second, they built an unsupervised ML model (an auto-encoder) to detect fraud and demonstrated that the performance of this model increases when node embeddings are used as features for training.

3.4.3 Other Synthetic Transaction Datasets

In this section, we'll provide an overview of alternative approaches to generating financial transaction data. These techniques are not graph-based or agent-based, but are still noteworthy, although they all need real data for generating synthetic data.

Panigrahi et al. [40] propose a novel approach for credit card fraud detection. Their system was tested by generating synthetic data, using a synthetic data simulator. This simulator consists of three components: a Markov modulated poisson process module (MMPPM) for the timestamps of the transactions, a Genuine Gaussian distribution module (GGDM) for the transactions of genuine customers, and a Fraud Gaussian distribution module (FGDM) for the fraudulent transactions. Both Gaussian distribution modules have their own parameters for the mean μ and standard deviation σ. The authors did not validate their generated data.

Ul Haq et al. [21] also wrote a paper about generating synthetic transaction datasets for fraud detection, using a technique they refer to as innovative. This technique is claimed to allow generating synthetic datasets of any size, replicating characteristics of actual fraud data and therefore supporting fraud detection research. The technique generates characteristic measures using Ripple Down Rules (RDR) ruleset (an approach to knowledge acquisition), classification, and probability distribution. This approach leads to data with similar characteristics as the reference data.

This synthetic data was validated in terms of classification accuracy using C4.5, RDR, Naïve Bayes and RandomForest classification techniques. Further, instance-based learning classification techniques were used for classification accuracy, as these use similarity functions to classify the instances to nearest neighbor instances. Empirical evaluation showed a high level of accuracy, meaning the generated data was very similar to the reference data. The reference data was an obfuscated dataset of 1775 banking transactions.

Jensen et al. [22] created a synthetic dataset to benchmark anti-money laundering (AML) methods, synthAML. Real data from a Danish bank was used to generate this synthetic dataset. The Danish bank data contains 16 million transactions, and 20.000 AML alerts, and consists of two tables, one with alerts and one with transactions. The authors captured the dependencies between these tables by using the SDV library [41], which employs conditional parameter aggregation. This resulted in an extended alert table, which holds all conditional parameters. A Gaussian copula process is then applied to the extended table, resulting in a probabilistic model which accounts for covariance between the original alert features, and the conditional distribution parameters of the associated transactions. This way, an observation can be simulated by sampling an observation from the extended alert table, and use this to simulate the associated transactions. The authors validated their synthetic data in two ways. First, they compared the distribution of the synthetic data with the real data. Second, they used ML experiments to measure the performance of these models on synthetic data. Then, the model was used on real data, to see how it performs in the real world. As a baseline comparison, the authors trained and tested the model using only the real data.

3.4.4 Summary

Multiple methods exist to generate synthetic data in finance. A promising method is agent-based modelling. However, this method can be unstable, which is undesirable. Graph-based approaches may solve this problem, making them an interesting research topic. Other approaches exist, but these require real datasets, while in this research we aim to generate data without real data. Therefore, we focus on graph-based approaches.

4 Model

In this section, the model developed to generate a synthetic transaction dataset is described, focusing on an intrabank network. The model is an evolutionary model for graphs. At every time step, there can be densification (adding edges) or growth (adding a node and the corresponding number of edges). The model works with node-types. This means each node is assigned a node-type at birth, which comes with certain properties. For example, a matrix $\boldsymbol{H}$ determines which node-types can connect to other node-types. With this matrix, the probability of generating an edge from a given node-type to another can be calculated. Further, we have a matrix $\boldsymbol{N}$ with specifies the number of incoming and outgoing edges of a node-type generated at birth. When a node is created, and a node-type is chosen to connect to, the specific node with this node-type will be chosen preferentially. At the end of each time-step, b weights are distributed uniformly at random to existing edges. The model can be found on `https://github.com/kariekanarie/graph_model`.

4.1 Model components

We developed a model that is capable of generating directed networks, here the parameters of the model are described.

- p is the probability of generating a node. $(1-p)$ is the probability of doing a densification step.
- r is the number of node-types
- $\boldsymbol{q} \in (\mathbb{R}^+)^r$ a vector of size r where entry q_i is the probability of generating a node of type $i \in \{0, .., r\}$. The entries of $\boldsymbol{q}$ satisfy the condition $\sum q_i = 1$.
- d_{in} and d_{out} is the number of added incoming and outgoing edges during the densification step.
- $\mathbf{H} \in (\mathbb{R}^+)^{r\times r}$
  With the non-negative arbitrary real elements H_{ij}. H_{ij} are suitably chosen constants that establish the probability of generating edges between the given node-types. If we generate an outgoing edge for a node of type i, we consider row i of the matrix to calculate the probability to which type it will connect. For incoming edges we consider column i. Calculating for example the probability of choosing node-type j given node-type k when generating an outgoing edge $k \rightarrow j$ can be calculated by:

  $$\frac{H_{kj}}{\sum_{i=1}^{r} H_{ki}} \tag{1}$$

  We assume that the sum of items in H_{ki} is not equal to zero.
- $\mathbf{N} \in (\mathbb{N})^{r\times 2}$
  The new node matrix, with non-negative integer elements N_{ij}. Each node type is assigned a specific number of incoming and outgoing edges that it will generate at birth. The first column represents the number of incoming edges, while the second column represents the number of outgoing edges.
- b is the number of times a weight of 1 is added to an existing edge in each time-step.

4.2 Model execution

We start with a seed graph, containing 5 nodes which are fully connected in both directions. At each iteration, the following steps are performed:

- With probability p a new node joins the network. This node is assigned a type c according to $\boldsymbol{q}$, and receives N_{c0} incoming and N_{c1} outgoing edges. For each edge, the type of the source or target node is determined by calculating the probabilities using $\mathbf{H}$. For an incoming edge from nodes of type j, this probability is found by calculating for $k \in [0, \cdots, r]$

$$\frac{H_{kj}}{\sum_{i=1}^{r} H_{ij}} \tag{2}$$

We assume that the sum of items in H_{ij} is not equal to zero, and choose type k using these probabilities. For an outgoing edge we can calculate the probabilities using equation 1 for $j \in [0, \cdots, r]$. A node of the chosen type is then selected to connect with according to degree preferential attachment.

- With probability $1 - p$, d_{in} random nodes are uniformly chosen that get an incoming edge among these for which $N_{i0} > 0$ and d_{out} random nodes are chosen that will get an outgoing edge among these for which $N_{i1} > 0$. The chosen node will connect to a node-type by using the probabilities which can be calculated by Equation 1 (for outgoing) and Equation 2 (for incoming). When this node-type is chosen, the endpoint of the edge will be chosen using degree preferential attachment. If an edge already exists between the selected nodes, no new edge is added.
- Finally, b edges are chosen uniformly at random (with possible repetitions), and the weight of each edge will be increased by 1.

4.3 Degree distribution

We now analyze the resulting degree distribution of the networks generated by the described model. Since our model produces directed networks, we distinguish between in-degree and out-degree distributions. The degree distribution is investigated by analyzing the expected rate at which node i of type c that is already in the network acquires new edges. Acquiring edges can happen in two cases:

1. During a network growth step, where a node i' is added to the network whose edges might connect to node i.
2. During a densification step, where edges are added to the network and node i may be chosen randomly or preferentially.

We differentiate between these two cases.

4.3.1 Network growth

Since all nodes are given incoming and outgoing degrees at birth as specified by $\boldsymbol{N}$, our analysis focuses on the in-degree and out-degree that existing nodes receive when new nodes are added to the network.

Let us consider timestep t, when a node with type c' is born. Since it forms $N_{c'0}$ incoming and $N_{c'1}$ outgoing edges, all the other nodes in the network have a probability of receiving out-degree and in-degree respectively.

Given c' as the type of the new node, the probability of this node connecting to a node of a type c can be written based on Equations 1 and 2. The probability of the new node connecting to a *particular* node i of type c is then based on the degree of node i:

$$\Pi_{in,i}^{c} = \frac{k_{in,i}^{c}(t)}{\sum_j k_{in,j}^{c}(t)}, \quad \Pi_{out,i}^{c} = \frac{k_{out,i}^{c}(t)}{\sum_j k_{out,j}^{c}(t)} \tag{3}$$

Furthermore, the matrix $\boldsymbol{N}$ defines how many incoming and outgoing edges the new node forms, given its type c'. Putting these together, we obtain the probability of a node i of type c to receive an in-coming edge upon the birth of a node of type c':

$$\mathbb{B}_{in,i}^{c} = \left(\sum_{c'=1}^{r} N_{c'0} \cdot q_{c'} \cdot \frac{H_{c'c}}{\sum_{l=1}^{r} H_{c'l}} \right) \Pi_{in,i}^{c} \tag{4}$$

Similarly, node i's probability of receiving a point of out-degree is:

$$\mathbb{B}^{c}_{out,i} = \left(\sum_{c'=1}^{r} N_{c'1} \cdot q_{c'} \cdot \frac{H_{cc'}}{\sum_{l=1}^{r} H_{lc'}} \right) \Pi^{c}_{out,i} \tag{5}$$

4.3.2 Densification

During densification, d_{in} nodes are chosen uniformly at random that receive an incoming edge, and d_{out} that will get an outgoing edge. The endpoints of these edges are chosen preferentially. For a node i of type c, getting an incoming edge can happen in two different ways: during in-densification and during out-densification. For in-densification, the probability of node i being chosen depends on the number of nodes in the network that can receive an incoming edge at timestep t:

$$\sum_{c':N_{c'0}\neq 0} n_{c'}(t) \tag{6}$$

where $n_{c'}(t)$ refers to the number of nodes of type c' at timestep t. Then the probability of picking node i is:

$$\frac{1}{\sum_{c':N_{c'0}\neq 0} n_{c'}(t)} \tag{7}$$

For out-densification, the probability that a node of type c is chosen to be the endpoint of an outgoing edge depends on $\boldsymbol{H}$ and the number of nodes that can get an outgoing edge. This can be calculated by:

$$\sum_{c':N_{c'1}\neq 0} \left(\frac{n_{c'}(t)}{\sum_{c'':N_{c''1}\neq 0} n_{c''}(t)} \cdot \frac{H_{c'c}}{\sum_{l=1}^{r} H_{c'l}} \right) \tag{8}$$

Which is the sum over all types c' that can generate an outgoing edge during densification, for each c' the probability of being picked during densification, multiplied with the probability of generating an edge from a node of type c' to a node of type c.

