Reciprocation in scale-invariant network models

Claus, Benjamin

Reciprocation in Scale-Invariant Network Models

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author : B.W.A. Claus
Student ID : [redacted]
Supervisor : Dr. D. Garlaschelli
Second corrector : Dr. F. Jansen

Leiden, The Netherlands, June 10, 2024

Reciprocation in Scale-Invariant Network Models

B.W.A. Claus

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

June 10, 2024

Abstract

In this research, a recently proposed renormalization group approach for networks to the case of random directed graphs is being generalized: we present a scale-invariant description of directed networks containing reciprocated edges. This allows us to neglect several strong assumptions that are currently necessary to renormalize directed networks such as financial transaction networks. As an application, a model of ING's transaction data has been derived across multiple coarse-grained partitions. In this article we provide detailed information on how this particular model has been structured and how its parameters are obtained. We show how we can use this model to determine the expected cumulative degree and weight distributions of ING's transaction network across multiple coarse-grained partitions of the network which we will compare to the empirical degree and weight distributions, respectively.

Contents

Version of June 10, 2024– Created June 10, 2024 - 13:16

Chapter 1

Introduction

Our modern society is driven by complex networks: transportation networks, communication networks, social networks, and of course economic networks. The actions of one element in one of these networks can have a ripple effect which in turn can causes other elements in that network to take a specific form of action as well. Everything is connected. The connectedness of these complex networks is the very reason why our modern society has evolved to be as powerful as we have never experienced before: global communication is faster and easier than ever before, the World-Wide-Web is growing at a rate that no one could ever have dreamed of, the speed with which people and goods travel around the globe would have been fictional only a few decades ago. Unfortunately, this connectedness also makes us vulnerable. A financial crisis or a contagious disease can have the entire world on its knees in a matter of days. It is therefore of no surprise that the study of complex networks has become of much interest for computer scientists, biologists, mathematicians and physicists.

In this thesis, we focus our attention on the complex networked system that is the economic network of our modern society. More precisely, we investigate several behavioural properties of the financial transaction network between firms and organizations. The data, not accessible to the public, is provided by the largest bank of the Netherlands: ING Bank N.V. [1]. Given the limited availability of firm-to-firm financial transaction data, it is ever so important to develop a mathematical framework that is capable of reproducing multiple structural properties of this type of networks given nothing more but partial information [2]. Scale-invariant network models have been proposed in the past in order to fulfill this purpose. It has become evident that the analyses of a complex system

at different scales is crucial in understanding certain aspects of the underlying physical system [3]. Garuccio, Lalli and Garlaschelli proposed a model that "remains invariant across all scales, for any desired (horizontal or vertical) partition of nodes into block-nodes" to describe the International Trade Network [4]. In a paper by Di Vece, Pijpers and Garlaschelli the role of reciprocated edges in the Dutch industry-industry network (as seen by the Dutch central bureau of statistics [5]) has been investigated with the use of a Directed Binary Configuration Model and a Reciprocal Binary Configuration Model [6]. In another paper, by Garlaschelli and Loffredo, the fundamentals of link reciprocity has been investigated. In this paper, Garlaschelli and Loffredo determine whether reciprocated links between vertex pairs occur in empirical networks as often as expected by chance. This has been investigated for multiple types of real world networks among which the World Trade Web and the World Wide Web [7].

Here we propose a directed scale-invariant, reciprocity controlling network model for an inter-firm financial transaction network. Scale-invariant in this context means that the model should be capable of describing the transaction network on multiple scales. Coarse-graining the transaction network brings us to a higher network scales which can be considered as an inter-(sub)sector financial transaction network when coarse-graining operates on the so called NAICS-codes of the firms which will indeed be the case in this research (more on this in chapter 3). Another example is to coarse-grain on the firm locations which yields a financial transaction network between cities. After having coarse-grained the network it is by construction more likely for the resulting network to be more dense. To properly describe such dense higher scaled networks we need our model to be able to control reciprocity. Reciprocity controlling for a network model means that it accurately describes the probability that a reciprocated connection exists, i.e. a link from A to B *and* from B to A *simultaneously* exist.

Since detailed financial transaction data remains to be confidential, it is nonetheless useful for banks to share information on the structure of their networks between them. This information transparency between banks allows for easier monitoring of the health of the economy. Our proposed directed scale-invariant, reciprocity controlling network model for inter-firm networks can possibly be used as a means of communication. With this, banks only need to communicate the network parameters between each other which are not directly dependent on private information of the banks of their clients.

In the next chapter we primarily discuss the mathematics of our proposed network model. We derive the model from the ground up and show how the Hamiltonian and the partition function behave. Chapter 3 describes the data as provided by ING and chapter 4 explains what methods are used to ”fit” the network model to the data. In chapter 5 the results are given and discussed. Finally, a summary and conclusion is given in chapter 6. Throughout the chapters we often refer to the appendix (see chapter 7) for additional information on a specific subject.

Chapter 2

Directed Scale-Invariant Model

In this chapter we focus on the derivation of the proposed directed scale-invariant network model that accounts for reciprocated edges in economic transaction networks. In order to do so, we first briefly take a look at the nature of the data in order to explain the notion of scale-invariance (the data will be described in more detail in chapter 3). With this information we can derive the probability distributions for financial transactions between firms to occur which can in turn provide us with a probability function for a network to exist with a given adjacency matrix $\mathbf{A}^l$ on network scale l. Given this probability function we can determine several other useful functions such as the likelihood function, the Hessian matrix, the Hamiltonian and the partition function.

2.1 Scale Invariance

For the purpose of this research, we assume that transactions and edge weights are an additive variable. This axiom allows us to derive a scale-invariant probability distribution for the existence of an edge between a pair of nodes due to the fact that we can treat clusters of nodes the same as how we treat individual nodes.

Definition: We say that a collection of nodes I connects to another collection of nodes J, if any of the nodes in I connects to any of the nodes in J. The probability of this is thus one minus the probability that none of the nodes in I connect to any of the nodes in J.

Definition: The in-strength of a node i is the sum of the weights of all

the edges that are directed to node i, while the out-strength of a node i is the sum of the weights of all the edges that are directed from node i.

Suppose $p_{i\to j}$ is the probability that a node i connects to a node j. We assume that this probability depends on four variables: the in- and out-strength of node i, and the in- and out-strength of node j. If $I = i_1, \cdots i_n$ is a collection of nodes and $J = j_1, \cdots j_m$ is another collection of nodes, then by the above definition

$$p_{I\to J}(x_I, y_I; x_J, y_J) = 1 - \prod_{i\in I}\prod_{j\in J}(1 - p_{i\to j}(x_I, y_I; x_J, y_J)). \tag{2.1}$$

For convenience define $l_{i\to j}(x_i, y_i; x_j, y_j) \equiv \log(1 - p_{i\to j}(x_i, y_i; x_j, y_j))$. Then, equation 2.1 can be rewritten:

$$l_{I\to J}(x_I, y_I; x_J, y_J) = \sum_{i\in I}\sum_{j\in J} l_{i\to j}(x_i, y_i; x_j, y_j) \tag{2.2}$$

where x_k and y_k denote the in- and out-strengths, respectively, of node k. Using the notation from equation 2.2, it is easy to show that we will have scale-invariance if

$$l\left(\sum_{i\in I} x_i, \sum_{i\in I} y_i; \sum_{j\in J} x_i, \sum_{j\in J} y_j\right) = \sum_{i\in I}\sum_{j\in J} l\left(x_i, y_i; x_j, y_j\right) \tag{2.3}$$

which is in line with the above definition.

In what follows, we denote by $\frac{\partial}{\partial a_x}$ the derivative with respect to the first argument (the in-strength of node i on the right hand side, the in-strength of node I on the left hand side) of l, by $\frac{\partial}{\partial a_y}$ the derivative with respect to the second argument (the out-strength of node i on the right hand side, the out-strength of node I on the left hand side), by $\frac{\partial}{\partial b_x}$ the derivative with respect to the third argument (the in-strength of node j on the right hand side, the in-strength of node J on the left hand side), and by $\frac{\partial}{\partial b_y}$ the derivative with respect to the fourth argument(the out-strength of node j on the right hand side, the out-strength of node J on the left hand side).

For two different nodes α and β from node collection I we get equa-

tions:

$$\frac{\partial^2 l}{\partial a_x^2}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$
$$\frac{\partial^2 l}{\partial a_y^2}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$
$$\frac{\partial^2 l}{\partial a_x \partial a_y}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$

So that

$$l(x_\alpha, y_\alpha; x_\beta, y_\beta) = f(x_\beta, y_\beta)x_\alpha + g(x_\beta, y_\beta)y_\alpha + h(x_\beta, y_\beta) \tag{2.4}$$

Similarly for two different nodes α and β from collection J we get equations:

$$\frac{\partial^2 l}{\partial b_x^2}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$
$$\frac{\partial^2 l}{\partial b_y^2}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$
$$\frac{\partial^2 l}{\partial b_x \partial b_y}(x_\alpha, y_\alpha; x_\beta, y_\beta) = 0$$

So that

$$l(x_\alpha, y_\alpha; x_\beta, y_\beta) = k(x_\alpha, y_\alpha)x_\beta + r(x_\alpha, y_\alpha)y_\beta + s(x_\alpha, y_\alpha) \tag{2.5}$$

Equations 2.4 and 2.5 can only be true simultaneously if $l(x_\alpha, y_\alpha; x_\beta, y_\beta) = -\delta_{xx}x_\alpha x_\beta - \delta_{xy}x_\alpha y_\beta - \delta_{yx}y_\alpha x_\beta - \delta_{yy}y_\alpha y_\beta$ where the deltas are constants. This expression can be conveniently written by collecting the deltas in a constant matrix D such that $l(x_\alpha, y_\alpha; x_\beta, y_\beta) = -(x_\alpha, y_\alpha)^T \cdot D \cdot (x_\beta, y_\beta)$. From this, we obtain our scale-invariant network model: the probability that a node i connects to a node j is given by

$$p_{i\to j}(x_i, y_i; x_j, y_j) = 1 - e^{-(x_i, y_i)^T \cdot D \cdot (x_j, y_j)} \tag{2.6}$$

where x_k and y_k denote the in- and out-strengths, respectively, of node k. In what follows, the in- and out-strengths of a node will be represented by a 2-dimensional vector (e.g.: $\vec{x_k} = (x_k, y_k)^T$)

Note that above, nothing has been stated about the probability that node j connects to node i. Thus far, we can therefore only say anything conclusively about the probability that node i connects to node j. In other words, the probability $p_{i\to j}$ that node i connects to node j, and the probability $p_{i\leftarrow j}$ that node j connects to node i are independent (note that they are however correlated due to their dependence on the same strengths).

In what follows, we construct a scale-invariant network model that has control over reciprocity by making the probabilities $p_{i\to j}$ and $p_{i\leftarrow j}$ dependent of each other [8]. To do this, we need to define four probability distributions:

1. p_{ij}^{11}: the probability that both node i connects to node j, and node j connects to node i. Here, $i = j$ is allowed meaning that a self-loop exists.

2. p_{ij}^{10}: the probability that node i connects to node j, but node j does not connect to node i. Here, $i = j$ is not allowed.

3. p_{ij}^{01}: the probability that node i does not connect to node j, but node j does connect to node i. Here, $i = j$ is not allowed.

4. p_{ij}^{00}: the probability that neither node i connects to node j, nor node j connects to node i. Here, $i = j$ is again allowed meaning that a self-loop does not exists.

Now if we have a node i and a node k ($i \neq k$), and we want to know if the collection of the two $I = \{i, k\}$ connects to a third node j, then this is given by

$$p_{Ij}^{00} = p_{ij}^{00} p_{kj}^{00} \tag{2.7}$$

$$p_{Ij}^{10} = p_{ij}^{10} p_{kj}^{10} + p_{ij}^{10} p_{kj}^{00} + p_{ij}^{00} p_{kj}^{10} \tag{2.8}$$

$$p_{Ij}^{01} = p_{ij}^{01} p_{kj}^{01} + p_{ij}^{01} p_{kj}^{00} + p_{ij}^{00} p_{kj}^{01} \tag{2.9}$$

$$p_{Ij}^{11} = \text{ all other combinations } = 1 - p_{Ij}^{00} - p_{Ij}^{10} - p_{Ij}^{01} \tag{2.10}$$

We will use the above equations to determine the functional forms of the quantities p_{Ij}^{00}, p_{Ij}^{01}, p_{Ij}^{10} and p_{Ij}^{11}. The results are given below while the

derivations are given in the appendix (7.1).

$$p_{ij}^{00} = \begin{cases} e^{-\vec{x}_i \cdot M \cdot \vec{x}_j}, & \text{if } i \neq j \\ e^{-\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_i - \vec{u}^T \cdot \vec{x}_i}, & \text{if } i = j \end{cases} \tag{2.11}$$

$$p_{ij}^{01} = \begin{cases} e^{-\vec{x}_i \cdot M \cdot \vec{x}_j} \left(e^{\vec{x}_i \cdot A \cdot \vec{x}_j} - 1 \right), & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{2.12}$$

$$p_{ij}^{10} = \begin{cases} e^{-\vec{x}_i \cdot M \cdot \vec{x}_j} \left(e^{\vec{x}_i \cdot A^T \cdot \vec{x}_j} - 1 \right), & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{2.13}$$

$$p_{ij}^{11} = \begin{cases} 1 + e^{-\vec{x}_i \cdot M \cdot \vec{x}_j} \left(1 - e^{\vec{x}_i \cdot A \cdot \vec{x}_j} - e^{\vec{x}_i \cdot A^T \cdot \vec{x}_j} \right), & \text{if } i \neq j \\ 1 - e^{-\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_i - \vec{u}^T \cdot \vec{x}_i}, & \text{if } i = j \end{cases} \tag{2.14}$$

with $\vec{x}_i = (x_i, y_i)^T = (\text{in-strength}_i, \text{out-strength}_i)^T$. Also, the probabilities need to be normalized so that $p_{ij}^{00} + p_{ij}^{01} + p_{ij}^{10} + p_{ij}^{11} = 1$. Moreover, the probabilities need to be non-negative and bounded by 1. This leads to a constraint that needs to be imposed on the values of the elements of the matrices M and A, and the vector $\vec{u}$ as given in equation 2.15 (once more the derivation is given in the appendix in section 7.2).

$$\begin{aligned} 0 \leq A + A^T \leq M < \infty \\ 0 \leq \vec{u} < \infty \end{aligned} \tag{2.15}$$

Where $M = M^T$ since $p_{ij}^{00} = p_{ji}^{00}$, $p_{ij}^{11} = p_{ji}^{11}$ and $p_{ij}^{01} = p_{ji}^{10}$ by construction. The practical implementation of these inequalities will be covered in chapter 4.

2.1.1 Renormalization

Above derivations resulted from the postulate that transactions are additive quantities (other examples of additive quantities for financial transactions networks are the number of payments between two nodes and the account balance of a node). The practical application of this is that we can treat clusters of nodes the same as how we treat individual nodes. This means that the strength of a cluster of nodes I is given by $\vec{x}_I = \sum_{i \in I} \vec{x}_i$. This (deterministic) partition, which we will denote by the letter Ω, allows us to evaluate the network at different scales. The adjacency matrix of the network observed at a specific scale will be denoted by A^l where l

denotes the scale of the observed network. Let $P_l(A^l)$ be a random process that generates a network on scale l as function of the adjacency matrix A^l. Since the partition Ω is fixed, $P_0(A^0)$ induces a random process at higher network scales as given by [8]:

$$P_l(A^l) = \sum_{\{A^0\} \xrightarrow{\Omega_{l\dots\Omega_0}} A^l} P_0(A^0). \tag{2.16}$$

Given equations 2.11, 2.12, 2.13 and 2.14 we obtain the following for $P_l(A^l)$ the following:

$$P_l(A^l) = \prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} \left(\overrightarrow{p_{i_l,j_l}}\right)^{\overrightarrow{a_{i_l,j_l}}} \left(\overleftarrow{p_{i_l,j_l}}\right)^{\overleftarrow{a_{i_l,j_l}}} \left(\overleftrightarrow{p_{i_l,j_l}}\right)^{\overleftrightarrow{a_{i_l,j_l}}} \left(\overline{p_{i_l,j_l}}\right)^{\overline{a_{i_l,j_l}}} \tag{2.17}$$

where we have adopted the following notation $\overrightarrow{p_{i_l,j_l}} = p^{10}_{i_l,j_l}$, $\overleftarrow{p_{i_l,j_l}} = p^{01}_{i_l,j_l}$, $\overleftrightarrow{p_{i_l,j_l}} = p^{11}_{i_l,j_l}$ and $\overline{p_{i_l,j_l}} = p^{00}_{i_l,j_l}$ for the probability functions. In what follows we will use both notations interchangeably depending on the context. We can rewrite this equation

$$\begin{aligned}
P_l(A^l) &= \left(\prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} \overline{p_{i_l,j_l}}\right) \cdot \left(\prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} \left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} \left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} \left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}}\right) \\
&= \left(\prod_{i_l=1}^{N_l} \overline{p_{i_l,i_l}} \prod_{j_l=1}^{i_l-1} \overline{p_{i_l,j_l}}\right) \cdot \left(\prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} \left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} \left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} \left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}}\right) \\
&= Q \left(\prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} \left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} \left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} \left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}}\right)
\end{aligned} \tag{2.18}$$

where the factor Q is a scale independent constant since it does not depend on the adjacency matrix A^l. We can explicitly calculate Q by substituting $\overline{p_{i_l,i_l}} = e^{-\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_i - \vec{u}^T \cdot \vec{x}_i}$ and $\overline{p_{i_l,j_l}} = e^{-\vec{x}_i \cdot M \cdot \vec{x}_j}$.

$$\begin{aligned}
Q &= \left(\prod_{i_l=1}^{N_l} e^{-\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_i - \vec{u}^T \cdot \vec{x}_i} \prod_{j_l=1}^{i_l-1} e^{-\vec{x}_i \cdot M \cdot \vec{x}_j}\right) \\
&= e^{-\sum_{i_l=1}^{N_l}\left(\vec{u}^T \cdot \vec{x}_i\right) - \sum_{i_l=1}^{N_l}\left(\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_i + \sum_{j_l=1}^{i_l-1} \vec{x}_i \cdot M \cdot \vec{x}_j\right)} \\
&= e^{-\sum_{i_l=1}^{N_l}\left(\vec{u}^T \cdot \vec{x}_i\right) - \sum_{i_l=1}^{N_l}\sum_{j_l=1}^{N_l}\left(\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_j\right)} \\
&= e^{-\frac{1}{2}\sum_{i_l=1}^{N_l}\sum_{j_l=1}^{N_l}\left(\vec{x}_i \cdot M \cdot \vec{x}_j\right) - \vec{u}^T \sum_{i_l=1}^{N_l} \vec{x}_i}
\end{aligned} \tag{2.19}$$

where the sums in the exponent run over all nodes in the network. Above we have stated that $\vec{x_I} = \sum_{i \in I} \vec{x_i}$ for a cluster of nodes I. Similarly for the entire network we thus obtain $\vec{x_{i_\infty}} = \sum_{i_l=1}^{N_l} \vec{x_{i_l}}$ where we define $\vec{x_{i_\infty}}$ as the two-dimensional vector representing the in- and out-strength of the entire network. Thus for Q we obtain

$$\begin{aligned} Q =& e^{-\frac{1}{2}\vec{x_{i_\infty}} \cdot M \cdot \vec{x_{i_\infty}} - \vec{u}^T \cdot \vec{x_{i_\infty}}} \\ =& 1 - \overleftrightarrow{p_{i_\infty, i_\infty}}. \end{aligned} \tag{2.20}$$

We can simplify equation 2.18 further

$$\begin{aligned} P_l(A^l) =& Q \prod_{i_l=1}^{N_l} \prod_{j_l=1}^{i_l} e^{log\left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}}} \\ =& Q e^{\sum_{i_l=1}^{N_l} \sum_{j_l=1}^{i_l} \left[log\left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}} \right]}. \end{aligned} \tag{2.21}$$

Now it is clear that we can write the probability $P_l(A^l)$ as function of the effective Hamiltonian $\mathcal{H}^l_{eff}$ and partition function $\mathcal{Z}^l$:

$$P_l(A^l) = \frac{e^{-\mathcal{H}^l_{eff}}}{\mathcal{Z}^l} \tag{2.22}$$

where for the effective Hamiltonian we have

$$\mathcal{H}^l_{eff} = -\sum_{i_l=1}^{N_l} \sum_{j_l=1}^{i_l} \left[log\left(\frac{\overrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overrightarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftarrow{a_{i_l,j_l}}} + log\left(\frac{\overleftrightarrow{p_{i_l,j_l}}}{\overline{p_{i_l,j_l}}}\right)^{\overleftrightarrow{a_{i_l,j_l}}} \right] \tag{2.23}$$

and for the partition function we have

$$\mathcal{Z}^l = \frac{1}{Q} = \frac{1}{1 - \overleftrightarrow{p_{i_\infty, i_\infty}}}. \tag{2.24}$$

We clearly see that the partition function is scale-invariant. As a result, according to Kadanoff, free energy is thus conserved [9].

Version of June 10, 2024– Created June 10, 2024 - 13:16

Chapter 3

Data

This chapter describes the provided data for this research. We will look into how the entire dataset has been filtered in order to obtain a financial transaction network solely between firms and organizations. We will then analyze the cleaned dataset by investigating the behaviour of (among others) the empirical degree and strength distributions.

3.1 Financial Transactions

The data we are interested in for the purpose of this research consists of financial transaction data between firms and organizations. We therefore discard any data from private individuals. The raw data has been provided by ING's Wholesale banking Advanced Analytics tribe (WBAA) [10]. Since the year 2018, ING stores transaction information to and from all of their clients as a table format. This table provides information on the amount of money being transferred (in euros as well as in the applicable currency) and the time at which the transaction took place. The payers and beneficiaries of the transactions are listed by their names, international bank account number (IBAN), and their IDs which are unique for every economic entity (however, a large organization can have more than one ID, for example mother and daughter companies have different IDs since they are in principle different entities). Two separate columns in this table tell us whether the payer and/or beneficiary are ING clients, and whether the payers and/or beneficiaries are private individuals (freelancers and one man businesses are also flagged as private individuals).

For non-private individual accounts, sector specific information is re-

quired. This information is listed in a separate data table. This second table contains more detailed information on the account holder. Sector specific information is given by the NAICS-code (North American Industry Classification System) [11]. This is a 6-digit code where the first two digits (count from left to right) specify the sector, the third specifies the subsector, the forth specifies the industry group, the fifth specifies the NAICS industry and the sixth specifies the national industry.

The data provided in the transaction data table can in principle already be seen as a complex network. However in this network, every node pair can have multiple transactions (edges) between them. In addition, since transactions have been registered by the above described method since the year 2018, this network will be extremely large which upon analyzing may result in computational challenges that are beyond the scope of this research. Thus, in this research we analyse the transaction network for solely one year. This makes the size of the data much more manageable while not being effected to seasonal changes in the data. The year in question is 2022 since the data for 2018 is incomplete, 2019 and 2020 were due to the COVID pandemic considered to be unusual years, and data from 2021 (possible still skewed by the aftermath of the pandemic) is obviously less up to date than data from 2022.

3.2 Financial Transaction Network

In the transaction data from 2022, nodes can still be linked to each other via multiple edges. By grouping the IDs together and summing the number of transactions and the amount of money that is being transferred in each transaction we obtain a graph in which every pair of nodes is connected by at most one edge. Next, we remove all private individuals from this network as well as all accounts that are not ING clients. This gives us a closed transaction network of firms/organizations (closed meaning that we do not consider transactions from/to accounts that are not from ING clients. We refer to these accounts as the rest of the world (ROTW) which, for convenience, we have clustered to form one large node). Finally we need to filter out accounts from public administrations (which include the Dutch tax office) and accounts from financial institutions. The NAICS-codes of these accounts start with the numbers 92 and 52, respectively. It is assumed that these accounts exhibit ”unusual” transaction behaviour (they execute transactions very frequently while the amount of money be-

ing transferred is often relatively large) in comparison to the vast majority of accounts from firms/organizations. Keep in mind that in this construction an edge is allowed to start and end at the same vertex. This often occurs for large companies that have multiple accounts. If they manage their cash over these accounts, self-loops are created. Thus we do not end up with a simple graph.

The result is a (very) sparse network of financial transactions consisting of 323880 vertices with $2.6 \cdot 10^6$ edges. The maximum allowed number of edges for a network with 323880 nodes is approximately equal to $\frac{N^2}{2} \approx 52 \cdot 10^9$. From these edges, approximately $3.2 \cdot 10^4$ are self-loop edges (edges that start and end at the same vertex), while $5.6 \cdot 10^5$ are reciprocated links (a connection/link/edge from node A to node B is said to be reciprocated if a connection/link/edge exists in the opposite direction as well).

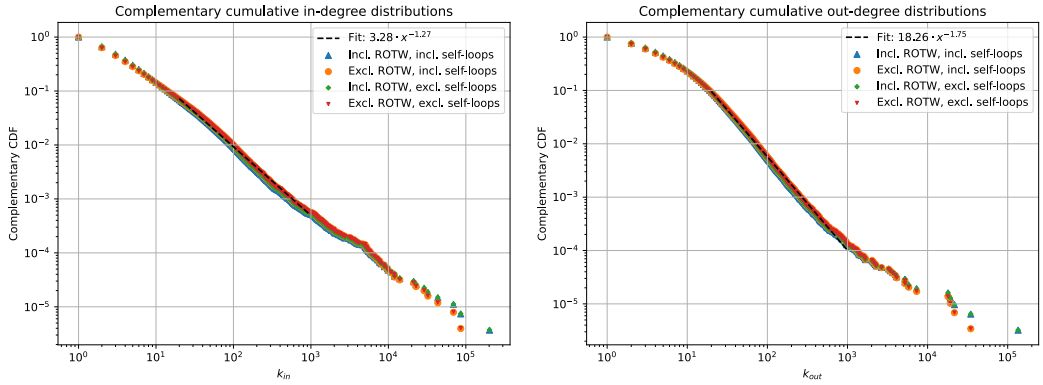
In figure 3.1 the complementary cumulative in-degree and out-degree distributions have been plotted to give a more detailed description of some of the properties of the network. We calculate the in-degree and out-degrees (k_i^{in}, k_i^{out}) of a certain node i by summing the entries of the adjacency matrix:

$$k_i^{in} = \sum_{i,j,(i\neq j)} a_{i\leftarrow j} \qquad\qquad k_i^{out} = \sum_{i,j,(i\neq j)} a_{i\rightarrow j}. \tag{3.1}$$

The complementary cumulative distributions are then given by the inverse cumulative sum of k_i^{in} and k_i^{out} for all nodes i in the transaction network. When calculating the degree distributions, we first distinguished four types of adjacency matrices:

- Including connections to and from the ROTW-node and including self-loop connections;
- Excluding connections to and from the ROTW-node and including self-loop connections;
- Including connections to and from the ROTW-node and excluding self-loop connections;
- Excluding connections to and from the ROTW-node and excluding self-loop connections.

In figure 3.1 it can be clearly seen on the log-log scale, that the complementary cumulative degree distributions follow a near power law distribution as illustrated by the black dashed lines. The tails however do not precisely obey to this power law distribution ($3.28x^{-1.27}$ for the in-degree distribution and $18.26x^{-1.75}$ for the out-degree distribution). Of course it is to be expected that the four different types of distributions for both k_i^{in} and k_i^{out} are nearly identical to each other since the difference in degrees can be at most two: consider two different nodes a and b. Node a is connected to the ROTW-node while node b is not which is the only key difference between the degrees of these nodes. It holds that $k_a = k_b + 1$. Similarly, for two other unequal nodes c and d where node c has a connection with itself while node d does not, $k_c = k_d + 1$.



(a) *Complementary cumulative in-degree* $\left(k_i^{in} = \sum_{i,j,(i \neq j)} a_{i \leftarrow j}\right)$ *distribution.*

(b) *Complementary cumulative out-degree* $\left(k_i^{out} = \sum_{i,j,(i \neq j)} a_{i \rightarrow j}\right)$ *distribution.*

Figure 3.1: *Complementary cumulative in-degree (left) and out-degree (right) distributions for financial transaction networks including the rest-of-the-world node (ROTW) and self-loop connections (blue triangles), excluding ROTW-node and including self-loops (orange dot), including ROTW-node and excluding self-loops (green diamante), and excluding ROTW-node and self-loops (red upside down triangle). On these distributions a power law distribution has been fitted which is visualized by a black dashed line. Node that both the x and y-axis are log-scaled.*

Through every edge, a certain amount of money is being transferred. This amount is called the weight of an edge (or weight for short). The weight of an edge spanning from node i to node j is denoted by w_{ij}. Now since every edge is an aggregate of transactions from node i to node j in the complete year of 2022, we can also identify a transaction weight. The

complementary cumulative distributions of both the edge weights and the transaction weights are given in figure 3.2. In this figure we see that the weights per edge are approximately two orders of magnitude larger than the weights per transaction. This means that an edge on average consists of order a hundred individual transactions.