The probability that a node of type c gets an outgoing edge can be calculated similarly. The probability of generating an outgoing edge for node i of type c during out-densification is:

$$\frac{1}{\sum_{c':N_{c'1}\neq 0} n_{c'}(t)} \tag{9}$$

The probability that a node of type c gets an outgoing edge during in-densification is:

$$\sum_{c':N_{c'0}\neq 0} \left(\frac{n_{c'}(t)}{\sum_{c'':N_{c''0}\neq 0} n_{c''}(t)} \cdot \frac{H_{cc'}}{\sum_{l=1}^{r} H_{lc'}} \right) \tag{10}$$

The expected rate at which a node i acquires incoming edges during densification can then be obtained by:

$$\mathbb{D}^{c}_{in,i} = d_{in} \cdot \frac{1}{\sum_{c':N_{c'0}\neq 0} n_{c'}(t)} + d_{out} \cdot \sum_{c':N_{c'1}\neq 0} \left(\frac{n_{c'}}{\sum_{c'':N_{c''1}\neq 0} n_{c''}} \cdot \frac{H_{c'c}}{\sum_{l=1}^{r} H_{c'l}} \right) \Pi(k^{c}_{in,i}) \tag{11}$$

The expected rate at which a node i acquires outgoing edges during densification can be obtained similarly, by:

$$\mathbb{D}^{c}_{out,i} = d_{out} \cdot \frac{1}{\sum_{c':N_{c'1}\neq 0} n_{c'}(t)} + d_{in} \cdot \sum_{c':N_{c'0}\neq 0} \left(\frac{n_{c'}}{\sum_{c'':N_{c''0}\neq 0} n_{c''}} \cdot \frac{H_{cc'}}{\sum_{l=1}^{r} H_{lc'}} \right) \Pi(k^{c}_{out,i}) \tag{12}$$

By combining Equations 4 and 11 the expected rate of acquiring incoming edges can be found, and similarly for outgoing edges by combining 5 and 12. The expected rate at which node i of type c acquires incoming edges can be calculated by:

$$\frac{dk^c_{in,i}}{dt} = p \cdot \mathbb{B}^c_{in,i} + (1-p) \cdot \mathbb{D}^c_{in,i} \tag{13}$$

And the expected rate of acquiring outgoing edges can be calculated by:

$$\frac{dk^c_{out,i}}{dt} = p \cdot \mathbb{B}^c_{out,i} + (1-p) \cdot \mathbb{D}^c_{out,i} \tag{14}$$

The number of edges grows proportionally with the number of iterations. We therefore hypothesize the degree distributions for in- and out-degree to be power-laws, as its expected rate of acquiring edges is similar to adding internal links of BA [3], which can be power-law distributed, or exponentially, depending on the parameters.

4.4 Weight distribution

In our model, the weights are added randomly each iteration. At each iteration, in expectation

$$m = p\left(\sum_{i=1}^{r} q_i N_{i0} + q_i N_{i1}\right) + (1-p)(d_{in} + d_{out})$$

edges are added. At each time step, b weights are added to the existing edges, with equal probability. Suppose at time $t = 1$ we add m edges and b weights. These edges will all get a weight of $\frac{b}{m}$ in expectation. Then, at time $t = 2$, again b weights are distributed among the existing edges, meaning the m new edges will get weight $\frac{b}{2m}$ and the edges that were generated at time $t = 1$ will have $\frac{b}{m} + \frac{b}{2m}$ weight in expectation. When continuing this process, at time i, m edges are generated that will get weight $\frac{b}{im}$. The expected weight of these m edges at time t will be

$$\begin{aligned} w_i^{(t)} =& \frac{b}{im} + \frac{b}{(i+1)m} + \ldots + \frac{b}{tm} \\ =& \frac{b}{m}\left(\frac{1}{i} + \frac{1}{i+1} + \ldots + \frac{1}{t}\right) \end{aligned}$$

This can be simplified using the approximation that $\sum_{k=1}^{r} \frac{1}{k} = \ln(n) + \gamma$, which results in:

$$w_i^{(t)} = \frac{b}{m}\left(\ln\left(\frac{t}{i}\right)\right)$$

Edges that have an expected number of received weight less than w at time t then satisfy:

$$w_i^{(t)} = \frac{b}{m}\left(\ln\left(\frac{t}{i}\right)\right) < w$$

These edges are born at time $i \geq t$, such that:

$$i > te^{-\frac{wm}{b}}$$

The fraction of edges that have an expected weight less than w at time t, or the cumulative weight distribution, is:

$$\begin{aligned} F(w) =& \frac{t - te^{-\frac{wm}{b}}}{t} \\ =& 1 - e^{-\frac{wm}{b}} \end{aligned}$$

We know that $F(x) = 1 - e^{-\lambda x}$ is the cumulative distribution of a distribution $\lambda e^{-\lambda x}$, meaning

$$p_w = \frac{m}{b} e^{-\frac{wm}{b}}$$

which is an exponential degree distribution. We thus hypothesize that the weights in our model follow an exponential distribution. Note that in this proof we assumed that m is a constant, while in reality it is not; m depends on probabilities such as p and $\boldsymbol{q}$, meaning we cannot be sure whether the weights will follow an exponential distribution. Also, we assume that the added weights are as per the expectation (this is a mean-field approximation).

4.5 Relation to BA

Our model generalizes the BA model [2] using some specific parameter setting. Only one type of node is needed, meaning $r = 1$. We do not consider directions, meaning we only need a value for the number of edges generated at birth: $N_{00} = m$. $p = 1$, to ensure no densification occurs. Finally, $b = 0$ to generate an unweighted network. This parameter setting will yield BA graphs.

5 Simulations

5.1 Model Parameters

This section analyzes, through simulations, how the parameters influence the properties of the resulting network. Using a fixed parameter setting, each parameter is analyzed individually to observe its effect. Although a complete parameter search would be more informative, this strategy was not chosen due to time constraints and to maintain clarity. Each parameter is analyzed over a specified range, and a graph is generated 15 times with 10,000 iterations. The mean and standard deviation are calculated for the analysis. The base settings that were used can be seen in Table 1. For simplicity, three node types were utilized: 'collectors', which have only incoming edges; 'distributors', which have only outgoing edges; and 'hybrid' nodes, which have both incoming and outgoing edges.

Parameter	**Value**
r	3
p	0.5
t	10000
d_{in}	2
d_{out}	2
$\mathbf{H}$	$\begin{bmatrix} 0 & 10 & 1000 \\ 0 & 0 & 0 \\ 0 & 500 & 2500 \end{bmatrix}$
$\boldsymbol{q}$	$(0.4, 0.3, 0.3)$
$\mathbf{N}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$
b	100

Table 1: Parameter Values Used in the Model

Creating a network with these parameters yields the metric values presented in Table 2. The clustering coefficient is low. Both degree and strength assortativity are also low, with slightly negative values. Overall, our metrics exhibit low standard deviations, indicating model stability. However, this stability does not extend to the power-law distribution of the in-degree, which shows a high standard deviation and a coefficient of variation (CV) of 64%. This suggests that the in-degree power-law of the model is unstable under the current parameter settings. A possible explanation will be provided in Section 5.2.

Metric	**Value**
Clustering Coefficient (CC)	0.0383 ± 0.0035
Assortativity (s)	-0.0209 ± 0.0083
Assortativity (d)	-0.1494 ± 0.0071
In-degree Power Law Exponent	2.7298 ± 1.7383
Out-degree Power Law Exponent	2.4676 ± 0.0289
Density	0.0010 ± 3.39E-05
Nodes	4995 ± 42.3
Edges	25559 ± 173.8

Table 2: Metrics for base parameters

5.1.1 Parameter p

For the parameter p, the values $[0.1, 0.3, 0.5, 0.7, 0.9]$ were investigated.

Hypothesis 1.1 Increasing p will result in lower clustering coefficient and density.

The results of increasing the value of p can be found in Table 3. We increased p from 0.1 to 0.9 and see that that the clustering coefficient indeed decreases from 0.33 ($p = 0.1$) to 0.0098 ($p = 0.9$), as well as the density, which decreases from 0.023 to 0.0002 for $p = 0.1$ and $p = 0.9$ respectively. Our hypothesis is thus supported by the results.

It is expected that as the value of p increases, the network will grow larger because more nodes will be generated. Consequently, this would result in a lower clustering coefficient and density, because there will be less densification steps.

Hypothesis 1.2 A higher value of p will yield higher degree assortativity.

The results can be found in Table 3. We observe that increasing the value of p indeed yields higher degree assortativity, supporting our hypothesis.

As mentioned, a high value of p will yield a lower number of densification steps. The strongly connected nodes will be less strongly connected with fewer densification steps, resulting in higher degree assortativity. While the assortativity remains negative, it is less negative than when performing many densification steps.

During the parameter analysis, the means and standard deviations of the graph metrics were calculated. An unexpected observation is the exceptionally large standard deviation in the power-law exponent for the in-degree distribution at low values of p. Interestingly, this is not the case for out-degree power-law. The values are shown in Table 4. An explanation of this outcome can be found in Section 5.2.

p	**CC**	**Density**	**Assort. (s)**	**Assort. (d)**	**Nodes**	**Edges**
0.1	0.3333 ± 0.0188	0.0231 ± 0.0013	-0.0594 ± 0.0302	-0.3378 ± 0.0206	1011.7 ± 40.6	28830.3 ± 173.2
0.3	0.0977 ± 0.0061	0.0032 ± 0.0001	-0.0416 ± 0.0150	-0.2198 ± 0.0164	3027.6 ± 57.9	25558.5 ± 129.9
0.5	0.0401 ± 0.0034	0.0010 ± 0.00002	-0.0377 ± 0.0100	-0.1768 ± 0.0099	4993.3 ± 40.6	20752.7 ± 119.4
0.7	0.0197 ± 0.0020	0.0004 ± 0.000008	-0.0573 ± 0.0061	-0.1530 ± 0.0123	7007.7 ± 37.0	18234.1 ± 113.3
0.9	0.0098 ± 0.0010	0.0002 ± 0.000002	-0.0647 ± 0.0101	-0.1028 ± 0.0064	8999.1 ± 37.0	15640.5 ± 107.6

Table 3: Clustering coefficient, density, strength assortativity, and degree assortativity statistics for different values of p

p	**Power-law Exp in**	**Power-law Exp out**	**Nodes**	**Edges**
0.1	11.21 ± 8.43	2.20 ± 0.081	1011.7 ± 40.6	28830.3 ± 173.2
0.3	4.02 ± 3.85	2.27 ± 0.032	3027.6 ± 57.9	25558.5 ± 129.9
0.5	3.40 ± 2.64	2.44 ± 0.030	4993.3 ± 40.6	20752.7 ± 119.4
0.7	2.80 ± 0.54	2.72 ± 0.063	7007.7 ± 37.0	18234.1 ± 113.3
0.9	2.88 ± 0.24	2.76 ± 0.183	8999.1 ± 37.0	15640.5 ± 107.6

Table 4: Power-law exponent statistics for in-degree and out-degree for different values of p

5.1.2 Parameter d_{in} and d_{out}

The values for d_{in} and d_{out} were chosen from the range $[1, 3, 5]$ for both parameters.

Hypothesis 2.1 High values for d will increase the network density and clustering coefficient.

For this simulation, values of 1, 3 or 5 were used for d_{in} and d_{out}, all combinations were investigated. The results of this analysis can be seen in Table 5, showing that when high values are used, such as 3 and 5, clustering coefficient and density are higher than when using lower values. Thus, the results support our hypothesis.

An explanation for this result is that generating more edges during densification is expected to yield higher network density and clustering coefficient.

Hypothesis 2.2 The cumulative degree distribution is influenced by d_{in} and d_{out} values. Generating more outgoing edges than incoming edges leads to a higher maximum in-degree, and vice versa.

Figures 2 and 3 illustrate the cumulative degree distributions for different d values. Here, different combinations of d_{in} and d_{out} are plotted to investigate the effect on the cumulative degree distribution. We observe an effect of these different parameter settings on the cumulative degree distribution, supporting our hypothesis. If $d_{in} = 1$ and $d_{out} = 5$, we observe a higher maximum in-degree, and if $d_{in} = 5$ and $d_{out} = 1$, we observe a higher maximum out-degree.