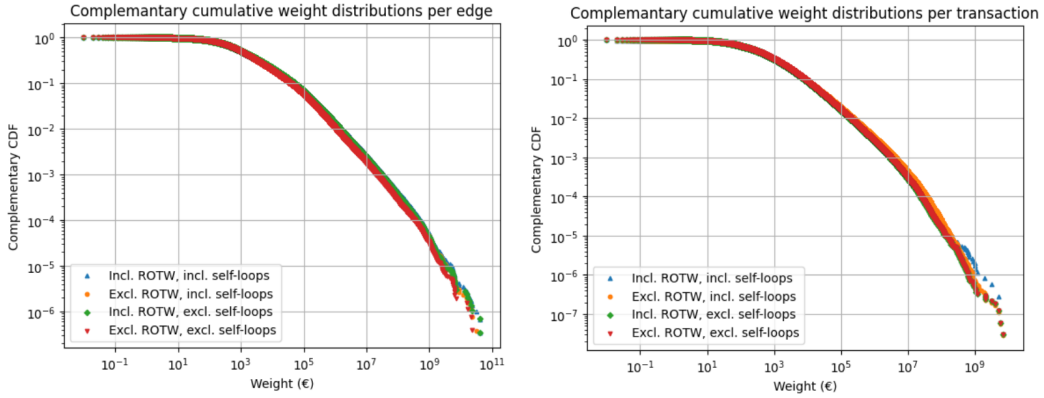


Figure 3.2: *Complementary cumulative edge weight distribution (in euros) (left), and complementary cumulative transaction weight distribution (in euros) (right) for the financial transaction network including the ROTW-node and self-loops (blue triangles), excluding ROTW-node and including self-loops (orange dot), including ROTW-node and excluding self-loops (green diamante), and excluding ROTW-node and self-loops (red upside down triangle) on a double log scale.*

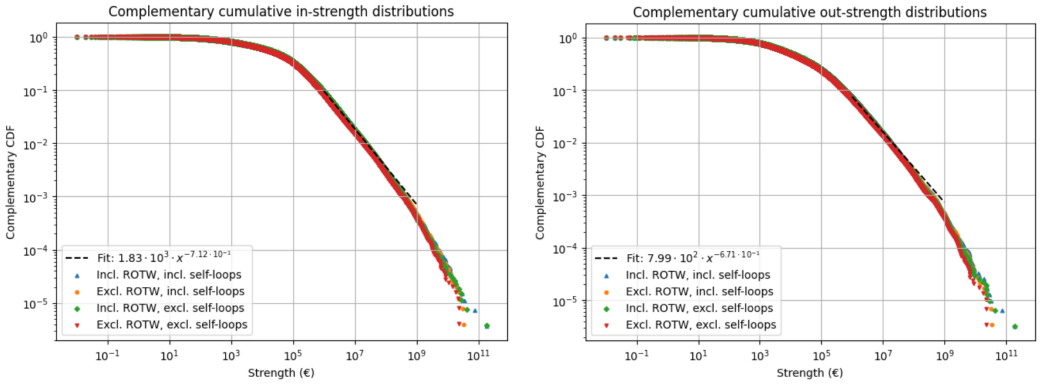
Using these weights we can determine the node strength (or strength for short) which is calculated by

$$s_i^{in} = \sum_{i,j,(i\neq j)} w_{i\leftarrow j} \qquad s_i^{out} = \sum_{i,j,(i\neq j)} w_{i\rightarrow j}. \tag{3.2}$$

Here $w_{i\rightarrow j}$ is the edge weight for an edge from node i to node j. In figure 3.3 the complementary cumulative distributions for the in-strength and the out-strength are plotted. As can be seen in the figure, these distributions exhibit significantly less of a power law distributions in comparison to the complementary cumulative degree distributions. We can see that the vast majority of the nodes have an in- and out-strength of at least $\sim 10^3$ euros. We can conclude this since we see that the distributions start curving at approximately 10^3 euros. In addition, examining the tails of these distributions, we see that there are some outlier nodes that have a strength of well over a billion (10^9) euros. This means that in our financial transaction network there are a handful of firms/organisations that transferred

Version of June 10, 2024– Created June 10, 2024 - 13:16

and/or received in the year 2022 over a billion euros.



(a) *Complementary cumulative in-strength* $\left(s_i^{in} = \sum_{i,j,(i\neq j)} w_{i\leftarrow j}\right)$ *distribution.*

(b) *Complementary cumulative out-strength* $\left(s_i^{out} = \sum_{i,j,(i\neq j)} w_{i\rightarrow j}\right)$ *distribution.*

Figure 3.3: *Complementary cumulative in-strength (left) and out-strength (right) distributions for financial transaction networks including the rest-of-the-world node (ROTW) and self-loop connections (blue triangles), excluding ROTW-node and including self-loops (orange dot), including ROTW-node and excluding self-loops (green diamante), and excluding ROTW-node and self-loops (red upside down triangle). On these distributions an exponential distribution has been fitted which is visualized by a black dashed line. Node that both the x and y-axis are log-scaled.*

To conclude our thorough description of the data at hand, we plot the in- and out-degree distributions versus the in- and out-strength distributions in order to visualize any correlated behaviour between these distributions. These plots have been given in figure 3.4 for a financial network that excludes connections to and/or from the rest-of-the-world node (ROTW-node) and includes self-loop connections. As described above, the model given in chapter 2 will be fitted to this closed network. However, to give an accurate description of the entire data set at hand, similar heatmap plots for networks including both the ROTW-node and self-loops connections, including the ROTW-node and excluding self-loop connections, and excluding both the ROTW-node and self-loop connections are given in the appendix section 7.3, figures 7.2, 7.3 and 7.4 respectively. Note that since empirical degrees are only integer values, the y-axis in figure 3.4 is obviously not continuous.

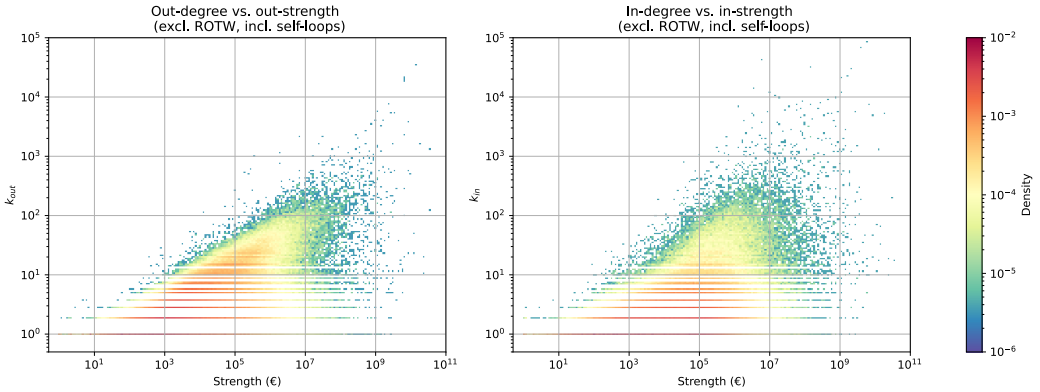


Figure 3.4: *Out-degree vs. out-strength (left) and in-degree vs. in-strength (right) for a financial transaction network excluding connections towards and from the ROTW-node while including self-loop connections. Both plotted on double log scales including a log scale for the density scale.*

Examining figure 3.4, we see that the degree and the strength distributions are rather strongly positively correlated given that the heat-maps are skewed to the right. Therefore we see that the average degree (average over the vertical) increases with strength, and the average strength (average over the horizontal) increases with the degree. This means that a node with a relatively high strength is more likely to have a relatively high degree than a relatively low degree. This is especially clearly visible in the plot showing the out-degree vs. the out-strength. Moreover, again we see that a strength of $\sim 10^3$ is most abundant in this network with a degree of $\sim 10^0$ to $\sim 10^1$.

In appendix section 7.3, information regarding the payment distributions (number of in- and out-payments per node, and the number of payments per edge in figures 7.5 and 7.6) is given as well.

Chapter 4

Method

In the previous chapters we have derived our scale-invariant network model and we have analyzed the data to which the model needs to be fitted. This chapter describes the methods that are used to obtain an algorithm that allows to properly fit the theoretical model to the transaction data.

4.1 Network Generator

Given the fact that the data set described in chapter 3 is significantly large (on the order of 10^5 vertices and 10^6 edges), it would be virtually impossible to implement an accurate fitting algorithm directly to this dataset. During the development of the fitting algorithm it would be beneficial if we can (partially) test our fitting algorithm on a relatively small fictitious dataset of which the network properties are known. We therefore first want to have a method for generating fictitious transaction networks given the node strengths and the values for matrices M and A and vector u from equations 2.11-2.14. The benefit of this is that we know what the results of the fitting algorithm should be and thus we can measure its performance during the development of the algorithm. Moreover, generating a small fictitious network using Google `JAX` improves the computation speed (the Google `JAX` library is also used in the fitting algorithm).

We start off by randomly generating node strengths for $\mathcal{N}$ number of nodes and (randomly) choosing values for M, A and $\vec{u}$ (while obeying eq. 2.15). Now we can explicitly calculate the probability distributions from equations 2.11-2.14 (setting the probability distribution for the self-

loops aside for now) which are in the form of symmetric $\mathcal{N}$-by-$\mathcal{N}$ matrices when we allow all four types of connections to exist between all pairs of nodes. From these symmetric matrices we will take the upper triangle elements to decrease the computation time. We can randomly choose connection types as function of the probability distributions using `jax.random.choice(key=key, a=a, p=P)` with `key` being JAX's pseudo random number generator key, `a = jnp.arange(4)` (because there are four different types of connections apart from the self-loops), and `P` being a concatenated array of the upper triangle elements of the probability distributions (`jnp.concatenate(`$\overline{p_{i,j}}$, $\overrightarrow{p_{i,j}}$, $\overleftarrow{p_{i,j}}$, $\overleftrightarrow{p_{i,j}}$`)`). The result is a $\mathcal{N}$-by-$\mathcal{N}$ "adjacency" matrix (denoted by $\mathcal{A}$) with an empty diagonal (which will be filled later) containing $0s$, $1s$, $2s$ and $3s$. These numbers represent how the nodes are connected to each other. If $\mathcal{A}_{ij} = 0$, than there is no edge between nodes i and j, if $\mathcal{A}_{ij} = 1$ and $\mathcal{A}_{ij} = 2$ than there is one directed edge between a pair of nodes (from i to j and from j to i respectively), if $\mathcal{A}_{ij} = 3$ than there is a reciprocated connection between a pair of nodes.

For randomly generating the self-loop connections we follow practically the same procedure: explicitly calculate the probability that a vertex is connected to itself as function of matrix M, vector u and its strength. Doing this for all $\mathcal{N}$ nodes in our fictitious network we can again randomly choose which self-loop connections are indeed realized using a random Bernoulli distribution (`jax.random.bernoulli(key, p_self)`) with `key` being JAX's pseudo random number generator key and `p_self` an array with length $\mathcal{N}$ containing the probabilities that the nodes are connected to themselves. The result is an array with length $\mathcal{N}$ containing $0s$ and $1s$ representing the diagonal in our "adjacency" matrix $\mathcal{A}$. If matrix element $\mathcal{A}_{ii} = 1$, than node i is connected to itself.

For computational convenience, our fitting algorithm needs to be designed such that it can take ING's financial transaction table as input instead of a complete adjacency matrix. Thus, it would also be convenient to store the information of our fictitious adjacency matrix $\mathcal{A}$ in table format as well. We only need two columns (two separate arrays or lists will suffice) both with length equal to the amount of connections in our fictitious network. Define a *payer*- and a *beneficiary*-column. The payer-column contains all nodes that transfer a certain amount of money while the beneficiary-column contains all nodes that receive a certain amount of money. Node pairs that share a reciprocated connection are listed in both the payer- and beneficiary-column. The same goes for nodes that are connected to themselves.

Now that the connections of our fictitious transaction network are stored in a table format (an example is given in table 4.1), we can build and test our fitting algorithm. We do not need to change the structure of our algorithm when it is being implemented on the real-world transaction data since that is also stored in table format.

Index	Payer	Beneficiary	€	Number of payments
1	0	3	5	2
2	1	0	16	3
3	1	2	9	5
4	1	4	37	10
5	2	1	18	4
6	2	4	8	1
7	3	3	21	3

Table 4.1: *Example of transaction records for a small network containing 5 nodes. In the actual data set, columns containing codes related to the identity of the organizations are provided as well.*

4.2 Parameter Estimation

This paragraph describes the algorithm that is used to fit the reciprocated scale-invariant network model from equations 2.11-2.14 to ING's transaction data. In other words, the goal of this algorithm is to find values for the elements of the 2-by-2 matrices M and A as well as for the elements of the two dimensional vector $\vec{u}$.

4.2.1 Prepare Data

Chapter 3 describes how the raw transaction data is filtered to a useful data set for this research. In this paragraph we look at how the data is used in our fitting algorithm. The data set is formatted as a table of which an example is given in 4.1. In this table we see that a node with label "0" transfers €5 to a node with label "3" within a total of 2 separate payments. Closely analyzing table 4.1 we conclude that node "0" has an in-strength of €16 and an out-strength of €5, node "1" has an in-strength of €18 and an out-strength of €16 + €9 + €37 = €62, etc. Upon extracting the strengths of all nodes from the transaction data, the in- and out-strength are taken

to be in units of €10^9.

Despite the fact that the labels of the nodes in the columns "payer" and "beneficiary" are convenient numbers ranging from 0 to $\mathcal{N} - 1$ with $\mathcal{N}$ being the number of nodes in the network (in the example network from table 4.1 we have $\mathcal{N} = 5$), it remains computationally challenging (when faced with a large transaction network) to identify exactly how the nodes are connected to each other. We will therefore introduce a new labeling system in which we label the connections instead of the nodes themselves (in principle we count their location in the adjacency matrix). We define tail-nodes to be given by $p\mathcal{N} + b$ and head-nodes to be given by $b\mathcal{N} + p$ (where we use the analogy between an arrow and the direction in which money is being transferred). Here, p and b are the standard 0 to $\mathcal{N} - 1$ labels for the payer- and beneficiary-nodes respectively. Using this transformation on the "payer" and "beneficiary" columns of table 4.1, we obtain table 4.2.

Index	Tails ($p\mathcal{N} + b$)	Heads ($b\mathcal{N} + p$)	€	Number of payments
1	3	15	5	2
2	5	1	16	3
3	7	11	9	5
4	9	21	37	10
5	11	7	18	4
5	14	22	8	1
7	18	18	21	3

Table 4.2: *Example of transformed transaction for a small network containing 5 nodes. In the actual data set, columns containing codes related to the identity of the organizations are provided as well.*

Analyzing table 4.2 we can easily identify which node pair is connected via a single directed edge or via a reciprocated edge: for every pair of node-label that is listed in both the tails and heads column, its corresponding nodes are connected via a reciprocated connection. Thus we see that at indexes 3 and 5 labels 11 and 7 are both listed as tails and heads. Their corresponding nodes "1" and "2" (obtained from the inverse of both $p\mathcal{N} + b$ and $b\mathcal{N} + p$) are thus linked via a reciprocated connection. Similarly, every label that is listed in both the tails and heads column on the same index suggests a self-loop connection for that corresponding node. For example, at index 7 we see that label 18 is listed both as a tail and a head which means that its corresponding node (node "3") is connected to itself. Every other pair of nodes is connected via a single directed edge.

4.2.2 Optimization Problem

Using the above described method we know which nodes are connected via a single directed edge or via a reciprocated edge. We also know which nodes are connected to themselves and how much money is being transferred across every edge. We are now ready to determine the parameters from equations 2.11-2.14. There are a number of ways to solve this problem, however we choose to do this by minimizing the negative log-likelihood (NLL) functions. Equation 4.1 gives the NLL-function that is derived from the probability distribution functions for non-self-loop connections (thus $i \neq j$)

$$\begin{aligned} NLL = & \sum_{(i,j,(j>i))\in I^{00}} \vec{x}_i \cdot M \cdot \vec{x}_j \\ & - \sum_{(i,j,(j>i))\in I^{01}} log\left(e^{-\vec{x}_i\cdot M\cdot \vec{x}_j}\left(e^{\vec{x}_i\cdot A\cdot \vec{x}_j}-1\right)\right) \\ & - \sum_{(i,j,(j>i))\in I^{10}} log\left(e^{-\vec{x}_i\cdot M\cdot \vec{x}_j}\left(e^{\vec{x}_i\cdot A^T\cdot \vec{x}_j}-1\right)\right) \\ & - \sum_{(i,j,(j>i))\in I^{11}} log\left(1+e^{-\vec{x}_i\cdot M\cdot \vec{x}_j}\left(1-e^{\vec{x}_i\cdot A\cdot \vec{x}_j}-e^{\vec{x}_i\cdot A^T\cdot \vec{x}_j}\right)\right) \end{aligned} \tag{4.1}$$

while equation 4.2 gives the NLL-function that is derived from the probability distribution functions for self-loop connections (thus $i = j$).

$$\begin{aligned} NLL_{loops} = & \sum_{(i,j,(j=i))\in I^{00}} \frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_j - \vec{u}^T \cdot \vec{x}_i \\ & - \sum_{(i,j,(j=i))\in I^{11}} log\left(1-e^{-\frac{1}{2}\vec{x}_i\cdot M\cdot \vec{x}_j-\vec{u}^T\cdot \vec{x}_i}\right) \end{aligned} \tag{4.2}$$

Where the sums run over the node pair collections I^{00}, I^{01}, I^{10} and I^{11}. The pair i, j lives in I^{00} if and only if there is no connection between i and j, if and only if j connects to i, the pair lives in I^{01}, and in I^{10} if and only if i connects to j, lastly, the pair i, j lives in I^{11} if and only if i connects to j *and* j connects to i.

We use gradient descent (with the `adam` optimizer and a learning rate of 0.001) to minimize the NLL from equation 4.1. This give us values for the elements in matrices M and A. With the found matrix M held fixed we use gradient descent again (also with the `adam` optimizer, now with a

learning rate of 0.005) to minimize NLL_{loops} from equation 4.2 from which we retrieve the vector $\vec{u}$. Both these gradient descent methods terminate after having completed 4500 iterations (more on this in the next section). The choice for separating these two NLL-functions instead of defining one NLL-function that can also account for self-loops, is a result of the fact that self-loop connections are scarce in ING's transaction network (from a total of $\sim 2.6 \cdot 10^6$ edges, $\sim 3.2 \cdot 10^4$ self-loops). This would mean that the additional presence or absence of a small number of self-loops can have a larger impact on the estimation of matrices M and A in comparison to the additional presence or absence of the same amount of non-self-loop edges. Moreover, sending money to oneself is logically done for very different reasons than sending money to others. Therefore we assume that self-loop connections obey to different physics compared to non-self-loop connections. Since the probability distributions of these types of connections both depend on the matrix M, the new physics introduced by the self-loops can therefore "pull" on the true values for M if we were to define one NLL-functions accounting for all types of connections.

Since the in- and out-strengths of the nodes are taken to be in units of €10^9 (as described in the previous section), the outcomes of the dot-products in the exponents of equations 4.1 and 4.2 can become very small. In other words, this means that the combination $\vec{x}^T \cdot M \cdot \vec{x}$ can become very small which makes $e^{\vec{x}^T \cdot M \cdot \vec{x}}$ numerically unstable. To make sure that this can computationally still have meaning, we use the `jnp.expm1` function from `JAX`. This function allows us to calculate $e^x - 1$ for (very) small values of x. However, to utilize this function we have to rewrite equations 4.1 and 4.2 to "expose" all $e^x - 1$ terms. This leads to:

$$
\begin{aligned}
NLL = & \sum_{(i,j,(j>i))\in I^{00}} \vec{x}_i \cdot M \cdot \vec{x}_j \\
& + \sum_{(i,j,(j>i))\in I^{01}} \vec{x}_i \cdot M \cdot \vec{x}_j - \sum_{(i,j,(j>i))\in I^{01}} log\left(e^{\vec{x}_i \cdot A \cdot \vec{x}_j} - 1\right) \\
& + \sum_{(i,j,(j>i))\in I^{10}} \vec{x}_i \cdot M \cdot \vec{x}_j - \sum_{(i,j,(j>i))\in I^{10}} log\left(e^{\vec{x}_i \cdot A^T \cdot \vec{x}_j} - 1\right) \\
& - \sum_{(i,j,(j>i))\in I^{11}} log\left(1 + e^{-\vec{x}_i \cdot M \cdot \vec{x}_j} - e^{\vec{x}_i \cdot (A-M) \cdot \vec{x}_j} - e^{\vec{x}_i \cdot (A^T-M) \cdot \vec{x}_j}\right) \\
= & \sum_{(i,j,(j>i))\in I^{00}, I^{01}, I^{10}} \vec{x}_i \cdot M \cdot \vec{x}_j \\
& - \sum_{(i,j,(j>i))\in I^{01}} log\left(e^{\vec{x}_i \cdot A \cdot \vec{x}_j} - 1\right) - \sum_{(i,j,(j>i))\in I^{10}} log\left(e^{\vec{x}_i \cdot A^T \cdot \vec{x}_j} - 1\right) \\
& - \sum_{(i,j,(j>i))\in I^{11}} log\Bigg(e^{\vec{x}_i \cdot (A+A^T-M) \cdot \vec{x}_j} \cdot \left(e^{\vec{x}_i \cdot A \cdot \vec{x}_j} - 1\right)\left(e^{\vec{x}_i \cdot A^T \cdot \vec{x}_j} - 1\right) \\
& \qquad - \left(e^{\vec{x}_i \cdot (A+A^T-M) \cdot \vec{x}_j} - 1\right)\Bigg)
\end{aligned}
\tag{4.3}
$$

while equation 4.2 becomes

$$
\begin{aligned}
NLL_{loops} = & \sum_{(i,j,(j=i))\in I^{00}} \frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_j - \vec{u}^T \cdot \vec{x}_i \\
& - \sum_{(i,j,(j=i))\in I^{11}} log\left(-\left(e^{-\frac{1}{2}\vec{x}_i \cdot M \cdot \vec{x}_j - \vec{u}^T \cdot \vec{x}_i} - 1\right)\right).
\end{aligned}
\tag{4.4}
$$

For all $e^x - 1$ terms in the above functions for NLL we can use `jnp.expm1` for a correct calculation in the small number regime.

In equations 4.3 and 4.4 we are not directly able to obey the boundary conditions for M, A and $\vec{u}$ as given in equation 2.15. We can only do this after having performed the following substitutions:

$$
M = \begin{pmatrix} m_{00}^2 & m_{01}^2 \\ m_{10}^2 & m_{11}^2 \end{pmatrix} \tag{4.5}
$$

where m_{kl} (with $k,\ l \in \{0, 1\}$ and $m_{01} = m_{10}$ as discussed in chapter 2) can be any number in $\mathbb{R}$. This substitution ensures that $0 \leq M < \infty$. Similarly

for vector $\vec{u}$ we take

$$\vec{u} = \begin{pmatrix} u_0^2 \\ u_1^2 \end{pmatrix} \tag{4.6}$$

where also u_k (with $k \in \{0, 1\}$) can be any number in $\mathbb{R}$ which ensures that $0 \leq \vec{u} < \infty$. Finally, we need to ensure that $0 \leq A + A^T \leq M$ holds which is done by taking

$$A = \begin{pmatrix} \frac{1}{2}cos^2(\theta_{00})m_{00}^2 & cos^2(\theta_{01})sin^2(\phi)m_{01}^2 \\ cos^2(\theta_{01})cos^2(\phi)m_{10}^2 & \frac{1}{2}cos^2(\theta_{11})m_{11}^2 \end{pmatrix}. \tag{4.7}$$

Where ϕ and θ_{kl} (with k, $l \in \{0, 1\}$) can be any number in $\mathbb{R}$. As a result $A + A^T$ becomes

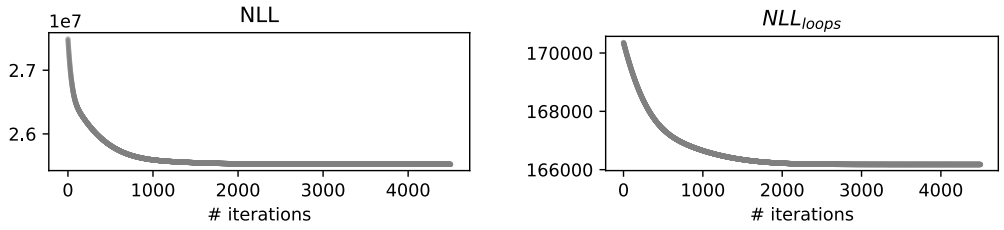
$$\begin{aligned} A + A^T &= \begin{pmatrix} 2\left(\frac{1}{2}cos^2(\theta_{00})m_{00}^2\right) & cos^2(\theta_{01})m_{01}^2\left(sin^2(\phi) + cos^2(\phi)\right) \\ cos^2(\theta_{01})m_{01}^2\left(sin^2(\phi) + cos^2(\phi)\right) & 2\left(\frac{1}{2}cos^2(\theta_{11})m_{11}^2\right) \end{pmatrix} \\ &= \begin{pmatrix} cos^2(\theta_{00})m_{00}^2 & cos^2(\theta_{01})m_{01}^2 \\ cos^2(\theta_{01})m_{01}^2 & cos^2(\theta_{11})m_{11}^2 \end{pmatrix} \end{aligned} \tag{4.8}$$

from which we can conclude that $0 \leq A + A^T \leq M$ holds since $0 \leq cos^2(x) \leq 1$.

To conclude, with the above described method for minimizing the NLL-functions, we can find estimated values for the parameters m_{00}, m_{01}, m_{11}, u_0, u_1, θ_{00}, θ_{01}, θ_{11} and ϕ. The initial values of these parameters are randomly generated real-valued numbers. The initial values of u_0 are in the range $[10, 15]$, while for all other parameters their initial values are in the range $[0, 10]$. At the end of the gradient descent method, these parameters finally determine the sought after matrices M and A and the vector $\vec{u}$.

4.2.3 Error Measurement

The above described method for determining the values for the parameters is not perfect due to the fact that the initial values need to be guessed. In theory, the gradient descent method should nonetheless find values for the parameters such that the gradients of the NLL-functions go to 0. However, this only occurs when the number of iterations of the method got to ∞. Of course this is not feasible in real life which is why the number of iterations have been set to a large value such that the gradients will indeed approach significantly small numbers within a manageable amount



(a) *Convergence of the NLL-function given in equation 4.3*

(b) *Convergence of the NLL-function for self-lops given in equation 4.4*

Figure 4.1: *Convergence of NLL-functions within a total of 4500 iterations.*

of time. The number of iterations is taken to be 4500. To prove that the gradients indeed decrease (very) small values, the convergence of the NLL-functions have been plotted in figure 4.1. This finite value for the number of iterations does however mean that the values of the parameters will never be perfect since the gradients can never truly reach 0. Thus, we are subjective to two types of errors that emerge using this method: an initialization error due to the random guess of the initial values and a standard error due to the finite gradient descent method.

We can measure the initialization error by performing the above described gradient descent method multiple times and determining the standard deviation of the resulting set of values of the parameters. We have chosen to execute the gradient descent method 10 separate times. At the start of every execution, new random initial values for the parameters are generated. The final values of the NLL-functions are stored in a list as well as the final parameter values. Since the gradient descent method is designed to minimize the NLL-functions, we can argue that the best execution of this method is the one that yields the lowest value for the NLL-functions. Its corresponding set of parameter values are label as the ”best” final values for the parameters. Using the complete set of parameter values from all executions we can determine the initialization error by using `np.std`.

The standard error can be determined by utilizing the Hessian matrix [12]. The Hessian matrix of the negative log-likelihood functions is defined as given in equation 4.9 where the derivatives are with respect to the set of parameters $\{m_{00}, m_{01}, m_{11}, \theta_{00}, \theta_{01}, \theta_{11}, \phi\}$ and $\{m_{00}, m_{01}, m_{11}, u_0, u_1\}$, respectively. Upon analyzing the NLL-functions, we can conclude that their second derivatives are continuous functions which results in the

Hessian matrix being symmetric. However, we do not need to explicitly calculate this Hessian matrix as we can use the function `jax.hessian` to calculate H_{NLL} and $H_{NLL_{loops}}$. Taking the inverse of these Hessian matrices yield the covariance-matrices $\left(CV_{NLL} \text{ and } CV_{NLL_{loops}}\right)$ for both sets of parameters. Taking the square root of the diagonal of CV_{NLL} and $CV_{NLL_{loops}}$ leaves us with the standard error on every parameter.

$$\begin{aligned} \left(H_{NLL}\right)_{ij} &= \frac{\partial^2 NLL}{\partial x_i \partial x_j} \\ \left(H_{NLL_{loops}}\right)_{ij} &= \frac{\partial^2 NLL_{loops}}{\partial x_i \partial x_j} \end{aligned} \tag{4.9}$$

4.3 Coarse Graining

After having obtained explicit values for the elements of the matrices M and A and vector $\vec{u}$ by fitting the model to the data as described above, we need to determine the performance of our (now explicitly determined) reciprocated model as a "scale-invariant model".

We start by coarse graining our transaction data to obtain transaction networks of higher levels (think of a network of transactions between sectors instead of transactions between individual organizations). We can do this by utilizing the NAICS-codes in our data as described in chapter 3 (an example of how to utilize the NAICS-codes is given in table 4.3). In this research we have coarse grained three times and thus we have obtained four transaction networks on different levels/scales. In what follows, we will refer to these networks as network scale 1 (being the network of transactions between organizations), and scales 2, 3 and 4. In the transaction data of network scale 1, the NAICS-codes are 6-digits numbers, for scale 2 they are coarse grained to 5-digit numbers, 4-digit numbers for scale 3 and finally on scale 4 the NAICS-codes are 3-digit numbers. This means that the number of 323880 vertices for network scale 1 has been reduces to 972 vertices for scale 2, 647 for scale 3 and 305 vertices for scale 4.

Note that we do not fit our model on every separate network scale. Rather, we fit the model once on the network with scale 1 and evaluate the performance of the obtained explicit model on *all* four networks. Than the performance of our model is determined by calculating expected degree

NAICS-code	Title
21	Mining, Quarrying, and Oil and Gas Extraction
2111	Oil and Gas Extraction
211120	Crude Petroleum Extraction
211130	Natural Gas Extraction
2121	Coal Mining
212114	Surface Coal Mining
212115	Underground Coal Mining
2122	Metal Ore Mining
212210	Iron Ore Mining
212220	Gold Ore and Silver Ore Mining height

Table 4.3: *Fraction of the NAICS-codes table within the mining sector [11].*

and weight distributions an comparing them to the empirical degree and weight distributions for all four network scales (more on this in the next chapter).