During densification, if more outgoing than incoming edges are generated, the in-degree will reach higher maximum values than the out-degree. This phenomenon arises due to preferential attachment: the more outgoing edges generated, the stronger the nodes with incoming edges will be, which results in a higher maximum in-degree. When the values are equal (see Figure 2), the out-degree has a slightly higher maximum value than the in-degree. This is likely caused by the lower proportion of collector (30%) and hybrid (30%) nodes, compared to distributor nodes (40%). During densification, if an edge already exists between two nodes, no new edge is added. Since there are more opportunities to generate an edge during out-densification, it is less likely that no edge is added compared to in-densification, which may result in a lower maximum in-degree.

d_{in}	d_{out}	**CC**	**Assort (s)**	**Density**	**Assort (d)**	**Nodes**	**Edges**
1	1	0.0278 ± 0.0032	-0.0554 ± 0.0141	0.00065 ± 1.36E-05	-0.1642 ± 0.0143	5007.5 ± 46.1	16209.9 ± 51.3
1	3	0.0305 ± 0.0019	0.0144 ± 0.0094	0.0010 ± 3.06E-05	-0.1468 ± 0.0086	5024.3 ± 52.7	25426.4 ± 171.7
1	5	0.0361 ± 0.0021	0.0592 ± 0.0075	0.0014 ± 4.05E-05	-0.0986 ± 0.0117	5019.1 ± 46.6	33818.7 ± 289.8
3	1	0.0527 ± 0.0035	-0.0569 ± 0.0090	0.0010 ± 2.04E-05	-0.1647 ± 0.0066	5003.1 ± 53.3	24933.1 ± 164.2
3	3	0.0502 ± 0.0035	-0.0346 ± 0.0113	0.0014 ± 3.21E-05	-0.1889 ± 0.0100	4996.7 ± 41.4	34243.3 ± 195.7
3	5	0.0516 ± 0.0030	-0.0060 ± 0.0122	0.0017 ± 3.20E-05	-0.1827 ± 0.0081	5003.6 ± 38.0	42491.2 ± 375.7
5	1	0.0707 ± 0.0043	-0.0525 ± 0.0076	0.0013 ± 3.38E-05	-0.1506 ± 0.0099	4986.7 ± 49.8	31854.3 ± 328.4
5	3	0.0613 ± 0.0040	-0.0419 ± 0.0090	0.0016 ± 5.11E-05	-0.1876 ± 0.0095	5016.5 ± 45.7	41545.3 ± 234.3
5	5	0.0626 ± 0.0027	-0.0287 ± 0.0105	0.0020 ± 5.98E-05	-0.2011 ± 0.0113	5009.4 ± 64.7	50167.4 ± 334.8

Table 5: Statistics for various combinations of d values

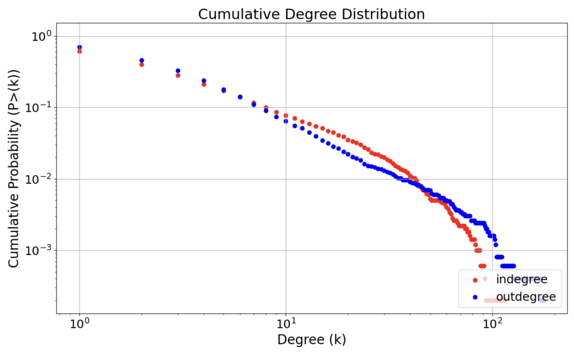
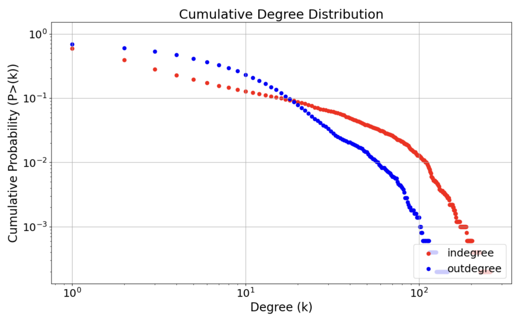
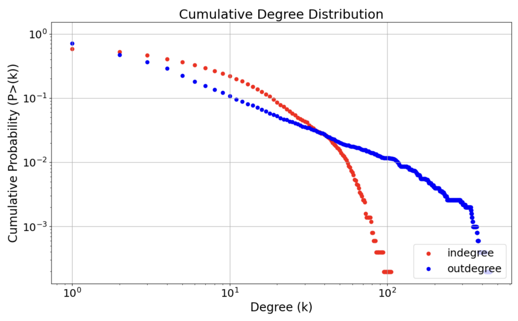


Figure 2: Cumulative degree distribution where d_{in}, $d_{out} = 1$



(a) Cumulative degree distribution where $d_{in} = 1$ and $d_{out} = 5$



(b) Cumulative degree distribution where $d_{in} = 5$ and $d_{out} = 1$

Figure 3: Cumulative degree distributions for different Dpi and Dpo values.

For degree assortativity, we observe that when more edges are generated during densification, assortativity decreases (Table 5). As explained in the previous paragraph, performing more densification steps yields lower assortativity because the highly connected nodes become even more connected, thereby lowering assortativity.

5.1.3 Parameter $\boldsymbol{q}$

For the parameter $\boldsymbol{q}$, the following ratios were investigated:

$$(0.2, 0.2, 0.6), \quad (0.2, 0.6, 0.2), \quad (0.2, 0.4, 0.4),$$

$$(0.4, 0.3, 0.3), \quad (0.6, 0.2, 0.2), \quad (0.8, 0.1, 0.1).$$

Hypothesis 3.1 The ratios used for node-types will yield similar node-type ratios in the generated graph.

Observations of the node types for different parameter settings confirm this expectation, indicating that the parameter works as intended. We find for each run a total of approximately $p \cdot t = 0.5 \cdot 10000 = 5000$ nodes, and each type occurs according to the probabilities that were specified in $\boldsymbol{q}$.

Hypothesis 3.2 The bow-tie structure aligns with the ratios of the parameter. The ratio of hybrid nodes will be similar to the ratio of nodes in the SCC, the ratio of distributor nodes with the ratio of nodes in IN and collector nodes with the ratio of OUT.

The resulting bow-tie ratios when analyzing this parameter can be seen in Table 6. These results support our hypothesis, as the ratios of distributor, collector and hyrid nodes align with the ratios of IN, OUT and SCC components respectively. However, they do not completely align, because of the definition of bow-tie, and the matrix $\mathbf{H}$. A hybrid node can also be a TUBE, and if a distributor only connects to collector nodes, it can be an OUTTENDRIL, or an INTENDRIL. Further, hybrid nodes can also be part of the IN or OUT component.

We assume that the bow-tie structure aligns with the ratios of the parameter. Here, SCC would be similar to a group of hybrid nodes. If we would only generate hybrid nodes and no other types, we can prove by induction that this yields an SCC.
Base case: In our model, the base-graph G is a fully connected directed network, which is an SCC.
Induction hypothesis: Assume after k iterations, G_k is a strongly connected component.
Induction step: We need to prove that after $k+1$ iterations the graph G_{k+1} also is an SCC. Let v be the node that is added to G_k. Let u be the node that gets an incoming edge from v, e.g. $v \to u$ and let w be the node that gives an outgoing edge to v, e.g. $w \to v$. Then, for each node $x \in G_k$, we know it can be reached by v,

because v can reach u, which can reach x. For each node $y \in G_k$ we know that y can reach v, because y can reach w, which can reach u. Since every node in G_{k+1} is reachable from every other node in G_{k+1}, we conclude that G_{k+1} is an SCC.

Further, IN refers to distributor nodes and OUT to collector nodes, as nodes in the IN component have edges to the SCC, and nodes in OUT component receive edges from the SCC.

Hypothesis 3.3 Generating more distributor or collector nodes yields higher clustering and lower degree assortativity.

The results of the assortativity and clustering coefficient during the parameter analysis can be found in Table 7. We observe that when the ratio of hybrid nodes is lower than the ratio of either distributors, collectors or both, we observe higher clustering and lower degree assortativity compared to the case where the ratio of hybrids is higher than the other types. Our results therefore support this hypothesis.

When generating more distributor or collector nodes, these nodes are likely to connect to hybrid nodes at birth. This happens because low-degree nodes (distributors and collectors) connect to high-degree nodes (hybrids), increasing the degree of the hybrid nodes. If more distributors and collectors are generated, these nodes are more likely to be picked during densification. Densification randomly selects nodes and connects them preferentially. For example, when more distributor nodes exist, these are more likely to be chosen during out-densification. This node will then likely connect to a hybrid node, due to **H**. As these distributor nodes gain more edges to hybrid nodes, the clustering coefficient will increase, because more triangles will be closed.

d	c	h	**SCC**	**IN**	**OUT**	**TUBES**	**INTEN**	**OUTTEN**	**OTH**
0.2	0.2	0.6	0.53	0.24	0.22	0.0055	0.0022	0.0012	0
0.2	0.6	0.2	0.18	0.22	0.60	0.0014	0.0032	0.00020	0
0.2	0.4	0.4	0.35	0.24	0.40	0.0048	0.0038	0.0012	0
0.4	0.3	0.3	0.27	0.42	0.30	0.0024	0.0012	0.0012	0
0.6	0.2	0.2	0.17	0.62	0.21	0.0028	0.0018	0.0012	0
0.8	0.1	0.1	0.087	0.81	0.10	0.00080	0.00020	0.0032	0

Table 6: Bow-tie components for different values of $\boldsymbol{q}$, d refers to distributor nodes, c to collectors and h to hybrids.

d	c	h	**CC**	**Assort (s)**	**Assort (d)**	**Density**	**Nodes**	**Edges**
0.2	0.2	0.6	0.0150 ± 0.0012	0.1002 ± 0.0154	0.0314 ± 0.0061	0.0011 ± 0.0000	5020.1 ± 49.9	27332.7 ± 122.8
0.2	0.6	0.2	0.0916 ± 0.0074	-0.1169 ± 0.0106	-0.2920 ± 0.0127	0.0010 ± 0.0000	4977.7 ± 59.4	24642.3 ± 217.7
0.2	0.4	0.4	0.0305 ± 0.0032	0.0312 ± 0.0089	-0.0949 ± 0.0122	0.0011 ± 0.0000	5004.8 ± 53.0	26156.8 ± 117.2
0.4	0.3	0.3	0.0396 ± 0.0036	-0.0407 ± 0.0109	-0.1791 ± 0.0111	0.0010 ± 0.0000	4995.3 ± 58.9	25559.6 ± 183.9
0.6	0.2	0.2	0.0653 ± 0.0045	-0.1144 ± 0.0144	-0.2614 ± 0.0162	0.0010 ± 0.0000	5004.0 ± 49.0	24567.8 ± 166.7
0.8	0.1	0.1	0.1442 ± 0.0100	-0.1798 ± 0.0151	-0.3341 ± 0.0280	0.0009 ± 0.0000	5001.1 ± 60.4	22675.7 ± 257.6

Table 7: Metric values for different values of $\boldsymbol{q}$
d refers to distributor nodes, c to collectors and h to hybrids.

5.1.4 Parameter $\boldsymbol{H}$

The values of **H** were also examined. Initially, we explored the impact of generating a single node type. To construct a network, this type had to be a hybrid; otherwise, the edges wouldn't connect. Thus, we also modified the parameter $\boldsymbol{q}$ accordingly, ensuring that only hybrid nodes could be generated. Next, we investigate the effect of generating two types: hybrids and distributors, and hybrids and collectors. Once again, adjusting $\boldsymbol{q}$ to generate these type of networks. In both cases, the probability of generating each type was set to 0.5. Finally we also generate networks which contain all three types. The different ratios that were used were:

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

The results can be seen in Table 8, and 9 here, h represents hybrid nodes, d distributors and c collectors.

Hypothesis 4.1 Generating only hybrid nodes yields one big strongly connected component (SCC), and thus only one bow-tie component (SCC).

The bow-tie ratios found were: (SCC: 1.0, IN: 0, OUT: 0, TUBES: 0, INTENDRILS: 0, OUTTENDRILS: 0, OTH: 0), supporting the hypothesis.

When generating only hybrid nodes, all nodes will get an in- and out-degree, meaning we generate one big SCC, and thus only one bow-tie component (SCC), as proven in *Hypothesis 3.2*.

Hypothesis 4.2 Generating only hybrid nodes yields higher clustering coefficient, density and degree assortativity compared to generating more types.

Table 8 presents the results. We observe that the clustering- and degree assortativity coefficient indeed decrease when generating multiple types of nodes. However, the decrease in density is only marginal.