Chapter 5

Results and Discussion

This chapter covers the results of the methods described in the previous chapter. We will first analyze the parameter values including their error measurements after which the expected degree and weight distributions will be discussed.

5.1 Estimated Parameter Values

Parameter	Parameter value	Standard error	Initialization error
M_{00}	0.2030	0.0047	0.2540
M_{01}	1.1373	0.0043	0.1998
M_{11}	1.6113	0.0055	0.1594
A_{00}	0.1015	0.0021	0.1174
A_{01}	0.4812	0.0022	0.0823
A_{10}	0.5376	0.0026	0.1009
A_{11}	0.6837	0.0025	0.0721
u_0	65.6394	1.3686	0.0005
u_1	115.5169	1.6431	0.0005

Table 5.1: *Parameter values calculated using methods described in chapter 4 including the standard and initialization errors resulting from the described methods.*

The methods described in the previous chapter yield values for the parameters m_{00}, m_{01}, m_{11}, u_0, u_1, θ_{00}, θ_{01}, θ_{11} and ϕ. With the use of equations 4.5, 4.6 and 4.7 the values for the elements of the matrices M and A, and vector $\vec{u}$ can than be obtained. The resulting parameter values are listed

in table 5.1 including the standard and initialization errors that emerged during the calculations.

Upon analyzing table 5.1, we see that the standard errors (determined by utilizing the Hessian matrix) are approximately 2 orders of magnitude smaller than the determined parameter values. We assume this to be a result of the fact that the data set is indeed considerable large. Combine this with the fact that the terms in the NLL-functions are well behaved which allow the gradient descent method to smoothly minimize these functions, results in a low standard error. The initialization error clearly has more influence on the parameter values * (especially on the M_{00} and A_{00} parameters where it could be argued that the parameter values should be equal to zero since the initialization errors are approximately equal to the parameter values, however we have chosen not to manually change the outcome of the fitting algorithm), except for the parameters of vector $\vec{u}$ which are more than a magnitude larger than the other parameters. We assume this to be a consequence of self-loop connections to obey to different physics than the non-self-loops connections.

Now that the parameter values of equations 2.11, 2.12, 2.13 and 2.14 are known (these are only calculated once which is done on the network with scale 1), we can explicitly calculate the probabilities that pairs of nodes are connected to each other in a certain configuration for given values of the node strengths which varies across different network scales.

5.2 Degree distributions

Upon calculating the expected degree distributions using equations 2.11, 2.12, 2.13 and 2.14, we can identify three flavors of expected degrees: conditional degree distributions, unconditional degree distributions and undirected degree distributions (we use `joblib.Parallel` to speed up the computation of these distributions). These three expected degree distributions are calculated for all four different network scales and compared to their empirical degree distributions.

We calculate the expected conditional degree distributions using the

*The large difference between the values of the initialization errors and the standard errors should not go unnoticed and would be interesting to investigate further in follow up research. This is for now outside of the scope of this thesis.

following set of equations:

$$\left\langle k_i^{01} \right\rangle = \sum_{(i,j,(j\neq i))\in I^{01}} p_{ij}^{01} \tag{5.1}$$

$$\left\langle k_i^{10} \right\rangle = \sum_{(i,j,(j\neq i))\in I^{10}} p_{ij}^{10} \tag{5.2}$$

$$\left\langle k_i^{11} \right\rangle = \sum_{(i,j,(j\neq i))\in I^{11}} p_{ij}^{11} \tag{5.3}$$

where, $\left\langle k_i^{01} \right\rangle$ gives us the expected in-degree of a node i. This means that, for node pair (i, j), i does not connect to j while j *does* connect to i. Similarly, $\left\langle k_i^{10} \right\rangle$ gives us the expected out-degree of node i (meaning that i connects to j while j does not connect to i), and $\left\langle k_i^{11} \right\rangle$ gives the expected reciprocated degree distributions for a node i (for a pair (i, j), both i and j are connected to each other).

The expected unconditional degree distributions are calculated using the following two equations:

$$\left\langle k_i^{*1} \right\rangle = \sum_{(i,j,(j\neq i))\in I^{01}} p_{ij}^{01} + \sum_{(i,j,(j\neq i))\in I^{11}} p_{ij}^{11} \tag{5.4}$$

$$\left\langle k_i^{1*} \right\rangle = \sum_{(i,j,(j\neq i))\in I^{10}} p_{ij}^{10} + \sum_{(i,j,(j\neq i))\in I^{11}} p_{ij}^{11} \tag{5.5}$$

where $\left\langle k_i^{*1} \right\rangle$ gives us the expected unconditional in-degree of node i provided that we do not know whether node i connects to node j, thus we only know that j connects to i. Vice versa, $\left\langle k_i^{1*} \right\rangle$ gives us the expected unconditional out-degree of node i.

Finally, to calculate the expected undirected degree distribution we use

$$\langle k_i \rangle = \sum_{(i,j,(j\neq i))\in\{I^{01},I^{10},I^{11}\}} \left(1 - p_{ij}^{00}\right) \tag{5.6}$$

which gives us the expected degree of a node i that is connected to a node j in some configuration.

To accurately compare these distributions to the empirical degree distributions, we take the cumulative distribution for both the expected and empirical degree distributions. The resulting plots in which the expected

degree distributions are compared to the empirical degree distributions are given in the appendix.The cumulative expected and empirical degree distributions for network scale 1 are given in section 7.4.1, for network scales 2, 3, and 4 see sections 7.4.2, 7.4.3 and 7.4.4 respectively.

For the transaction network at scale 1 (with 323880 vertices), figures 7.7, 7.8 and 7.9 show that the predicted degrees are smaller than the empirical degrees since the orange curves are throughout the whole domain below the blue curves. Now since degrees need to be integer valued numbers (it is impossible to be connected to half a vertex), we round the expected degree distributions (orange curve) to integers (green curves). In the figures we see that these cumulative rounded expected degree distributions are still below the cumulative empirical degree distributions for the largest part of the domain.

We can extract a similar conclusion for the curves of the transaction network at scale 2 (with 972 vertices. See figures 7.10, 7.11 and 7.12). However, here we see that the three curves (the expected, the expected rounded to integers, and the empirical) are significantly closer together where in some cases the green and the blue curves even seem to overlap each other on a significant portion of the domain.

For the transaction network at scale 3 having 647 vertices (see figures 7.13, 7.14 and 7.15) this overlapping of curves seems to be similar. Now the orange curves (the not-rounded cumulative expected degree distributions) seems to be mimicking the trajectory of the blue curves more closely as the model start ”rejecting” (very) low valued degrees for this data set.

Finally, the curves of the cumulative degree distributions for the transaction network at scale 4 which has 305 vertices (figures 7.16, 7.17 and 7.18) almost completely overlap each other. With the exception that the cumulative expected conditional in- and out-degree distributions (figure 7.16) abruptly drops of and thus cannot get close to the cumulative empirical degree distributions for large valued degrees.

Thus, we see that the performance of the model for accurately predicting the cumulative degree distributions increases as the network scale increases. Increasing the network scale decreases the total number of nodes in a network. Therefore, the maximum allowed degree per node decreases as well. Additionally, the minimum empirical degree per node increases. This becomes evident upon analyzing the horizontal axis for the empiri-

cal degree distribution in figures 7.7 through 7.18 where we see that the domain of the empirical degree distributions decreases from $[k_{min}, k_{max}] = [10^0, 10^5]$ to $[k_{min}, k_{max}] = [10^1, 10^3]$ as the scale increases. It is believed that this decrease in the values for the global maxima and minima in the empirical data allows for an increased accuracy of the predictions.

5.3 Weight distributions

In a similar fashion to how the performance of the model can be tested by calculating the expected degree distributions and comparing them to the empirical degree distributions as described in the previous section, we can calculate the expected weight distributions which we will compare to the empirical weight distribution and the empirical average weight distribution of the entire network. We can identify two flavours of expected weight distributions that we will compare with the empirical weight distribution: weight distributions for directed edges, and weight distributions for undirected edges.

The weight distributions for directed edges are calculated by

$$\left\langle w_{\overrightarrow{ij}} \right\rangle = \frac{s_i^{out} s_j^{in} p_{ij}^{10}}{W\left(p_{ij}^{10} + p_{ij}^{11}\right)} \quad \forall i, j \tag{5.7}$$

$$\left\langle w_{\overleftrightarrow{ij}} \right\rangle = \frac{s_i^{out} s_j^{in} p_{ij}^{11}}{W\left(p_{ij}^{10} + p_{ij}^{11}\right)} \quad \forall i, j \tag{5.8}$$

$$\left\langle w_{\overleftrightarrow{ij}|a_{ij}} \right\rangle = \frac{s_i^{out} s_j^{in} a_{ij}}{W\left(p_{ij}^{10} + p_{ij}^{11}\right)} \quad \forall i, j \tag{5.9}$$

which give us 1) the expected weight of a connection between nodes i and j where node i is connected to node j while j is *not* connected to i. 2) the expected weight of a reciprocated connection between nodes i and j, and 3) the expected weight of a connection between nodes i and j of which its existence is known (i.e. a_{ij} is element (i, j) of the adjacency matrix). In the above equations, s_i^{out} is the out-strength of node i, s_i^{in} is the in-strength of node i, and W is the total weight of the network $\left(W = \sum_i s_i^{out} = \sum_j s_j^{in}\right)$. Similarly, the weight distributions for undirected edges can be calculated

using

$$\left\langle w_{ij|a_{ij}} \right\rangle = \frac{s_i^{out} s_j^{in} a_{ij}}{W p_{ij}} \tag{5.10}$$

where $p_{ij} = 1 - p_{ij}^{00}$. Again a_{ij} is element (i, j) of the adjacency matrix. Lastly, the empirical average weight distribution of the entire network is determined by

$$\bar{w}_{empirical} = \frac{s_i^{out} s_j^{in}}{W}. \tag{5.11}$$

Again we determine the cumulative distribution of these weight distribution functions (in units of €$\cdot 10^{-2}$ for convenience) to be able to accurately compare the weight distributions with each other. The resulting plots are given in the appendix (sections 7.5.1, 7.5.2, 7.5.3 and 7.5.4 for the transaction networks at scales 1 through 4 respectively).

For the transaction network at scale 1, we see in figure 7.19 the cumulative expected, average and empirical weight distributions for directed edges while figure 7.20 shows the cumulative expected, average and empirical weight distributions for undirected edges. We can directly see that the cumulative distributions of $\left\langle w_{\overrightarrow{ij}} \right\rangle$ (green triangles) and $\left\langle w_{\overleftrightarrow{ij}} \right\rangle$ (red squares) closely follow the cumulative distribution of $\bar{w}_{empirical}$ (blue dots). However, the cumulative distribution of $\left\langle w_{\substack{\overrightarrow{\longleftarrow} \\ ij|a_{ij}}} \right\rangle$ (orange diamonds) follows more closely the cumulative empirical weight distribution (purple stars). This holds on all 4 network scales (see remaining figures 7.21, 7.23 and 7.25). In all of these figures we see that the orange diamonds curves are well below the cumulative empirical weight distributions for the majority of the domain. Only for high valued weights these curves seem to converge.

This also seems to be true for the cumulative expected weight distribution for undirected edges (figures 7.20, 7.22, 7.24, and 7.26). Here the cumulative curves of $\left\langle w_{ij|a_{ij}} \right\rangle$ stay well below the cumulative empirical curves while they only move towards each other for high valued weights.

Finally, it is worth mentioning that the cumulative empirical and average distributions naturally move towards each other the further we coarse grain the transaction network. This is of coarse a result of the fact that the number of vertices significantly decreases while coarse graining and

as a result, making the evaluated transaction network more dense with a smaller deviation in the amount of money being transferred per edge. Similarly to what we discussed in the previous section, we belief that this decrease in global maxima and minima allows for more accurate predictions of the model.

Chapter 6

Summary and Conclusion

In chapter 2 we presented our directed scale-invariant model where our reasoning started of with the premise that transactions and edge weights (read: transferred amount of money) are additive variables. This allowed us to obtain an expression for the probability that a node i connects to a different node j. However, in this expression the behaviour of node j remained unknown. To that end we derived a network model that has control over reciprocity. We defined four probability distributions which together described the behaviour of both nodes i and j (equations 2.11-2.14). To ensure these probability distributions live only within the domain $[0, 1]$ we introduced boundary conditions on the values of the parameters as given in equation 2.15. Additionally, in the same chapter we derived the effective Hamiltonian of this system as well as the partition function which has proven to be constant and scale-invariant.

A discussion of the data has been presented in chapter 3. In this chapter we explained how the raw transaction data has been cleaned in order to obtain a transaction network solely between firms and organizations. This sparse network, consisting of approximately $3.2 \cdot 10^5$ nodes and approximately $2.6 \cdot 10^6$ edges, is said to be closed (meaning that there are no financial transactions going to or coming from other banks) while it allows for self-loop connections to exist.

To obtain values for the parameters of the model we used the method described in chapter 4. We first explained that a small artificial transaction network is required to test the fitting algorithm during the process of its development. The specific details of these artificial networks themselves are not as important as the method with which they are generated to make

them as realistic as possible.

Before the fitting algorithm is explained we introduced a new labeling system in which we (in principle) count the locations of the nodes in the adjacency matrix. We defined the "tail-nodes" by $p\mathcal{N} + b$ and the "head-nodes" by $b\mathcal{N} + p$ where p and b are respectively the standard payer and beneficiary node labels. Using this new labeling system we argued how to obtain values for the parameters by minimizing the negative log-likelihood functions (NLL for connections between nodes i and j with $i \neq j$, and NLL_{loops} for connections between nodes i and j with $i = j$). Within these functions the constraints on the parameters are realized by introducing a new set of parameters (m_{00}, m_{01}, m_{11}, u_0, u_1, θ_{00}, θ_{01}, θ_{11} and ϕ) as given in equations 4.5, 4.6 and 4.7. The standard deviations are than determined by executing this method multiple times, and by utilizing the Hessian matrix.

Finally, in chapter 4 we explain how to use the provided NAICS-codes of the data to coarse-grain the transaction network. With it, the performance of the model to determine the expected cumulative degree and weight distributions are measured on four different network scales.

The results are discussed in chapter 5 (figures are provided in the appendix). For the expected degree distributions, the model's predictability improved the further we coarse-grained the transaction network. We saw that the model tends to put more emphasis on small degrees. As the network scale is increased, this emphasis seems to slowly fade towards moderate valued degrees. It has been discussed that this is most likely a result of the increasing uniformity of the empirical data across a smaller domain that is imposed on the network by coarse-graining. Similarly for the expected weight distributions, we discussed that the decrease in global maxima and minima upon coarse-graining the network allowed for more accurate predictions on the weight distributions. In spite of this, the expected cumulative weight distributions seemed to be significantly more in line with the empirical cumulative weight distributions compared to what has been shown for the degree distributions.

Clearly further investigations are required to improve the model's performance on predicting the expected degree and weight distributions. Adding extra parameters to the model such that, for example, the account balance of organizations are accounted for might benefit the performance.