When we generate only one type of node, receiving one incoming and one outgoing edge, is generated, we expect higher network density and clustering coefficients compared to scenarios where multiple types of nodes are generated. Additionally, degree assortativity tends to be higher when there is only one node type as similar nodes are more likely to connect. However, assortativity is not extremely high due to preferential attachment.

Hypothesis 4.3 When generating three types of nodes, with a high probability of connecting to highly interconnected hybrid nodes, degree assortativity will be low, and clustering coefficient high.

The results of the three different matrices used to investigate the effect of different ratios of connections between types can be seen in Table 9. In the first matrix, these ratios are all equal, meaning there is no preference generating edges between types. In the second matrix, the probability of connecting to a hybrid node is higher than to other types. In the third matrix, the probability is higher that distributor nodes connect with collector nodes and vice versa. We observe that the second matrix has the highest clustering coefficient, and the lowest assortativity, compared to the other cases. Thus, our hypothesis is supported by the results.

As explained in the previous paragraph, this occurs because when collectors and distributors are likely to connect to hybrid nodes, the existing dense connections among hybrid nodes are reinforced. Then, the assortativity will be low and clustering will be high.

Hypothesis 4.4 The bow-tie corresponding to the third matrix contains many INTENDRILS and OUTTENDRILS.

The found ratios of the bow-tie structure is: (SCC: 0.21, IN: 0.32, OUT: 0.24, TUBES: 0.020, INTENDRILS: 0.093, OUTTENDRILS: 0.12, OTH: 0). We observe a higher ratio of INTENDRILS and OUTTENDRILS compared to other bow-tie structures measured in this research. In the analysis of parameter $\boldsymbol{q}$, the highest ratios of INTENDRILS and OUTTENDRILS found were both 0.0032.

In the third matrix, distributor nodes have high probability of connecting to collector nodes, and collector nodes have high probability of connecting to distributor nodes. This results in a significant presence of INTENDRILS, attributed to collector nodes, and OUTTENDRILS, attributed to distributor nodes.

Types	CC	Assort (s)	Assort (d)	Density	Nodes	Edges
only h	0.1213 ± 0.0037	0.1448 ± 0.0077	0.0858 ± 0.0047	0.0012 ± 2.54E-05	5034.7 ± 62.1	29567.1 ± 113.9
d and h	0.0135 ± 0.0011	0.1106 ± 0.0113	0.0472 ± 0.0186	0.0011 ± 3.29E-05	5007.7 ± 56.9	26857.0 ± 122.4
c and h	0.0242 ± 0.0024	0.0902 ± 0.0065	0.0160 ± 0.0077	0.0011 ± 2.82E-05	5011.1 ± 30.3	26752.0 ± 85.6
d, c and h	0.0118 ± 0.0014	0.0851 ± 0.0172	0.0174 ± 0.0119	0.0010 ± 2.74E-05	4990.9 ± 45.8	25755.0 ± 135.3

Table 8: Metric values for different values of **H**, h refers to hybrid nodes, d to distributors and c to collectors.

$\boldsymbol{H}$	CC	Assort (s)	Assort (d)	Nodes	Edges
$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	0.0118 ± 0.0014	0.0851 ± 0.0172	0.0174 ± 0.0119	4990.9 ± 45.8	25755.0 ± 135.3
$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$	0.0325 ± 0.0024	-0.0344 ± 0.0106	-0.1676 ± 0.0123	5028.3 ± 46.6	25396.0 ± 168.8
$\begin{bmatrix} 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$	0.0098 ± 0.0011	0.0502 ± 0.0205	-0.0157 ± 0.0098	4998.8 ± 55.2	25683.4 ± 146.9

Table 9: Metric values for different values of $\boldsymbol{H}$

5.1.5 Parameter $\boldsymbol{N}$

For the parameter that defines the number of edges a new node with given type gets, the following values were investigated:

Case 1:	**Case 2:**	**Case 3:**	**Case 4:**	**Case 5:**
$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 2 \\ 2 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 5 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 5 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 5 & 5 \end{bmatrix}$

Hypothesis 5.1 Clustering increases when distributor and/or collector nodes generate more edges at birth than hybrid nodes.

In cases 2, 3 and 4, the distributor and/or collector node generate more edges at birth than hybrids. In Table 10 the results are presented. We observe that the clustering coefficients are higher in case 2, 3 and 4 compared to 1 and 5, supporting our hypothesis.

When distributor or collector nodes generate more edges at birth, the clustering coefficient will increase. These node-types are likely to connect to hybrid nodes, which are highly inter-connected. When these nodes generate more than 1 edge at birth, and connect to hybrid nodes, the probability is relatively high that these hybrid nodes are connected to each other, increasing the probability of closing a triangle. On average, more triangles will be closed, resulting in higher clustering.

Hypothesis 5.2 Degree assortativity decreases when distributor and/or collector nodes generate more edges at birth than the other node types.

Again, cases 2, 3 and 4 test this hypothesis. In Table 10 we observe that assortativity indeed is lower than in case 5, supporting our hypothesis, but not significantly when compared to case 1. Our hypothesis is therefore only partially supported.

When distributor or collector nodes generate more edges at birth, they tend to connect preferentially to high-degree hybrid nodes. This preferential attachment increases the degree of hybrid nodes even further, thereby lowering the assortativity of the network.

Case	CC	Density	Assort. (s)	Assort. (d)	Nodes	Edges
1	0.0413 ± 0.0037	0.0010 ± 2.85E-05	-0.0421 ± 0.0142	-0.185930 ± 0.016961	5000.3 ± 55.4	25494.9 ± 127.7
2	0.0588 ± 0.0042	0.0011 ± 2.59E-05	-0.0789 ± 0.0102	-0.204053 ± 0.010924	5011.9 ± 55.7	26963.5 ± 147.1
3	0.0806 ± 0.0077	0.0011 ± 2.42E-05	-0.1087 ± 0.0145	-0.198579 ± 0.017363	5012.1 ± 49.8	27920.9 ± 171.7
4	0.0567 ± 0.0041	0.0011 ± 2.59E-05	-0.0706 ± 0.0106	-0.197355 ± 0.009546	5000.4 ± 52.0	27593.3 ± 119.7
5	0.0280 ± 0.0020	0.0012 ± 2.00E-05	0.0342 ± 0.0099	-0.079377 ± 0.009885	5027.5 ± 47.9	29568.9 ± 104.8

Table 10: Metrics for different values of **N** with standard deviations

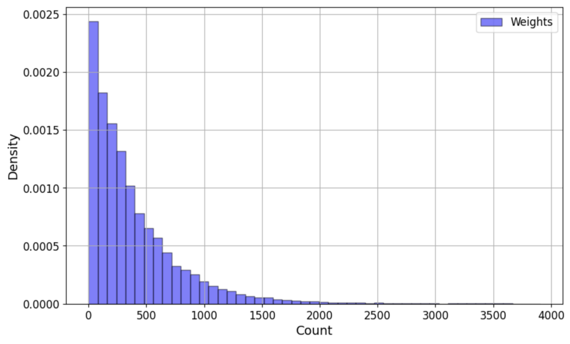


Figure 4: Edge weight distribution for $b = 1000$

5.1.6 Parameter b

For b, values were investigated in the range [10, 100, 1000, 10000].

Hypothesis 6.1 b does not significantly affect the network metrics, except for strength values and the weight distribution.

This can be explained by weights being added at the end of an iteration. They do not influence preferential attachment, as it is based on degree and not strength. The results support this hypothesis. Figure 4 illustrates the distribution of weights for $b = 1000$.

5.1.7 Combined parameters

Certain parameters yield interesting results when they are combined. For instance, experimenting with different values for **H** alongside $\boldsymbol{q}$ can produce interesting results. As observed in the previous paragraphs, the clustering coefficients of the graphs that were generated typically lie around 0.05.

Hypothesis 7.1 Generating a small proportion of hybrid nodes while allowing hybrid nodes to have high connectivity results in a high clustering coefficient and low assortativity.

For **H** the following value was taken:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1000 \end{bmatrix}$$

For $\boldsymbol{q}$ = (0.495, 0.495, 0.01), the results demonstrate a high clustering coefficient of 0.22 (std 0.02). Additionally, we observed a negative degree assortativity, -0,64 (std 0.05), which is the most negative value obtained so far (see Table 11). Thus, the results support our hypothesis. This indicates that generating nodes that are not hybrid and connecting them to hybrid nodes, which themselves are highly interconnected, leads to high clustering and low assortativity. The observed bow-tie ratio of this configuration is: (SCC: 0.091, IN: 0.44, OUT: 0.46, TUBES: 0, INTENDRILS: 0, OUTTENDRILS: 0, OTH: 0).

Hypothesis 7.2 If we generate more nodes that are either distributors or collectors, the clustering coefficient will be higher than in *Hypothesis 7.1*.

The same parameter setting was applied for **H**, but $\boldsymbol{q}$ was varied to investigate the effect on generating a higher proportion of distributor or collector nodes. We use: (0.9, 0.05, 0.05), and (0.05, 0.9, 0.05). The results for clustering and assortativity can be found in Table 11. As expected, the clustering coefficient values increased with these parameter settings, influenced by the densification process described previously. Interestingly, we also observed an increase in degree assortativity for these configurations. This unexpected result can be attributed to the more balanced distribution of nodes among types. There will be a lower difference in densification when distributors or collectors contain the same amount of nodes as the hybrid group. There will be no favour in densification for this type of node, resulting in more interconnected hybrid nodes compared to the first parameter setting.

d	c	h	**CC**	**Assort (s)**	**Assort (d)**	**Nodes**	**Edges**
0.495	0.495	0.01	0.2225 ± 0.0207	-0.5999 ± 0.0612	-0.6449 ± 0.0552	5018.5 ± 59.9	12916.2 ± 1499.0
0.9	0.05	0.05	0.3767 ± 0.0273	-0.2381 ± 0.0189	-0.3566 ± 0.0282	5008.1 ± 31.8	18640.1 ± 443.6
0.05	0.9	0.05	0.4246 ± 0.0235	-0.2617 ± 0.0148	-0.4797 ± 0.0202	5005.4 ± 46.2	20091.7 ± 273.5

Table 11: Metrics for different $\boldsymbol{q}$ values combined with **H** with standard deviations. d refers to distributor nodes, c to collectors and h to hybrids.

5.2 Stability

In this section, the stability of the model will be analyzed through examination of the standard deviations of metrics across various parameter ranges. Additionally, the impact on metrics will be assessed as networks grown bigger.

For the parameter p, we observe low CV values when the value of p is higher than 0.5. Conversely, for values of 0.5 and lower, we note high CV values for strength assortativity and power-law exponents of the in-degree. This variability can be attributed to the generation of smaller networks, where weights can be distributed in various ways, leading to differences in strength assortativity. Additionally, we observe that smaller graphs exhibit high power-law exponents for in-degree with significant standard deviations. This unexpected finding may be attributed to our densification mechanism. In our base settings, the probability of generating a distributor node is higher than that of generating a collector or hybrid node. During densification, edges are only added between nodes if no edge already exists. When there are few nodes capable of receiving incoming edges and existing nodes are already well-connected due to preferential attachment, it is possible that no edges are added during densification because the chosen nodes are already connected. This could explain the observed high power-law exponents and their variability.

We observe high standard deviations for strength assortativity across all parameters, likely attributable to the random distribution of weights.

For $\boldsymbol{H}$ we observe a high standard deviation in degree assortativity when generating three types using the matrix

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

This matrix implies no preference in generating an edge between two node types; instead, the type of endpoint for the edge is chosen uniformly at random. The node chosen from this type follows preferential attachment principles. However, using this matrix results in higher variability in assortativity compared to matrices where certain node types are favored over others.

Further, we generated larger networks with 100,000 and 200,000 iterations to investigate the effect on the network metrics when growing bigger networks. We used the same base settings as in the previous section, and measured the same metrics. The results can be found in Table 12.

Metric	**10k iter**	**100k iter**	**200k iter**
Median in-degree	1	1	1
StdDev in-degree	$1,1 \cdot 10^1$	$1.2 \cdot 10^1$	$1.2 \cdot 10^1$
Max in-degree	$1,2 \cdot 10^2$	$2.1 \cdot 10^2$	$2.1 \cdot 10^2$
Min in-degree	0	0	0
Median out-degree	2	2	2
StdDev out-degree	$1.4 \cdot 10^1$	$1.7 \cdot 10^1$	$1.8 \cdot 10^1$
Max out-degree	$2.1 \cdot 10^2$	$5.4 \cdot 10^2$	$6.1 \cdot 10^2$
Min out-degree	0	0	0
CC	$2.3 \cdot 10^{-2}$	$6.8 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$
Degree Assortativity	$-1.5 \cdot 10^{-1}$	$-1.1 \cdot 10^{-1}$	$-9.5 \cdot 10^{-2}$
Edge Density	$1.0 \cdot 10^{-3}$	$1.1 \cdot 10^{-4}$	$5.3 \cdot 10^{-5}$
Power-law in-degr	2,7298	2.43	6.42
Power-law out-degr	2,4676	2.44	2.43
Nodes	4995	50012	99872
Edges	25559	262906	528034

Table 12: Summary of network metrics.

We observe that the strengths of nodes are similar, especially the medians. For the clustering coefficient of 10k and 100k iterations, we see that it becomes approximately 10 times smaller, as well as the edge density. The degree assortativity increases slightly. The power-laws for out-degree are similar, but for in-degree we find a surprising high value when generating a network with 200,000 iterations. This may be caused by the value of p, as we explained in the beginning of this section. Low values of p can yield unstable power-law degrees for in-degree.

For 100,000 iterations and 200,000 iterations we generated only one graph, because it takes significantly longer to generate these large networks. Generating a graph with the base settings with 10.000 iterations takes approximately 20 seconds, while 100,000 and 200,000 iterations take 45 minutes and 8 hours respectively. We therefore do not know the standard deviations of the metrics, and thus cannot make conclusions about the stability of the measured metrics, other than the assumption that they might be similar to the smaller networks with the same parameter settings.

5.3 Weights

The weights we generate in these graphs follow an exponential distribution, as shown in Section 4.

The Rabobank weights both follow a power-law distribution with exponent $\alpha = 1.39$ and error $\sigma = 0.00034$. However, generating a power-law weight distribution takes significantly more time than generating an exponential distribution. To generate a power-law, weights must be added preferentially to the existing edges, meaning if many weights must be added, the time complexity increases for large graphs, because probabilities must be recalculated every iteration. For this reason, we chose to generate the weights randomly, resulting in an exponential distribution.

5.4 Conclusion

The model contains a wide range of parameters. An extensive parameter search might have been the most informative strategy for analyzing the behavior of our model, but due to time constraints we chose to take a fixed parameter setting and analyze almost all parameters separately. It was found that the assortativity coefficients were low, and sometimes slightly negative, meaning this model can generate networks that are neither assortative, nor disassortative, and slightly disassortative networks.

We found that less densification steps result in higher assortativity, and vice versa. Further, we found that generating a higher proportion distributor or collector nodes yield higher clustering coefficients, and lower assortativity, because these nodes will be picked more often during densification. The highest clustering coefficient found was 0.42, but in most cases the CC had a value around 0.05. Our model is capable of generating bow-tie structures with different distributions of components. The bow-tie structure can be edited by the parameters $\mathbf{H}$ and $\boldsymbol{q}$.

6 Use case: Rabobank

In this section, our model generates a synthetic transaction network that will be compared against a real network. We utilized the Rabobank dataset, which, to our knowledge, is the only publicly available non-synthetic dataset. This dataset contains 1.6 million nodes and 3.8 million edges.

We conducted clustering analyses on various network characteristics, such as clustering coefficient, in/out degree, in/out strength, and average neighbor in/out degree, to identify different types of entities (e.g., customers, companies) in the data. The clustering method used was K-Means, as other clustering algorithms were computationally too complex. Clear clusters emerged only when clustering on in- vs. out-degree or in- vs. out-strength. It was found that a large group (approximately 900,000 nodes) had an in-degree of zero, while a smaller, yet still significant group (around 300,000 nodes) had an out-degree of zero. The remaining nodes had both in- and out-degrees greater than zero. This pattern arises because the dataset only includes transactions within Rabobank and not with other banks, and because it is a sample of real data. While these clusters do not refer to specific entities, they will be used as node categories for the model.

Moving forward, we categorize nodes the same way as in Section 5.

6.1 Rabobank Dataset

In this section, metrics from the Rabobank network will be analyzed to identify suitable parameter settings for generating similar networks using our model. A parameter search was conducted to determine the settings that produce a graph most resembling the Rabobank dataset. The model was fitted on density, degree assortativity, clustering coefficient and degree power-law exponents. The results of this parameter search can be found in Table 17 presented in the Appendix. For the Rabobank network, the metrics can be seen in Table 13.

Metric	**Rabobank**	**Unweighted**
Median In Strength	0	0
StdDev In Strength	$4.1 \cdot 10^3$	$3.3 \cdot 10^1$
Max In Strength	$5.0 \cdot 10^6$	$2.0 \cdot 10^4$
Min In Strength	0	0
Median Out Strength	1	1
StdDev Out Strength	$2.9 \cdot 10^3$	$2.0 \cdot 10^1$
Max Out Strength	$3.1 \cdot 10^6$	$1.3 \cdot 10^4$
Min Out Strength	0	0
CC	-	$1.7 \cdot 10^{-2}$
Assortativity	$-5.1 \cdot 10^{-3}$	-0.024397
Edge Density	-	$1.0 \cdot 10^{-6}$
Power-law in-degr	-	3.3
Power-law out-degr	-	4.3
Nodes	1622173	-
Edges	3821514	-

Table 13: Rabobank metrics.

Observe that the median out-degree and strength are both 1, while the median in-degree and strength are both 0. In other words, the median node transacts with one account only once. Because of this phenomenon, the value of N_{01} was chosen to be 1. Similarly, for N_{10}, we also selected a value of 1, as indicated by the degree distribution shown in Figure 8, which reveals a significant number of nodes with an out-degree of 0 and an in-degree of 1. For N_{20} and N_{21}, values of 5 and 1 were investigated.

Additionally, note that the power-law exponent for in-degree is smaller than the exponent for out-degree. This indicates that the in-degree values are more distributed, meaning the power-law function has a longer tail. p was investigated for values higher than 0.5 to obtain a stable power-law. Further, the network is neither assortative, nor disassortative, and that the assortativity coefficient is slightly negative. The clustering coefficient is low, as is the density. Because of this low density, p was investigated for the range of [0.7, 0.8, 0.9]. d_{in} and d_{out} were investigated for the range of [1, 2, 5, 10].

When investigating different types that occur in the Rabobank data, we observe the following matrix with the number of edges between different types:

$$\begin{bmatrix} 0 & 8624 & 1224003 \\ 0 & 0 & 0 \\ 0 & 457620 & 2435032 \end{bmatrix}$$

This matrix was used as $\boldsymbol{H}$, so similar ratios would be generated in the networks. Further, when looking at the occurrence of types, we find the following ratios: distributors: 0.58, collectors: 0.20, hybrids: 0.22. These ratios are used for $\boldsymbol{q}$.

When investigating these parameters and the metric values of density, assortativity, clustering coefficient and power-law exponents, it was found that the values of parameters resulted in the best fit were the values shown in Table 14.

Parameter	**Value**
r	3
p	0.9
t	100000, 200000
d_{in}	2
d_{out}	2
$\boldsymbol{H}$	$\begin{bmatrix} 0 & 8624 & 1224003 \\ 0 & 0 & 0 \\ 0 & 457620 & 2435032 \end{bmatrix}$
$\boldsymbol{q}$	$(0.58, 0.20, 0.22)$
$\boldsymbol{N}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 5 & 5 \end{bmatrix}$
b	30

Table 14: Parameter values used for generating a synthetic Rabobank network

6.2 Analysis

In this section, the parameter settings that resulted in the best fit were used to grow two larger graphs, one of size 100,000 and one of size 200,000. These will be compared to the Rabobank network.

Metric	**Rabobank**	**Unw.**	**100k**	**Unw.**	**200k**	**Unw.**
Median In Strength	0	0	0	0	0	0
StdDev In Strength	$4.1 \cdot 10^3$	$3.3 \cdot 10^1$	$1.69 \cdot 10^2$	$6.29 \cdot 10^0$	$1.73 \cdot 10^2$	$6.35 \cdot 10^0$
Max In Strength	$5.0 \cdot 10^6$	$2.0 \cdot 10^4$	$9.38 \cdot 10^3$	$2.62 \cdot 10^2$	$1.35 \cdot 10^4$	$3.45 \cdot 10^2$
Min In Strength	0	0	0	0	0	0
Median Out Strength	1	1	9	1	9	1
StdDev Out Strength	$2.9 \cdot 10^3$	$2.0 \cdot 10^1$	$8.41 \cdot 10^1$	$3.65 \cdot 10^0$	$8.56 \cdot 10^1$	$3.73 \cdot 10^0$
Max Out Strength	$3.1 \cdot 10^6$	$1.3 \cdot 10^4$	$1.90 \cdot 10^3$	$1.20 \cdot 10^2$	$2.23 \cdot 10^3$	$1.68 \cdot 10^2$
Min Out Strength	0	0	0	0	0	0
CC	–	$1.7 \cdot 10^{-2}$	–	$5.0 \cdot 10^{-3}$	–	$5.8 \cdot 10^{-3}$
Assortativity	$-5.1 \cdot 10^{-3}$	$-2.4 \cdot 10^{-2}$	$-5.5 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$-5.0 \cdot 10^{-2}$
Edge Density	–	$1.0 \cdot 10^{-6}$	–	$2.5 \cdot 10^{-5}$	–	$1.3 \cdot 10^{-5}$
Power-law in-degr	–	3.30	–	3.76	–	3.98
Power-law out-degr	–	4.31	–	4.07	–	4.25
Nodes	$1.62 \cdot 10^6$	–	$9.01 \cdot 10^4$	–	$1.80 \cdot 10^5$	–
Edges	$3.82 \cdot 10^6$	–	$2.03 \cdot 10^5$	–	$4.05 \cdot 10^5$	–

Table 15: Combined metrics for Rabobank and different iterations

The metrics of our generated networks can be found in table 15. We observe that the median in- and out-degree for the unweighted variants of the model match the ones found in the Rabobank network. The median in-degree is 0 and out-degree is 1. Further, we observe that the maximum in- and out-degree of the Rabobank network are 20,000 and 13,000. In contrast, for our generated graphs, these values were considerably smaller, ranging between 120 and 345. Moreover, the maximum in-strength and out-strength in our generated graphs are also notably lower compared to those in the Rabobank network. This difference may partly stem from the smaller size of our generated networks compared to the Rabobank network. Furthermore, it is important to note that the weights in our network follow an exponential distribution, while the Rabobank network weights follow a power-law distribution.

The in- and out-degree distributions of Rabobank follow a power-law with exponent 3.3 and 4.3 respectively. The power-law for out-degree of our 100,000 iterations network was close to the one observed in Rabobank, and with a value of 3.76 slightly higher. The out-degree was 4.07 which is slightly lower. For 200,000 iterations we find a very similar power-law for out-degree, $\alpha = 4.25$, but the value for in-degree was further off, and too high: $\alpha = 3.98$.

The clustering coefficients in our generated networks are low when comparing to Rabobank. Our networks both showed a clustering coefficient of around 0.005, where the one in Rabobank is 0.017.