Chapter 7

Appendix

In this chapter we provide additional information to what is described (and often referred to) in the preceding chapters. In section 7.1 we carefully derive the probability distributions given in equations 2.11-2.14. Since these equations can only have a physical meaning within the range $[0, 1]$ we need to impose boundary conditions on the parameters which are derived in section 7.2. In section 7.3 we present additional information on the empirical data described in chapter 3. Finally, in sections 7.4 and 7.5 we present the expected cumulative degree plots and the expected cumulative weight plots, respectively, that are discussed in chapter 5.

7.1 Derivation of p_{Ij}^{00}, p_{Ij}^{01}, p_{Ij}^{10} and p_{Ij}^{11}

In this section we show in detail how the probability distributions given in equations 2.11-2.14 are derived.

7.1.1 Derivation of p_{Ij}^{00}

Calculate the probability that node cluster I (consisting of nodes i and k) does not connect to node j in neither direction.

$$p_{Ij}^{00} = p_{\{i,k\},j}^{00} = p^{00}(\vec{x}_i + \vec{x}_k; \vec{y}_j) = p_{ij}^{00}(\vec{x}_i; \vec{y}_j) p_{kj}^{00}(\vec{x}_k; \vec{y}_j) \tag{7.1}$$

First we take the derivative of the above probability with respect to the α^{th} index of the i^{th} node of the first argument: $\frac{\partial}{\partial x_i^\alpha}$.

$$\partial x^\alpha p^{00}(\vec{x}_i + \vec{x}_i; \vec{y}_j) \frac{\partial (x_i^\alpha + x_k^\alpha)}{x_i^\alpha} = p^{00}(\vec{x}_k; \vec{y}_j) \partial x^\alpha p^{00}(\vec{x}_i; \vec{y}_j) \frac{\partial x_i^\alpha}{\partial x_i^\alpha} + p^{00}(\vec{x}_i; \vec{y}_j) \partial x^\alpha p^{00}(\vec{x}_k; \vec{y}_j) \frac{\partial x_k^\alpha}{\partial x_i^\alpha} \quad (7.2)$$

Where in the above $\partial x^\alpha p^{00}(...;...)$ denotes the derivative of the function p^{00} with respect to the α^{th} index of the first argument of that function which can be rewritten to

$$\partial x^\alpha p^{00}(\vec{x}_i + \vec{x}_i; \vec{y}_j) = p^{00}(\vec{x}_k; \vec{y}_j) \partial x^\alpha p^{00}(\vec{x}_i; \vec{y}_j). \quad (7.3)$$

Using this, we can write eq. 7.2 in a more convenient form:

$$\frac{\partial x^\alpha p^{00}(\vec{x}_i + \vec{x}_i; \vec{y}_j)}{p^{00}(\vec{x}_i + \vec{x}_k; \vec{y}_j)} = \frac{p^{00}(\vec{x}_k; \vec{y}_j) \partial x^\alpha p^{00}(\vec{x}_i; \vec{y}_j)}{p^{00}(\vec{x}_i; \vec{y}_j) p^{00}(\vec{x}_k; \vec{y}_j)} = \frac{\partial x^\alpha p^{00}(\vec{x}_i; \vec{y}_j)}{p^{00}(\vec{x}_i; \vec{y}_j)}. \quad (7.4)$$

Now, we take the derivative with respect to the β^{th} index (note that α and β can be equal toe each other since there are no constraints on the indices) of the k^{th} node of the first argument $\left(\frac{\partial}{\partial x_k^\beta}\right)$. This leads to

$$\partial x^\beta \frac{\partial x^\alpha p^{00}(\vec{x}, \vec{y})}{p^{00}(\vec{x}, \vec{y})} = 0. \quad (7.5)$$

Notice that, since the two derivatives ∂x^α and ∂x^β are with respect to function arguments and not specific node values $\vec{x}_i$ or $\vec{x}_k$, we can write $\vec{x}$ in place of the node values $x_i + x_k$ and $\vec{y}$ in place of the node values $\vec{y}_j$. Moreover, eq. 7.5 holds for every β. Therefore, $\frac{\partial x^\alpha p^{00}}{p^{00}}$ cannot depend on any x^β, only on $\vec{y}$. Thus,

$$\frac{\partial x^\alpha p^{00}(\vec{x}, \vec{y})}{p^{00}(\vec{x}, \vec{y})} = c^\alpha(\vec{y}) \quad (7.6)$$

for some to be determined set of functions $c^\alpha(\vec{y})$. This means that $\partial x^\alpha p^{00} = c^\alpha(\vec{y}) p^{00}$, and therefore,

$$p^{00}(\vec{x}, \vec{y}) = e^{\sum_\alpha c^\alpha(\vec{y}) x^\alpha + c_0(\vec{y})} \quad (7.7)$$

with another to be determined function $c_0(\vec{y})$. Notice that the summation is in principle a dot-product.

Similarly, if we demand that

$$p_{i,\{j,k\}}^{00} = p^{00}(\vec{x}_i; \vec{y}_j + \vec{y}_k) = p_{ij}^{00}(\vec{x}_i; \vec{y}_j) p_{ik}^{00}(\vec{x}_i; \vec{y}_k) = p^{00}(\vec{x}_i + \vec{x}_k; \vec{y}_j) \tag{7.8}$$

which is the probability that node cluster J (consisting of nodes j and k) does not connect to node i in neither direction. Again, notice that above the derivatives are with respect to the function arguments and not specific node values. This means that $p^{00}(\vec{x}_i; \vec{y}_j + \vec{y}_k) = p^{00}(\vec{x}_i + \vec{x}_k; \vec{y}_j)$ holds by construction. And thus for $p_{i,\{j,k\}}^{00}$ we end up with a similar expression as for $p_{\{i,j\},k}^{00}$:

$$p_{i,\{j,k\}}^{00}(\vec{x}, \vec{y}) = e^{\sum_\alpha c'^\alpha(\vec{x}) y^\alpha + c'_0(\vec{x})}. \tag{7.9}$$

So we have simultaneously

$$e^{\sum_\alpha c^\alpha(\vec{y}) x^\alpha + c_0(\vec{y})} = e^{\sum_\alpha c'^\alpha(\vec{x}) y^\alpha + c'_0(\vec{x})} \tag{7.10}$$

giving

$$\sum_\alpha c^\alpha(\vec{y}) x^\alpha + c_0(\vec{y}) = \sum_\alpha c'^\alpha(\vec{x}) y^\alpha + c'_0(\vec{x}). \tag{7.11}$$

To determine $c^\alpha(\vec{y})$ and $c'^\alpha(\vec{x})$ we start by differentiating both sides with respect to x^m (with $m \neq \alpha$):

$$c^m(\vec{y}) = \sum_\alpha \frac{\partial c'^\alpha}{\partial x^m}(\vec{x}) y^\alpha + \frac{\partial c'^\alpha_0}{\partial x^m}(\vec{x}). \tag{7.12}$$

Next, we differentiate both sides with respect to y^n:

$$\frac{\partial c^m}{\partial y^n}(\vec{y}) = \frac{\partial c'^n}{\partial x^m}(\vec{x}). \tag{7.13}$$

The last equation can only be true if both sides are constant:

$$\frac{\partial c^m}{\partial y^n}(\vec{y}) = \frac{\partial c'^n}{\partial x^m}(\vec{x}) = M^{mn} \tag{7.14}$$

and therefore,

$$c^m(\vec{y}) = \sum_n M^{mn} y^n + U^m \tag{7.15}$$

$$c'^n(\vec{x}) = \sum_m M^{mn} x^m + V^n \tag{7.16}$$

for some set of constants U^m and V^n.

To determine $c^\alpha_0(\vec{y})$ and $c'^\alpha_0(\vec{x})$, we fill the above expressions for c^m and c'^n into equation 7.11 giving:

$$\sum_m U^m x^m + c_0(\vec{y}) = \sum_n V^n y^n + c'_0(\vec{x}) \tag{7.17}$$

from which you can read off that

$$c_0(\vec{y}) = \sum_n V^n y^n + W \tag{7.18}$$

$$c'_0(\vec{x}) = \sum_m U^m x^m + W \tag{7.19}$$

for some constant W. Thus, we can conclude that

$$p^{00}(\vec{x},\vec{y}) = e^{\sum_{m,n} x^m M^{mn} y^n + \sum_m U^m x^m + \sum_n V^n y^n + W}. \tag{7.20}$$

Now we need to compare this expression to our initial conditions. By doing this we see that the term $\sum_n V^n y^n$ does not agree with $p^{00}_{\{i,k\},j} = p^{00}_{i,j} p^{00}_{k,j}$, while the term $\sum_m U^m x^m$ does not agree with $p^{00}_{i,\{k,j\}} = p^{00}_{i,j} p^{00}_{i,k}$, and the term W does not agree with either. In the derivations this was not seen, because we differentiated and while terms did not take part in the respective derivatives after which they just got divided out. Equation 7.5 should therefore be a necessary condition which leads us to the final form for the probability p^{00}:

$$p^{00}(\vec{x},\vec{y}) = e^{\sum_{m,n} x^m M^{mn} y^n} = e^{\vec{x}^T \cdot M \cdot \vec{y}}. \tag{7.21}$$

This expression can only hold under the condition $p^{00}_{\{i,k\},j} = p^{00}_{i,j} p^{00}_{k,j} = p^{00}_{i,\{k,j\}}$ if matrix M is symmetric ($M = M^T$). For what follows in section 7.2 it will be more convenient to take a factor of -1 out of that matrix M. So we are left with

$$p^{00}(\vec{x},\vec{y}) = e^{-\vec{x}^T \cdot M \cdot \vec{y}}. \tag{7.22}$$

7.1.2 Derivation of p^{01}_{Ij} and p^{10}_{Ij}

The probability p^{01} is defined as follows

$$p^{01}_{\{i,k\},j} = p^{01}_{ij} p^{01}_{kj} + p^{01}_{ij} p^{00}_{kj} + p^{00}_{ij} p^{01}_{kj}. \tag{7.23}$$

Upon adding $p^{00}(\vec{x}_i + \vec{x}_k; \vec{y}_j)$ to both sides of the above equation we end up with a more manageable form for the initial condition

$$p^{01}_{\{i,k\},j} + p^{00}_{\{i,k\},j} = p^{01}_{ij}p^{01}_{kj} + p^{01}_{ij}p^{00}_{kj} + p^{00}_{ij}p^{01}_{kj} + p^{00}_{ij}p^{00}_{kj} = \left(p^{01}(\vec{x}_i, \vec{y}_j) + p^{00}(\vec{x}_i, \vec{y}_j)\right)\left(p^{01}(\vec{x}_k, \vec{y}_j) + p^{00}(\vec{x}_k, \vec{y}_j)\right). \quad (7.24)$$

Notice that at this point you have the same situation as you did for p^{00}: we have $f(\vec{x}_i + \vec{x}_k, \vec{y}_j) = f(\vec{x}_i, \vec{y}_j)f(\vec{x}_k, \vec{y}_j)$ and $f(\vec{x}_i, \vec{y}_k + \vec{y}_j) = f(\vec{x}_i, \vec{y}_k)f(\vec{x}_i, \vec{y}_j)$, with $f = p^{01} + p^{00}$. For this reason, the derivation of the form of f is exactly the same as above and leads to $f(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K \cdot \vec{y}}$ for some matrix K. This than gives us:

$$p^{01}(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}} \quad (7.25)$$

while the similar arguments also hold for p^{10}: now we have $f' = p^{10} + p^{00}$ with $f(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K' \cdot \vec{y}}$ for some matrix K'. Thus,

$$p^{10}(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K' \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}}. \quad (7.26)$$

however, due to symmetry, $p^{10}(\vec{x}_i + \vec{x}_k; \vec{y}_j) = p^{01}(\vec{x}_i; \vec{y}_j + \vec{y}_l)$ which leads to $K' = K^T$. To conclude, for p^{01} and p^{10} we are left with the following expressions:

$$p^{01}(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}} \quad (7.27)$$

$$p^{10}(\vec{x}, \vec{y}) = e^{-\vec{x}^T \cdot K^T \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}} \quad (7.28)$$

in chapter 7.2 it will turn out that it is convenient to write $K = M - A$ (and similarly $K^T = M - A^T$) for some non-symmetric matrix A.

7.1.3 Derivation of p^{11}_{Ij}

The probability p^{01} is defined by

$$p^{11}_{Ij} = 1 - p^{00}_{Ij} - p^{10}_{Ij} - p^{01}_{Ij}. \quad (7.29)$$

We can plug in the above results for p^{00}, p^{01} and p^{10} which yields the following expression for p^{11}_{Ij}:

$$\begin{aligned} p^{11}_{Ij} &= 1 - e^{-\vec{x}^T \cdot M \cdot \vec{y}} - \left(e^{-\vec{x}^T \cdot K \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}}\right) - \left(e^{-\vec{x}^T \cdot K^T \cdot \vec{y}} - e^{-\vec{x}^T \cdot M \cdot \vec{y}}\right) \\ &= 1 - e^{-\vec{x}^T \cdot K \cdot \vec{y}} - e^{-\vec{x}^T \cdot K^T \cdot \vec{y}} + e^{-\vec{x}^T \cdot M \cdot \vec{y}}. \end{aligned} \quad (7.30)$$

7.1.4 Self-loops: Derivation of p_{Ij}^{11} $(j \in I) \equiv p_I^s$

Assume a cluster I consisting of 2 nodes: i and j (its strength is given by $x_I = \sum_{a \in I} x_a = x_i + x_j$). Cluster I is said to have a self-loop when there exists a connection form i to j, and/or from j to i, and/or node i has a self-loop, and/or node j has a self-loop. Probability p_I^s will denote the probability that at least one of these four described edges exists within cluster I (i.e. the probability that cluster I has a self-loop).

$$p_I^s = 1 - p_I^{\text{no edges in } I} = 1 - p_I^n = 1 - p_{ij}^{00}(1 - p_i^s)(1 - p_j^s) \tag{7.31}$$

with $p_I^{\text{no edges in } I} = p_I^n$ for shorthand notation. Let $\frac{\partial}{\partial \alpha}$ be the derivative with respect to the in-strength of node i and let $\frac{\partial}{\partial \beta}$ be the derivative with respect to the out-strength of node j. Side note: we can also take the derivative w.r.t. the out-strength of i and in-strength of j in the following derivation.