The assortativity in Rabobank is close to zero, but slightly negative. The value for unweighted assortativity was -0.024, and weighted assortativity was -0.0051. In our generated networks we find for unweighted networks a value of 0.021 (100k) and -0.050 (200k), and for weighted -0.055 (100k), and 0.018 (200k). Comparing the weighted assortativity might not be informative, as the weights in our network are distributed randomly. The unweighted assortativity for 200,000 iterations was close to the assortativity observed in the Rabobank network.

Finally, the edge density of Rabobank was 0.000001. In our networks, this was equal to 0.000025 (100k) and 0.000013 (200k). The density in our generated graphs is higher than the density observed in Rabobank. However, the Rabobank graph is larger, which means that when our generated graphs would grow larger with the same parameter settings, the value for density might be more similar to the Rabobank value.

When investigating the different node-types that occur in the Rabobank data, we find the following matrix:

$$\begin{bmatrix} 0 & 8624 & 1224003 \\ 0 & 0 & 0 \\ 0 & 457620 & 2435032 \end{bmatrix}$$

Our generated networks showed the following matrices of edges between types:

$$\begin{bmatrix} 0 & 960 & 93684 \\ 0 & 0 & 0 \\ 0 & 40752 & 67721 \end{bmatrix}, \quad \begin{bmatrix} 0 & 2895 & 280448 \\ 0 & 0 & 0 \\ 0 & 122816 & 202187 \end{bmatrix}$$

When calculating the ratios for generating an outgoing edge, we find for Rabobank the following matrix:

$$\begin{bmatrix} 0 & 0.007 & 0.993 \\ 0 & 0 & 0 \\ 0 & 0.16 & 0.84 \end{bmatrix}$$

For our generated networks we find the following for 100k and 200k respectively:

$$\begin{bmatrix} 0 & 0.01 & 0.99 \\ 0 & 0 & 0 \\ 0 & 0.38 & 0.62 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0.01 & 0.99 \\ 0 & 0 & 0 \\ 0 & 0.38 & 0.62 \end{bmatrix}$$

We observe that for distributor nodes the ratios are similar, but for hybrid nodes the ratios differ. When calculating the ratios for generating an incoming edge, we find for Rabobank the following matrix:

$$\begin{bmatrix} 0 & 0.18 & 0.33 \\ 0 & 0 & 0 \\ 0 & 0.82 & 0.67 \end{bmatrix}$$

For our generated networks we find the following for 100k and 200k respectively:

$$\begin{bmatrix} 0 & 0.02 & 0.58 \\ 0 & 0 & 0 \\ 0 & 0.98 & 0.42 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0.02 & 0.58 \\ 0 & 0 & 0 \\ 0 & 0.98 & 0.42 \end{bmatrix}$$

The ratios for incoming edges in our generated networks are different than the ones found in the Rabobank, especially for hybrid nodes. This is likely due to the densification step, where nodes from node-types are picked uniformly at random to generate edges. When picking nodes randomly, node-types that occur more often are in this case more likely to be picked. This way, the distributor nodes will be chosen more often during out-densification.

The bow-tie of Rabobank has the following structure: (SCC: 361816, IN: 928280, OUT: 322426, TUBES: 20, INTENDRILS: 1973, OUTTENDRILS: 7382, OTHER: 276). When turning these into ratios we find the following: (S: 0.22, IN: 0.57, OUT: 0.20, TUBES: 0.000012, INTENDRILS: 0.0012, OUTTENDRILS: 0.011, OTHER: 0.00017). Our generated network show the following bow-tie structures for 100k and 200k respectively: (SCC: 0.22, IN: 0.57, OUT: 0.20, TUBES:0, INTENDRILS: 0.0020, OUTTENDRILS: 0.0024, OTH: 0), and (SCC: 0.22, IN: 0.58, OUT: 0.20, TUBES: 0, INTENDRILS:0.0018, OUTTENDRILS: 0.0027, OTH: 0). This means the bow-tie structures of the networks are very similar to the Rabobank network. This is caused by parameters $\boldsymbol{q}$ and $\boldsymbol{H}$.

The cumulative in- and out-degree distribution of the Rabobank network can be seen in Figure 5, and for our generated graphs in Figure 6 and 7.

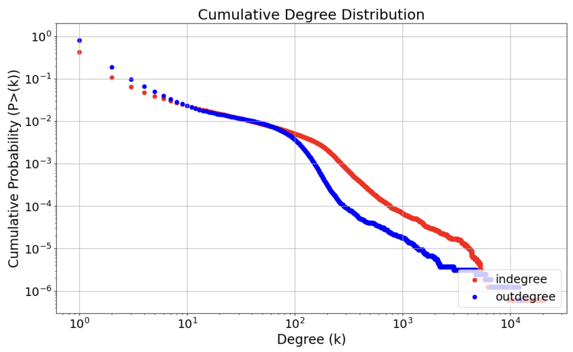


Figure 5: Cumulative in- and out-degree distribution of Rabobank

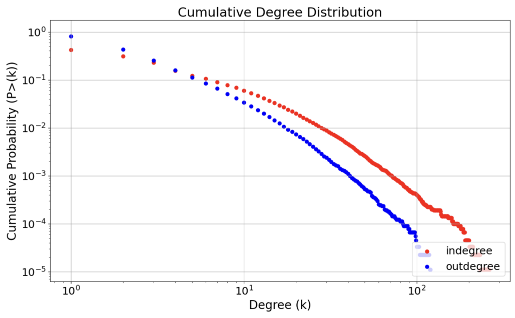


Figure 6: t = 100k iterations

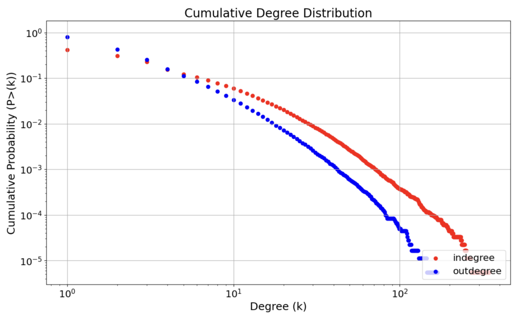


Figure 7: t = 200k iterations

We observe that the degree distributions have a long tail, indicating a power-law. However, for the Rabobank network we see that it has a slightly different curve compared to the networks generated with our model.

For degree correlation a heatmap of Rabobank can be seen in Figure 8, heatmaps of our generated graphs can be seen in Figure 9 and 10. The Spearman correlation between in- and out-degree of Rabobank was -0.15. For our generated graphs they were 0.0590 for 100k, and 0.0513 for 200k. In all graphs the correlation is low, but for the Rabobank network the correlation was slightly negative, while in our generated graphs the correlations were slightly positive.

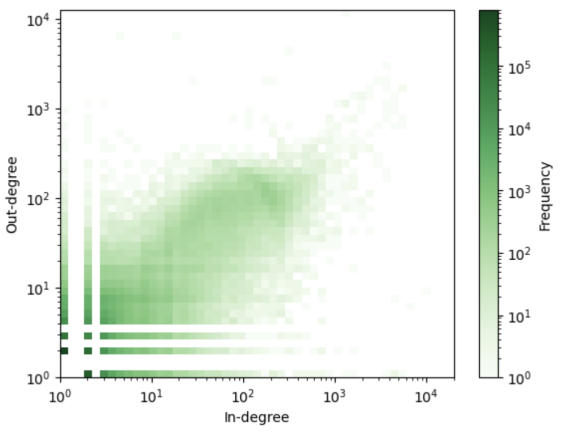


Figure 8: Heatmap of in- vs out-degree of Rabobank

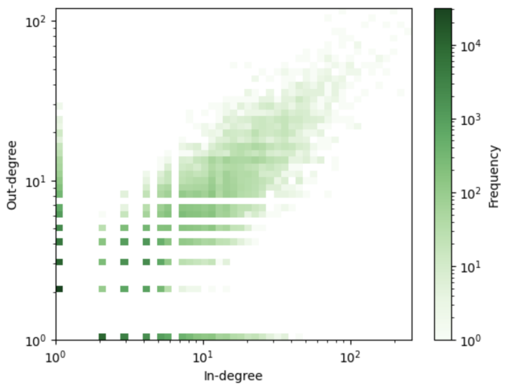


Figure 9: 100k iterations graph

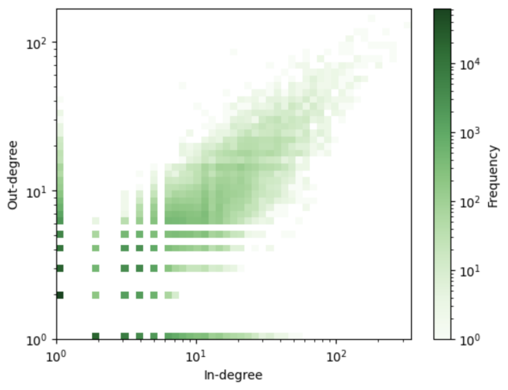


Figure 10: 200k iterations graph

Observe that the ranges in the Rabobank network are bigger, meaning the maximum in- and out-degrees are higher. This is due to the size of the graphs. We see a similar pattern where low degrees occur more often than high degrees, but in the Rabobank network we observe a wider range of different in- and out-degrees.

The weight distribution of Rabobank can be seen in Figure 11. The weight distribution for our generated networks can be seen in Figure 12 and 13.

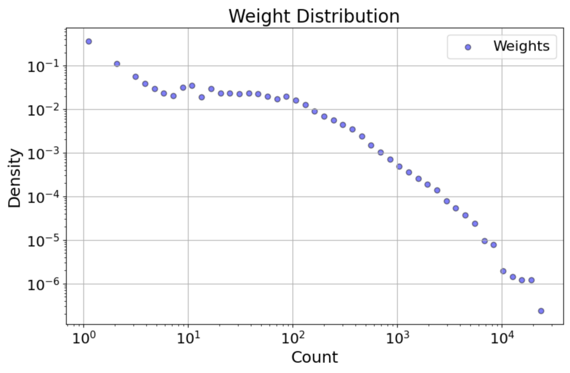


Figure 11: Weight distribution of counts in Rabobank data

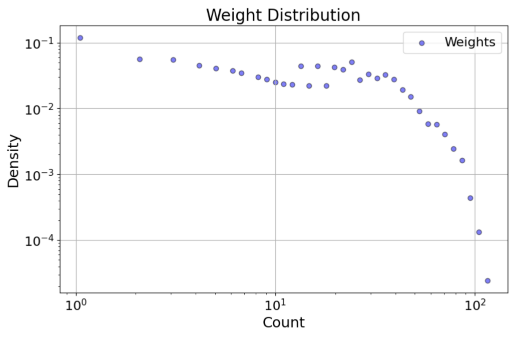


Figure 12: Weight distribution in 100k graph

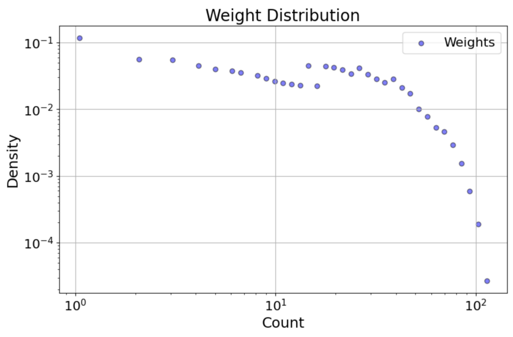


Figure 13: Weight distribution of 200k iterations graph

When fitting a power-law to the Rabobank weight distribution, we find an α of 1.39, and a statistic error σ of 0.00034. Our generated weight distribution likely follows an exponential distribution, as mentioned in Section 4. In the figures we see that the weight distribution of Rabobank decreases more gradually than the weight distribution of our generated graphs, indicating that Rabobank follows a power-law, and the generated weights follow an exponential.

The outcome of doing Leiden community detection on Rabobank and our generated graphs can be found in Table 16. The implementation for Leiden community detection is provided by [43].

Metric	**Rabobank**	**100k**	**200k**
Number of communities	165	16	24
Size of the smallest community	4	3645	4657
Size of the largest community	112341	7289	11294
Mean size of the communities	9831.35	5630.50	7501.25
StdDev of community sizes	19493.57	880.11	1668.36
Modularity	0.65	0.66	0.69
Nodes	$1.62 \cdot 10^6$	$9.01 \cdot 10^4$	$1.80 \cdot 10^5$
Edges	$3.82 \cdot 10^6$	$2.03 \cdot 10^5$	$4.05 \cdot 10^5$

Table 16: Community detection results using the Leiden algorithm on Rabobank data and our synthetic data.

We observe that the community structures of the generated networks have a similar modularity as the Rabobank network. However, the community structure of our generated networks do not align with the Rabobank network. The Rabobank network contains more communities, and also many small-sized communities. Our generated networks do not contain small-sized communities. Further investigation is needed to make a good comparison between these community structures.

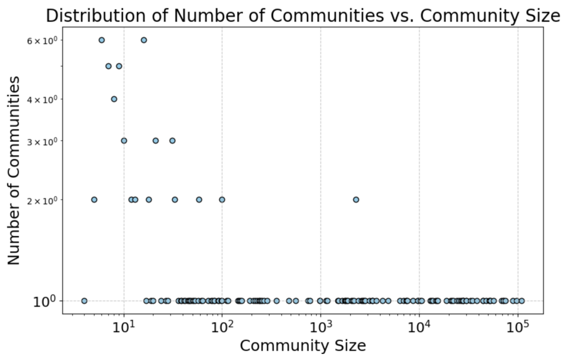


Figure 14: Nr of communities vs. community size Rabobank graph

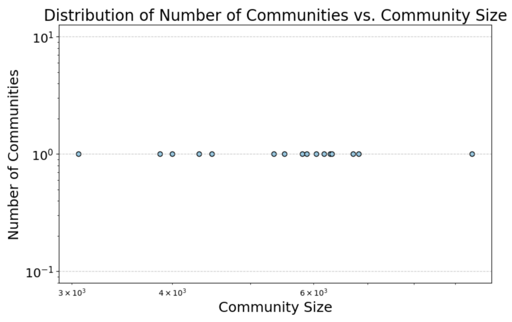


Figure 15: 100k iterations

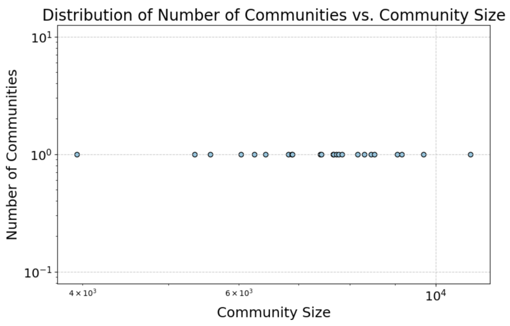


Figure 16: 200k iterations

The number of communities versus the community size for the Rabobank network is shown in Figure 14, and for our generated networks in Figures 15 and 16. We see that the Rabobank network contains some small sized communities, but that there are also a lot of larger sized communities, of which the largest contains 112341 nodes. In our generated networks, the number of communities is lower, and all sizes occur only once.

6.3 Summary

In this section a summary of the comparison between the Rabobank network and our generated networks is provided.

In the Rabobank network, the median in-degree was 0 and the median out-degree was 1, reflecting a network where most nodes have low connectivity. Our generated networks were similar. However, the Rabobank network had significantly higher maximum in-degrees and out-degrees compared to our generated networks. Despite this, both networks exhibited power-law distributions for degree. The Rabobank network had a higher clustering coefficient than our generated networks. This indicates a higher tendency for nodes to be interconnected compared to our generated networks. Assortativity was close to zero in Rabobank but slightly negative for unweighted and for weighted networks. In contrast, our generated networks showed varying assortativity values which were close to 0, but sometimes slightly positive, and sometimes slightly negative. The Rabobank network showed a lower edge density than our generated networks, potentially due to their smaller size. When examining the distribution of edges between different node types, we found that while distributor node ratios were similar between Rabobank and our networks, ratios for hybrid nodes and collector nodes differed, likely due to our densification process. The bow-tie structure analysis revealed strong similarities between Rabobank and our generated networks in terms of SCC, IN, and OUT components, influenced by parameters $\boldsymbol{q}$ and $\boldsymbol{H}$ in our model. Degree correlation, visualized in heatmaps, showed a slight negative correlation between in- and out-degree in Rabobank, but slight positive correlations in our generated networks. Weight distribution analysis indicated that Rabobank's weights follow a power-law, while our generated networks likely follow an exponential distribution, reflecting differences in edge weight distribution patterns. Community detection using the Leiden algorithm highlighted differences in community structures between Rabobank and our generated networks, with varying community sizes but similar modularity scores.

Summarizing, while our generated networks captured some aspects of the Rabobank network metrics, significant differences remain, particularly in extreme degree values, clustering coefficients, and community structures. These differences stem from variations in network size, edge weight distribution, and network generation parameters.

7 Discussion

7.1 Results

We provided a model that can mimic the bow-tie structure of a given graph. The degree distribution follows a power-law, and the weight distribution is exponential. The graphs exhibit a community structure, and are neither assortative nor disassortative. The model produces stable graphs when the value of p is not lower than 0.5. The model is able to generate power-law exponents for in- and out-degree, and edge density values similar to those of a real financial network.

7.2 Limitations

This research has several limitations, which are discussed in this section.

Firstly, it is important to note that the dataset on which the model is based is a sample of real data. The Rabobank dataset does not include all transactions between all users of Rabobank over the given time period. It is unclear how this sample was taken, whether it was a random sample or not. Furthermore, the dataset was aggregated, which means that some information might have been lost.

Another limitation is that the weights generated in the model follow an exponential distribution, whereas a power-law distribution was observed in the Rabobank data. Generating a power-law weight distribution would require adding weights using preferential attachment, but this becomes computationally complex when dealing with large graphs and large weights. When weights are added preferentially, the probability of choosing a node must be calculated every iteration, resulting in higher time-complexity. Therefore, we chose to add the weights randomly.

Additionally, the model has a limitation in that the matrix $\mathbf{H}$ does not specify the complete distribution of edge weights. During densification, a random node is selected from the already present nodes in the network. If one node type is overrepresented, there is a higher probability that this node type will be selected during densification, resulting in more edges generated by this node type, whereas in $\mathbf{H}$ the ratio might be different. A potential solution could be to first randomly select the node type and then randomly select a node from this type during densification, thereby maintaining the ratio specified in $\mathbf{H}$.

Validating a synthetic graph is challenging. The hierarchical building of metrics and a clear method to compare two graphs are not well-established. A graph kernel might offer a solution, but due to time constraints, this was not explored. Comparing the network metrics does not quantify how good the results actually are. We cannot tell whether our model out-performs state-of-the-art models, because there no baseline was used to compare with.

Furthermore, the parameter search for the use case was limited due to time constraints. With more time, a more extensive parameter search could potentially yield better results for our use case.

Finally, the model only generates intrabank networks based on one dataset from Rabobank. There is no information available about networks of other banks or about how these intrabank networks are connected to other intrabank networks

7.3 Future work

Firstly, modelling timestamps using a Poisson process could introduce temporal dynamics to the generated networks. These timestamps could be valuable for detecting criminal patterns in the data. Thus, the ability to generate realistic timestamps could be highly beneficial. However, validating these networks would be challenging due to the absence of realistic datasets containing timestamps.

Secondly, investigating and developing efficient algorithms for generating power-law weight distributions would make the model more realistic. Real weights observed in financial graphs often follow a power-law distribution. By accurately replicating this distribution in synthetic graphs, the model may better capture the complex dynamics and structural properties of real-world financial networks. Additionally, it would be interesting to investigate methods for increasing the clustering coefficient and improving the community structure. In our generated networks the clustering coefficient was too low, and the community structures did not align. Methods exist to generate a community structure, e.g. by generating local world properties [27], [58].

Finally, it would be interesting to investigate more on validating our model. In our research, an unfitted directed BA model was used which is not a state-of-the-art baseline. It would have been more informative to compare the generated networks to an agent-based model such as AMLWorld, or to a more complex graph generative model. Validation of the model may also be improved by using data from multiple banks or financial institutions to fit our model on. This can ensure its applicability across different financial networks. Further, investigating the properties of the bow-tie components of our generated graphs and comparing them to the bow-tie components of real data might also yield interesting insights about the properties of our generated networks and whether they are similar to real networks. Finally, an interesting research topic would be to develop robust methods to compare synthetic graphs with real networks, possibly utilizing graph kernels or other advanced techniques. It would be also interesting to explore how intrabank networks connect to each other. This way, one would be able to generate inter-bank networks, which may be useful for detecting criminal patterns.

8 Conclusion

In this thesis, we have presented a novel approach to generating weighted, directed, synthetic transaction networks, focusing on a single bank's intrabank network, using a graph-based generative model. Our model is designed to replicate characteristics observed in real-world financial networks, such as power-law degree distributions, weights, and community structures. The goal was to create a publicly available, reproducible tool that can generate synthetic financial transaction networks while preserving data privacy. This way, insight could be given into the evolution of these transaction networks.

We developed a graph generative model that incorporates node types identified through clustering analysis of a real transaction dataset. The model integrates parameters for node generation, edge densification, and a probability matrix to control the type-based connections within the network. It produces networks with desired structural properties and has been validated against a large intrabank network dataset from Rabobank.

Through extensive experimentation and analysis, we found that our model can generate synthetic networks that resemble some features of real financial transaction networks. The degree distribution follows a power-law, the networks are weighted and directed, have similar bow-tie structures, similar assortativity values, and exhibit a community structure. Our model demonstrated stable performance over 100,000 iterations or more, with generated networks showing comparable metrics to the real dataset in terms of degree distribution, edge density, and community detection.

Several limitations were identified. The use of a sampled and aggregated dataset limits the model's ability to capture the full complexity of real financial networks. Furthermore, the exponential distribution of weights in the model diverges from the observed power-law distribution in the real dataset. Future research could focus on improving the weight distribution modeling, exploring efficient algorithms for power-law distributions, and enhancing the model's validation methodologies.

In conclusion, this research provides a valuable contribution to the field of synthetic data generation for financial networks. The developed model offers a starting point for generating synthetic transaction networks that can be used for various applications, including training machine learning models for detecting criminal financial patterns. By making the model publicly available, we aim to facilitate further research and collaboration in this area.

References

[1] Amina Adadi. “A survey on data-efficient algorithms in big data era”. In: *Journal of Big Data* 8.1 (2021), p. 24.

[2] Réka Albert and Albert-László Barabási. “Statistical mechanics of complex networks”. In: *Reviews of modern physics* 74.1 (2002), p. 47.

[3] Réka Albert and Albert-László Barabási. “Topology of evolving networks: local events and universality”. In: *Physical review letters* 85.24 (2000), p. 5234.

[4] Erik Altman, Béni Egressy, Jovan Blanuša, and Kubilay Atasu. “Realistic synthetic financial transactions for anti-money laundering models”. In: *arXiv preprint arXiv:2306.16424* (2023).

[5] T Antal and PL Krapivsky. “Weight-driven growing networks”. In: *Physical Review E* 71.2 (2005), p. 026103.

[6] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. “Wherefore art thou R3579X? Anonymized social networks, hidden patterns, and structural steganography”. In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 181–190.

[7] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.