Let's start by taking the derivative with respect to x_i^{in} (denoted by $\frac{\partial}{\partial \alpha}$)

$$\begin{aligned} p_I^s(x_I) = 1 - p_I^n(x_I) &= 1 - p^{00}(x_i, x_j)(1 - p_i^s(x_i))(1 - p_j^s(x_j)) \\ \frac{\partial}{\partial \alpha}\left[1 - p_I^n(x_I)\right] &= \frac{\partial}{\partial \alpha}\left[1 - p^{00}(x_i, x_j)(1 - p_i^s(x_i))(1 - p_j^s(x_j))\right] \\ \frac{\partial}{\partial \alpha}\left[p_I^n(x_I)\right] &= (1 - p_j^s(x_j))\frac{\partial}{\partial \alpha}\left[p^{00}(x_i, x_j)(1 - p_i^s(x_i))\right]. \end{aligned} \tag{7.32}$$

Now we will divide the above by p_I^n:

$$\begin{aligned} \frac{\frac{\partial}{\partial \alpha}\left[p_I^n(x_I)\right]}{p_I^n} &= \frac{(1 - p_j^s(x_j))\frac{\partial}{\partial \alpha}\left[p^{00}(x_i, x_j)(1 - p_i^s(x_i))\right]}{p^{00}(x_i, x_j)(1 - p_i^s(x_i))(1 - p_j^s(x_j))} \\ \frac{\frac{\partial}{\partial \alpha}\left[p_I^n(x_I)\right]}{p_I^n} &= \frac{\frac{\partial}{\partial \alpha}\left[p^{00}(x_i, x_j)(1 - p_i^s(x_i))\right]}{p^{00}(x_i, x_j)(1 - p_i^s(x_i))} \\ \frac{\frac{\partial}{\partial \alpha}\left[p_I^n(x_I)\right]}{p_I^n} &= \frac{\frac{\partial}{\partial \alpha}\left[1 - p_i^s(x_i)\right]}{1 - p_i^s(x_i)} + \frac{\frac{\partial}{\partial \alpha}\left[p^{00}(x_i, x_j)\right]}{p^{00}(x_i, x_j)}. \end{aligned} \tag{7.33}$$

Once again we will take the derivative. However, this time the derivative will be taken with respect to the out-strength of node j (denoted by

$\frac{\partial}{\partial\beta}$)

$$\begin{aligned}\frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right]}{p_I^n}\right] &= \frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[1-p_i^s(x_i)\right]}{1-p_i^s(x_i)}+\frac{\frac{\partial}{\partial\alpha}\left[p^{00}(x_i,x_j)\right]}{p^{00}(x_i,x_j)}\right]\\ \frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right]}{p_I^n}\right] &= \frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[p^{00}(x_i,x_j)\right]}{p^{00}(x_i,x_j)}\right].\end{aligned} \tag{7.34}$$

Since we already know what $p^{00}(x_i, x_j)$ looks like, we can easily calculate the right-hand-sight of the above equation explicitly. Doing so, we end-up with

$$\frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right]}{p_I^n}\right] = -M_{a,b} \tag{7.35}$$

where a and b refer to the matrix element of M. In this example, $a = 1$ and $b = 2$ since we have taken the derivative first with respect to the in-strength of node i while the second derivative was taken with respect to the out-strength of node j. Since $M^T = M$, the following is also true:

$$\frac{\partial}{\partial\beta}\left[\frac{\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right]}{p_I^n}\right] = -\frac{1}{2}\left(M_{a,b}+M_{b,a}\right). \tag{7.36}$$

Intermezzo: We will calculate the derivatives of both 1) $-\vec{x_I}^T \cdot M \cdot \vec{x_I}$ and 2) $\vec{u}^T \cdot \vec{x_i}$ (with $\vec{u}^T$ a 2-dimensional vector like $\vec{x_i}$)

1)

$$\begin{aligned}-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} &= -\frac{1}{2}\left(\vec{x_i}^T+\vec{x_j}^T\right)\cdot M\cdot\left(\vec{x_i}+\vec{x_j}\right)\\ &= -\frac{1}{2}\left(\vec{x_i}^T\cdot M\cdot\vec{x_i}+\vec{x_i}^T\cdot M\cdot\vec{x_j}+\vec{x_j}^T\cdot M\cdot\vec{x_i}+\vec{x_j}^T\cdot M\cdot\vec{x_j}\right)\\ &= -\frac{1}{2}\Big[x_i^{in}\left(M_{11}x_i^{in}+M_{12}x_i^{out}\right)+x_i^{out}\left(M_{21}x_i^{in}+M_{22}x_i^{out}\right)\\ &\quad + x_i^{in}\left(M_{11}x_j^{in}+M_{12}x_j^{out}\right)+x_i^{out}\left(M_{21}x_j^{in}+M_{22}x_j^{out}\right)\\ &\quad + x_j^{in}\left(M_{11}x_i^{in}+M_{12}x_i^{out}\right)+x_j^{out}\left(M_{21}x_i^{in}+M_{22}x_i^{out}\right)\\ &\quad + x_j^{in}\left(M_{11}x_j^{in}+M_{12}x_j^{out}\right)+x_j^{out}\left(M_{21}x_j^{in}+M_{22}x_j^{out}\right)\Big]\end{aligned} \tag{7.37}$$

Apply $\frac{\partial}{\partial\alpha}$:

$$\frac{\partial}{\partial\alpha}\left[-\frac{1}{2}\vec{x_I}^T\cdot M\cdot\vec{x_I}\right] = -\frac{1}{2}\Big(2M_{11}x_i^{in} + M_{12}x_i^{out} + M_{21}x_i^{out} + M_{11}x_j^{in} + M_{12}x_j^{out} + M_{11}x_j^{in} + M_{21}x_j^{out}\Big). \quad (7.38)$$

Apply $\frac{\partial}{\partial\beta}$:

$$\frac{\partial}{\partial\beta}\left[\frac{\partial}{\partial\alpha}\left[-\frac{1}{2}\vec{x_I}^T\cdot M\cdot\vec{x_I}\right]\right] = -\frac{1}{2}\left(M_{12}+M_{21}\right) = -M_{12} \qquad (M=M^T). \quad (7.39)$$

2)

$$\vec{u}^T\cdot\vec{x_I} = \vec{u}^T\cdot\left(\vec{x_i}+\vec{x_j}\right) = u_1x_i^{in} + u_2x_i^{out} + u_1x_j^{in} + u_2x_j^{out} \quad (7.40)$$

Apply $\frac{\partial}{\partial\alpha}$:

$$\frac{\partial}{\partial\alpha}\left[\vec{u}^T\cdot\vec{x_I}\right] = u_1 \quad (7.41)$$

Apply $\frac{\partial}{\partial\beta}$:

$$\frac{\partial}{\partial\beta}\left[\frac{\partial}{\partial\alpha}\left[\vec{u}^T\cdot\vec{x_I}\right]\right] = 0 \quad (7.42)$$

With the use of the above intermezzo, we can integrate eq. 7.36 with respect to β:

$$\frac{\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right]}{p_I^n} = -\frac{1}{2}\Big(2M_{11}x_i^{in} + M_{12}x_i^{out} + M_{21}x_i^{out} + M_{11}x_j^{in} + M_{12}x_j^{out} + M_{11}x_j^{in} + M_{21}x_j^{out}\Big) - u_1 \quad (7.43)$$

$$\frac{\partial}{\partial\alpha}\left[p_I^n(x_I)\right] = -\frac{p_I^n}{2}\Big(2M_{11}x_i^{in} + M_{12}x_i^{out} + M_{21}x_i^{out} + M_{11}x_j^{in} + M_{12}x_j^{out} + M_{11}x_j^{in} + M_{21}x_j^{out}\Big) - u_1p_I^n. \quad (7.44)$$

Where u_1 is a constant that appears upon integrating with respect to β.

Now we will integrate with respect to α:

$$p_I^n = e^{-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} - \vec{u}^T \cdot \vec{x_I}} = 1 - p_I^s. \tag{7.45}$$

Thus, we can conclude that the probability that there exists a self-loop on cluster I (consisting of nodes i and j) is given by

$$p_I^s = 1 - e^{-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} - \vec{u}^T \cdot \vec{x_I}} \tag{7.46}$$

7.2 Bound on the parameters

The probabilities given in equations 2.11-2.14 must be non-negative and bounded by 1. This results in a set of conditions for the values of the parameters M, A, and u. In what follows we assume the values of the vector elements of $\vec{x}$ and $\vec{y}$ to be always positive.

7.2.1 Boundaries on p_{Ij}^{00}

Non-negative probability p^{00} that is bounded by 1 will imply the following:

$$\begin{aligned} p^{00} &\leq 1 \\ e^{-\vec{x}^T M \vec{y}} &\leq 1 \\ x^k y^l M^{kl} &\geq 0 \\ 0 &\leq M \end{aligned} \tag{7.47}$$

and of course, after examining the shape of the function $e^{-\vec{x}^T M \vec{y}}$, we can conclude that there is no real value of M which forces p^{00} to be smaller than 0. Thus, we conclude the following condition on the matrix M:

$$0 \leq M < \infty. \tag{7.48}$$

7.2.2 Boundaries on p_{Ij}^{10} and p_{Ij}^{01}

First we examine $0 \leq p^{10}$. To simplify this task, we define $K = M - A$ which we will use from here after.

$$\begin{aligned} 0 &\leq e^{-\vec{x}^T K^T \vec{y}} - e^{-\vec{x}^T M \vec{y}} \\ \vec{x}^T M \vec{y} &\geq \vec{x}^T K^T \vec{y} \\ M &\geq K^T \\ M &\geq M^T - A^T = M - A^T \end{aligned} \tag{7.49}$$

from which we can conclude that $0 \leq A^T$.

Next, we will examine $p^{01} \leq 1$:

$$\begin{aligned} e^{-\vec{x}^T K^T \vec{y}} - e^{-\vec{x}^T M \vec{y}} &\leq 1 \\ e^{-\vec{x}^T M \vec{y}} \left(e^{\vec{x}^T A^T \vec{y}} - 1 \right) &\leq 1 \\ e^{\vec{x}^T M \vec{y}} &\geq e^{\vec{x}^T A^T \vec{y}} - 1 \\ \lim_{x^k y^l \to \infty} e^{\vec{x}^T M \vec{y}} &\geq \lim_{x^k y^l \to \infty} \left(e^{\vec{x}^T A^T \vec{y}} - 1 \right) \\ e^{\vec{x}^T M \vec{y}} &\geq e^{\vec{x}^T A^T \vec{y}} \\ A^T &\leq M. \end{aligned} \tag{7.50}$$

With these conclusions together with the bound on M found by conditioning p^{00} we conclude the following:

$$0 \leq A^T \leq M < \infty. \tag{7.51}$$

Similarly, normalizing p^{01} $\left(p^{01} = e^{-\vec{x}^T K \vec{y}} - e^{-\vec{x}^T M \vec{y}} \right)$ yields

$$0 \leq A \leq M < \infty. \tag{7.52}$$

7.2.3 Boundaries on p_{Ij}^{11}

First we examine the inequality $0 \leq p^{11}$:

$$\begin{aligned} 0 &\leq 1 - e^{-\vec{x}^T K^T \vec{y}} - e^{-\vec{x}^T K \vec{y}} + e^{-\vec{x}^T M \vec{y}} \\ 0 &\leq 1 - e^{-\vec{x}^T M^T \vec{y}} e^{\vec{x}^T A^T \vec{y}} - e^{-\vec{x}^T M \vec{y}} e^{\vec{x}^T A \vec{y}} + e^{-\vec{x}^T M \vec{y}} \\ 1 &\geq e^{-\vec{x}^T M \vec{y}} \left(e^{\vec{x}^T A^T \vec{y}} + e^{\vec{x}^T A \vec{y}} - 1 \right) \\ e^{\vec{x}^T M \vec{y}} &\geq e^{\vec{x}^T A^T \vec{y}} + e^{\vec{x}^T A \vec{y}} - 1. \end{aligned} \tag{7.53}$$

Let's rewrite this inequality by first defining some new parameters: let $\vec{x} = |x|\hat{x}$, $\vec{y} = |y|\hat{y}$ such that $z \equiv |x||y|$. This gives

$$e^{z\hat{x}^T M \hat{y}} \geq e^{z\hat{x}^T A^T \hat{y}} + e^{z\hat{x}^T A \hat{y}} - 1 \qquad \forall \quad z \geq 0 \quad \text{and} \quad \theta \in [0, \frac{\pi}{2}] \tag{7.54}$$

where $\hat{x} \cdot \hat{y} = cos(\theta)$. Now assume that $\hat{x}$ and $\hat{y}$ are fixed. This allows us to define the following parameters:

$$\left.\begin{aligned} max &= max(\hat{x}^T A \hat{y}, \quad \hat{x}^T A^T \hat{y}) \\ min &= min(\hat{x}^T A \hat{y}, \quad \hat{x}^T A^T \hat{y}) \end{aligned}\right\} \Rightarrow \delta \equiv max - min \tag{7.55}$$

and for the sake of notation we define $m = \hat{x}^T M \hat{y}$. The above inequality reduces to

$$\begin{aligned} e^{z\cdot max} + e^{z\cdot min} &\leq e^{z\cdot m} + 1 \\ e^{z\cdot max}\left(1 + e^{-z\cdot\delta}\right) &\leq e^{z\cdot m} + 1 \\ e^{z\cdot(max-m)} &\leq \frac{1 + e^{-z\cdot m}}{1 + e^{-z\cdot\delta}}. \end{aligned} \tag{7.56}$$

We can examine two cases: 1) $\delta = 0$ (meaning that $A = A^T$) and 2) $\delta > 0$ (meaning that $A \neq A^T$). In the first scenario the above inequality reduces to

$$e^{z\cdot(max-m)} \leq \frac{1}{2}(1 + e^{-z\cdot m}) \tag{7.57}$$

which naturally holds for $z = 0$. Also we need

$$\frac{d}{dz}|_{z=0}\left(e^{z\cdot(max-m)}\right) \leq \frac{d}{dz}|_{z=0}\left(\frac{1}{2}(1 + e^{-z\cdot m})\right) \tag{7.58}$$

to hold which is true for $A \leq \frac{1}{2}M$ (and since $\delta = 0$ also for $A^T \leq \frac{1}{2}M$). Now it can be shown that the inequality $e^{z\cdot(max-m)} \leq \frac{1}{2}(1 + e^{-z\cdot m})$ also holds for $z > 0$ by plugging in the minimum and maximum allowed values for matrix A, namely $A = 0$ and $A = \frac{1}{2}M$ respectively. Setting $A = 0$ we obtain the inequality $e^{-z\cdot m} \leq \frac{1}{2}(1 + e^{-z\cdot m}) \Longrightarrow e^{-z\cdot m} \leq 1$ which holds for all $z \geq 0$ and $m > 0$. Returning to the inequality $e^{z\cdot(max-m)} \leq \frac{1}{2}(1 + e^{-z\cdot m})$ setting $A = \frac{1}{2}M$ we get $e^{\frac{1}{2}z\cdot m} \leq \frac{1}{2} + \frac{1}{2}e^{z\cdot m}$ which obviously is true for $z \geq 0$ and $m > 0$. To conclude, for $\delta = 0$ our original inequality holds under the condition that

$$A \leq \frac{1}{2}M \qquad \left(\text{ or } A^T \leq \frac{1}{2}M\right). \tag{7.59}$$

Now we set $\delta > 0$ (in other words $A \neq A^T$). Again we need

$$\frac{d}{dz}|_{z=0}\left(e^{z\cdot(max-m)}\right) \leq \frac{d}{dz}|_{z=0}\left(\frac{1 + e^{-z\cdot m}}{1 + e^{-z\cdot\delta}}\right) \tag{7.60}$$

to hold. This inequality reduces to $max - m \leq \frac{1}{2}(\delta - m)$ which means that $A + A^T \leq M$ must be satisfied. This turns out to be the final requirement to satisfy $0 \leq p_{Ij}^{11}$.