[8] Ginestra Bianconi. “Emergence of weight-topology correlations in complex scale-free networks”. In: *Europhysics letters* 71.6 (2005), p. 1029.

[9] Ginestra Bianconi and A-L Barabási. “Competition and multiscaling in evolving networks”. In: *Europhysics letters* 54.4 (2001), p. 436.

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[11] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. “Netgan: Generating graphs via random walks”. In: *International conference on machine learning*. PMLR. 2018, pp. 610–619.

[12] Isabella Castiglioni, Leonardo Rundo, Marina Codari, Giovanni Di Leo, Christian Salvatore, Matteo Interlenghi, Francesca Gallivanone, Andrea Cozzi, Natascha Claudia D’Amico, and Francesco Sardanelli. “AI applications to medical images: From machine learning to deep learning”. In: *Physica Medica* 83 (2021), pp. 9–24.

[13] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. “R-MAT: A recursive model for graph mining”. In: *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM. 2004, pp. 442–446.

[14] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. “Community detection via maximization of modularity and its variants”. In: *IEEE Transactions on Computational Social Systems* 1.1 (2014), pp. 46–65.

[15] Anne Condon and Richard M Karp. “Algorithms for graph partitioning on the planted partition model”. In: *Random Structures & Algorithms* 18.2 (2001), pp. 116–140.

[16] David N Fisher, Matthew J Silk, and Daniel W Franks. “The perceived assortativity of social networks: methodological problems and solutions”. In: *Trends in Social Network Analysis: Information Propagation, User Behavior Modeling, Forecasting, and Vulnerability Assessment* (2017), pp. 1–19.

[17] Dongqi Fu, Wenxuan Bao, Ross Maciejewski, Hanghang Tong, and Jingrui He. “Privacy-Preserving Graph Machine Learning from Data to Computation: A Survey”. In: *ACM SIGKDD Explorations Newsletter* 25.1 (2023), pp. 54–72.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

[19] M Grin. *How to generate a huge financial graph with money laundering patterns?* Apr. 2019. URL: `https://medium.com/@mgrin/how-to-generate-a-huge-financial-graph-with-money-laundering-patterns-5c3e490dd683`.

[20] Xiaojie Guo and Liang Zhao. "A systematic survey on deep generative models for graph generation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2022), pp. 5370–5390.

[21] Ikram Ul Haq, Iqbal Gondal, Peter Vamplew, and Robert Layton. "Generating synthetic datasets for experimental validation of fraud detection". In: *Conferences in Research and Practise in Information Technology.* Vol. 170. 2016.

[22] Rasmus Ingemann Tuffveson Jensen, Joras Ferwerda, Kristian Sand Jørgensen, Erik Rathje Jensen, Martin Borg, Morten Persson Krogh, Jonas Brunholm Jensen, and Alexandros Iosifidis. "A synthetic data set to benchmark anti-money laundering methods". In: *Scientific data* 10.1 (2023), p. 661.

[23] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. "Do the rich get richer? An empirical analysis of the Bitcoin transaction network". In: *PloS one* 9.2 (2014), e86197.

[24] Franziskos Kyriakopoulos, Stefan Thurner, Claus Puhr, and Stefan W Schmitz. "Network and eigenvalue analysis of financial transaction networks". In: *The European Physical Journal B* 71 (2009), pp. 523–531.

[25] CC Leung and HF Chau. "Weighted assortative and disassortative networks model". In: *Physica A: Statistical Mechanics and its Applications* 378.2 (2007), pp. 591–602.

[26] Chunguang Li and Guanrong Chen. "Modelling of weighted evolving networks with community structures". In: *Physica A: Statistical Mechanics and its Applications* 370.2 (2006), pp. 869–876.

[27] Menghui Li, Jinshan Wu, Dahui Wang, Tao Zhou, Zengru Di, and Ying Fan. "Evolving model of weighted networks inspired by scientific collaboration networks". In: *Physica A: Statistical Mechanics and its Applications* 375.1 (2007), pp. 355–364.

[28] Xurui Li, Xiang Cao, Xuetao Qiu, Jintao Zhao, and Jianbin Zheng. "Intelligent anti-money laundering solution based upon novel community detection in massive transaction networks on spark". In: *2017 fifth international conference on advanced cloud and big data (CBD).* IEEE. 2017, pp. 176–181.

[29] Seung-Hwan Lim, Sangkeun Matt Lee, Sarah Powers, Mallikarjun Shankar, and Neena Imam. "Survey of approaches to generate realistic synthetic graphs". In: *Oak ridge national laboratory* (2015).

[30] Edgar Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. "PaySim: A financial mobile money simulator for fraud detection". In: *28th European Modeling and Simulation Symposium, EMSS, Larnaca.* Dime University of Genoa. 2016, pp. 249–255.

[31] Erik Lundin. *Generating Directed & Weighted Synthetic Graphs using Low-Rank Approximations.* 2022.

[32] Rashi Maheshwari and Aashika Jain. *Top AI statistics and Trends in 2024.* Jan. 2024. URL: `https://www.forbes.com/advisor/in/business/ai-statistics/`.

[33] Fragkiskos D Malliaros and Michalis Vazirgiannis. "Clustering and community detection in directed networks: A survey". In: *Physics reports* 533.4 (2013), pp. 95–142.

[34] Thomas M Martinetz, Helge J Ritter, and Klaus J Schulten. "Three-dimensional neural net for learning visuomotor coordination of a robot arm". In: *IEEE transactions on neural networks* 1.1 (1990), pp. 131–136.

[35] Mark Newman, Albert-László Barabási, and Duncan J Watts. *The structure and dynamics of networks.* Princeton university press, 2011.

[36] Mark EJ Newman. "Assortative mixing in networks". In: *Physical review letters* 89.20 (2002), p. 208701.

[37] Mark EJ Newman. “Mixing patterns in networks”. In: *Physical review E* 67.2 (2003), p. 026126.

[38] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. “Random graphs with arbitrary degree distributions and their applications”. In: *Physical review E* 64.2 (2001), p. 026118.

[39] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. “Intensity and coherence of motifs in weighted complex networks”. In: *Physical Review E* 71.6 (2005), p. 065103.

[40] Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and Arun K Majumdar. “Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning”. In: *Information Fusion* 10.4 (2009), pp. 354–363.

[41] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. “The synthetic data vault”. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2016, pp. 399–410.

[42] Luca Rendsburg, Holger Heidrich, and Ulrike Von Luxburg. “NetGAN without GAN: From random walks to low-rank approximations”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8073–8082.

[43] Giulio Rossetti, Letizia Milli, and Rémy Cazabet. “CDLIB: a python library to extract, compare and evaluate communities from complex networks”. In: *Applied Network Science* 4.1 (2019), pp. 1–26.

[44] Marc Sabek and Uta Pigorsch. “Local assortativity in weighted and directed complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 630 (2023), p. 129231.

[45] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. “Generalizations of the clustering coefficient to weighted complex networks”. In: *Physical Review E* 75.2 (2007), p. 027105.

[46] Akrati Saxena. “A survey of evolving models for weighted complex networks based on their dynamics and evolution”. In: *arXiv preprint arXiv:2012.08166* (2020).

[47] Akrati Saxena, Yulong Pei, Jan Veldsink, Werner van Ipenburg, George Fletcher, and Mykola Pechenizkiy. “The banking transactions dataset and its comparative analysis with scale-free networks”. In: *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2021, pp. 283–296.

[48] David W Shanafelt, KR Salau, and JA Baggio. “Do-it-yourself networks: a novel method of generating weighted networks”. In: *Royal Society Open Science* 4.11 (2017), p. 171227.

[49] Akila Somasundaram and U Srinivasulu Reddy. “Data imbalance: effects and solutions for classification of large and highly imbalanced data”. In: *international conference on research in engineering, computers and technology (ICRECT 2016)*. 2016, pp. 1–16.

[50] Toyotaro Suzumura and Hiroki Kanezashi. *Anti-Money Laundering Datasets: InPlusLab anti-money laundering datadatasets*. 2021.

[51] Xiao Tian, Mahashweta Das, and Chiranjeet Chetia. “Generating Large-Scale Synthetic Payment Graphs with Realistic Fraudulent and Money Laundering Patterns”. In: (2021).

[52] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific reports* 9.1 (2019), p. 5233.

[53] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[54] Wen-Xu Wang, Bo Hu, Tao Zhou, Bing-Hong Wang, and Yan-Bo Xie. “Mutual selection model for weighted networks”. In: *Physical Review E* 72.4 (2005), p. 046140.

[55] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. “Scalable graph learning for anti-money laundering: A first look”. In: *arXiv preprint arXiv:1812.00076* (2018).

[56] Rong Yang, Leyla Zhuhadar, and Olfa Nasraoui. “Bow-tie decomposition in directed graphs”. In: *14th International Conference on Information Fusion*. IEEE. 2011, pp. 1–5.

[57] Dongfang Zhang, Basu Bhandari, Dennis Black, et al. “Credit card fraud detection using weighted support vector machine”. In: *Applied Mathematics* 11.12 (2020), p. 1275.

[58] Zhongzhi Zhang, Lili Rong, Bing Wang, Shuigeng Zhou, and Jihong Guan. “Local-world evolving networks with tunable clustering”. In: *Physica A: Statistical Mechanics and its Applications* 380 (2007), pp. 639–650.

[59] Dafang Zheng, Steffen Trimper, Bo Zheng, and PM Hui. “Weighted scale-free networks with stochastic weight assignments”. In: *Physical Review E* 67.4 (2003), p. 040102.

9 Appendix

d_{in}	d_{out}	$\boldsymbol{N}$	p	**CC**	**Assort.**	**Power-law in**	**Power-law out**	**Density**
2	5	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 5\end{bmatrix}$	0.7	0.048 ± 0.005	-0.121 ± 0.007	2.631 ± 0.545	3.311 ± 0.049	0.000621432 ± 1.34E-5
2	5	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 1\end{bmatrix}$	0.7	0.020 ± 0.002	-0.171 ± 0.012	4.617 ± 0.736	2.796 ± 0.036	0.000638583 ± 1.31E-5
5	10	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 5\end{bmatrix}$	0.7	0.049 ± 0.004	-0.174 ± 0.010	2.477 ± 0.734	2.858 ± 0.091	0.000924358 ± 9.27E-5
5	10	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 1\end{bmatrix}$	0.7	0.031 ± 0.003	-0.200 ± 0.010	3.472 ± 2.186	2.557 ± 0.085	0.000870968 ± 1.07E-4
2	5	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 5\end{bmatrix}$	0.8	0.044 ± 0.006	-0.112 ± 0.010	2.354 ± 0.229	3.375 ± 0.096	0.000395355 ± 8.39E-6
2	5	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 1\end{bmatrix}$	0.8	0.015 ± 0.002	-0.165 ± 0.012	5.166 ± 1.556	2.861 ± 0.107	0.000411186 ± 7.15E-6
5	10	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 1 & 5\end{bmatrix}$	0.8	0.041 ± 0.003	-0.140 ± 0.011	2.392 ± 0.383	3.014 ± 0.096	0.000567444 ± 4.18E-5
5	10	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 1\end{bmatrix}$	0.8	0.021 ± 0.002	-0.183 ± 0.010	5.269 ± 1.795	2.674 ± 0.028	0.000545256 ± 5.37E-5
2	2	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 5\end{bmatrix}$	0.9	0.014 ± 0.001	-0.090 ± 0.010	3.505 ± 0.461	4.309 ± 0.961	0.000249084 ± 3.01E-6
2	2	$\begin{bmatrix}0 & 1\\ 1 & 0\\ 5 & 5\end{bmatrix}$	0.8	0.013 ± 0.002	-0.092 ± 0.012	3.124 ± 0.577	3.023 ± 0.263	0.000351121 ± 5.06E10

Table 17: Data table for parameter tuning for use-case