Examining the inequality $p^{11} \leq 1$:

$$1 - e^{-\vec{x}^T\cdot M\cdot\vec{y}}\left(e^{\vec{x}^T\cdot A^T\cdot\vec{y}} + e^{\vec{x}^T\cdot A\cdot\vec{y}} - 1\right) \leq 1 \tag{7.61}$$

The easiest way to check whether this inequality holds is by numerically analyzing it. In order to do this we need to rewrite this inequality to be depended on z (similar to the above inequalities).

$$1 - e^{-z\hat{x}^T \cdot M \cdot \hat{y}} \left(e^{z\hat{x}^T \cdot A^T \cdot \hat{y}} + e^{z\hat{x}^T \cdot A \cdot \hat{y}} - 1 \right) \leq 1 \tag{7.62}$$

where again we will assume that $\hat{x}$ and $\hat{y}$ are fixed. We will also assume that M and A are fixed while $0 \leq A + A^T \leq M < \infty$ holds. Than we plot p^{11} as function of z where z lives on a wide domain. The resulting plot is given in figure 7.1 where we see that p^{11} grows asymptotically to 1. Therefore, the overall conclusion is that 7.61 is true as long as $0 \leq A + A^T \leq M < \infty$ holds.

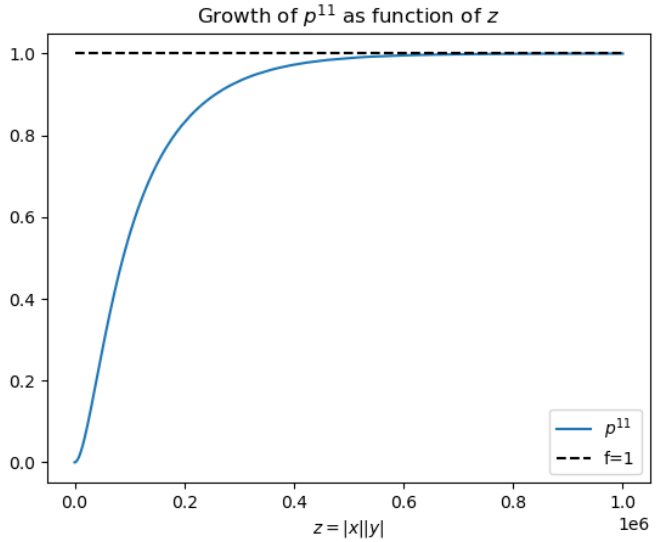


Figure 7.1: *Probability p^{11} as function of z growing asymptotically to 1*

7.2.4 Boundaries on p^{11}_{Ij} $(j \in I) \equiv p^s_I$

We know that p^s_I is given by

$$p^s_I = 1 - e^{-\frac{1}{2}\vec{x}_I^T \cdot M \cdot \vec{x}_I \vec{u}^T \cdot \vec{x}_I}. \tag{7.63}$$

Normalizing this expression leads to

$$\begin{aligned} 0 \leq p^s_I &\leq 1 \\ 0 \leq 1 - e^{-\frac{1}{2}\vec{x}_I^T \cdot M \cdot \vec{x}_I - \vec{u}^T \cdot \vec{x}_I} &\leq 1 \\ 0 \leq e^{-\frac{1}{2}\vec{x}_I^T \cdot M \cdot \vec{x}_I - \vec{u}^T \cdot \vec{x}_I} &\leq 1 \end{aligned} \tag{7.64}$$

where due to the nature of the exponential, the inequality $0 \leq e^{-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} - \vec{u}^T \cdot \vec{x_I}}$ will always be met resulting in the following upper-bound on u: $u < \infty$. The lower-bound will be determined by the inequality $e^{-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} - \vec{u}^T \cdot \vec{x_I}} \leq 1$:

$$\begin{aligned} e^{-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} - \vec{u}^T \cdot \vec{x_I}} &\leq 1 \\ e^{-\vec{u}^T \cdot \vec{x_I}} &\leq e^{\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I}} \\ -\vec{u}^T \cdot \vec{x_I} &\leq \frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} \\ \vec{u}^T \cdot \vec{x_I} &\geq -\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I}. \end{aligned} \tag{7.65}$$

Since $0 \leq M \leq \infty$, and every element of $\vec{x_I}$ is greater than 0 by construction, we can conclude the following regarding the right-hand-sight of the above inequality:

$$-\frac{1}{2}\vec{x_I}^T \cdot M \cdot \vec{x_I} \leq 0 \leq \vec{u}^T \cdot \vec{x_I}. \tag{7.66}$$

Using again the argument that $\vec{x_I}$ is always positive ($\vec{x_I} \in \mathbb{R}^2$), we conclude that

$$0 \leq \vec{u} < \infty \tag{7.67}$$

which means that every element of $\vec{u}$ lies in the interval $[0, \infty)$.

7.3 Additional Empirical Data Plots

In this section we present several additional plots that give additional insights into the empirical data. In figures 7.2, 7.3 and 7.4 we present heat maps scatter plots to show how the degrees and strengths are related to each other. In figure 7.2 the network contains self-loop edges and the rest-of-the-world node, in figure 7.3 the network does not contain self-loop edges while it does have the rest-of-the-world node, and finally in figure 7.4 the network does not contain the rest-of-the-world node or the self-loop edges.

Further, more in figures 7.5 and 7.6 we present the cumulative incoming and outgoing number of payments distributions *per node*, and the cumulative number of payments distribution *per edge*.

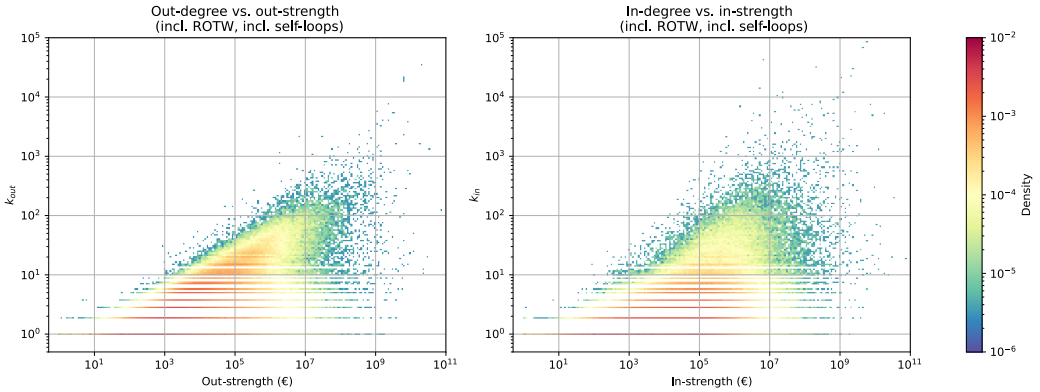


Figure 7.2: *Out-degree vs. out-strength (left) and in-degree vs. in-strength (right) for a financial transaction network including connections towards and from the ROTW-node and including self-loop connections. Both plotted on double log scales including a log scale for the density scale.*

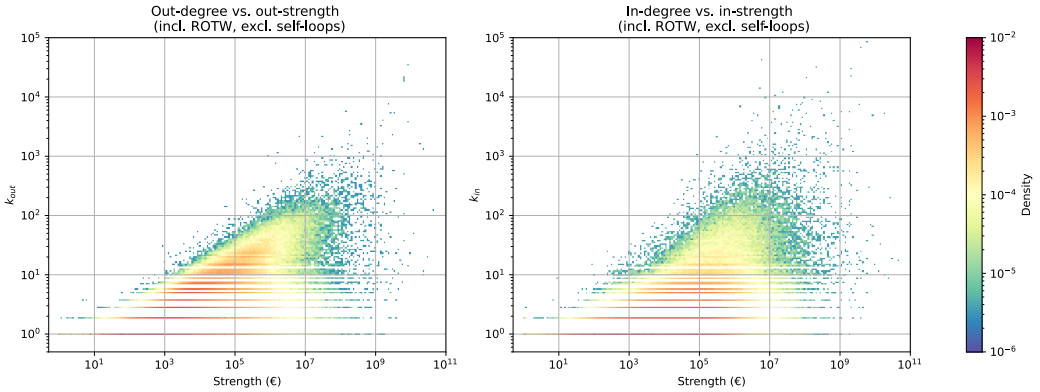


Figure 7.3: *Out-degree vs. out-strength (left) and in-degree vs. in-strength (right) for a financial transaction network including connections towards and from the ROTW-node while excluding self-loop connections. Both plotted on double log scales including a log scale for the density scale.*

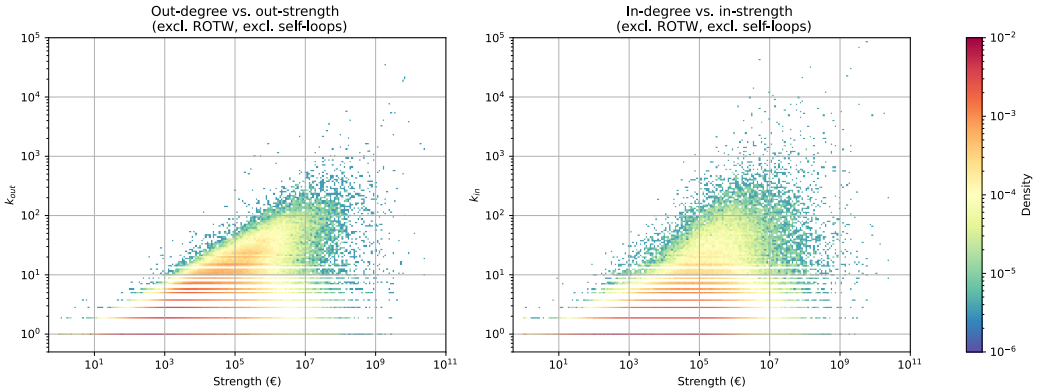


Figure 7.4: *Out-degree vs. out-strength (left) and in-degree vs. in-strength (right) for a financial transaction network excluding connections towards and from the ROTW-node and excluding self-loop connections. Both plotted on double log scales including a log scale for the density scale.*

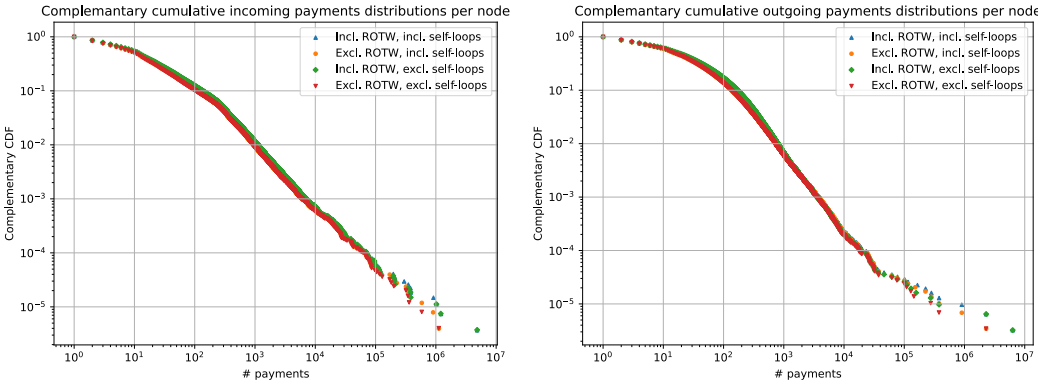


Figure 7.5: *Complementary cumulative incoming payments (left) and outgoing payments (right) distributions for financial transaction networks including the rest-of-the-world node (ROTW) and self-loop connections (blue triangles), excluding ROTW-node and including self-loops (orange dot), including ROTW-node and excluding self-loops (green diamante), and excluding ROTW-node and self-loops (red upside down triangle) on a double log scale.*

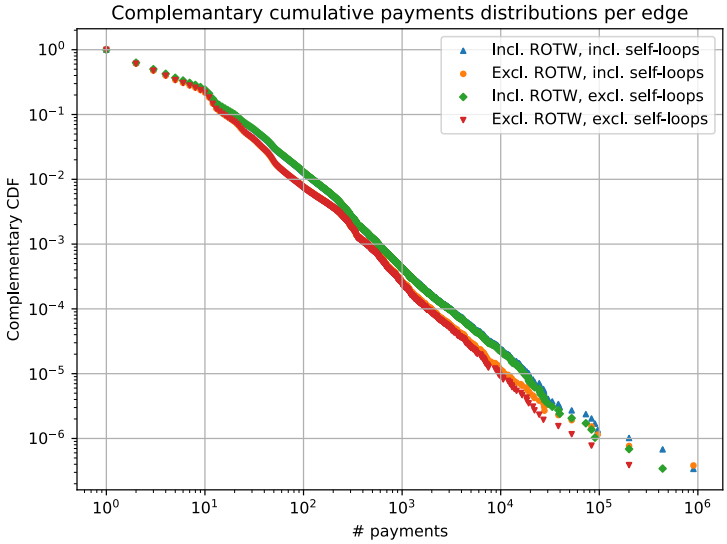
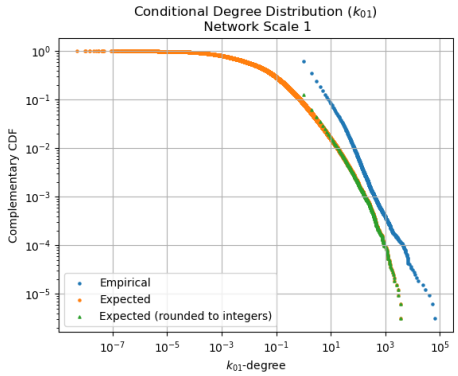


Figure 7.6: *Complementary cumulative distribution of the number of payments (number of transactions) per edge for financial transaction networks including the rest-of-the-world node (ROTW) and self-loop connections (blue triangles), excluding ROTW-node and including self-loops (orange dot), including ROTW-node and excluding self-loops (green diamante), and excluding ROTW-node and self-loops (red upside down triangle) on a double log scale.*

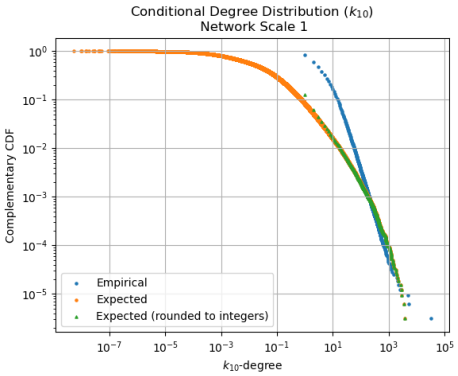
7.4 Empirical and Expected Degree Distribution Plots

In this section we provide the figures containing the cumulative degree distributions described in chapter 5. Subsections 7.4.1, 7.4.2, 7.4.3 and 7.4.4 provide the plots for the networks at scales 1, 2, 3 and 4 respectively.

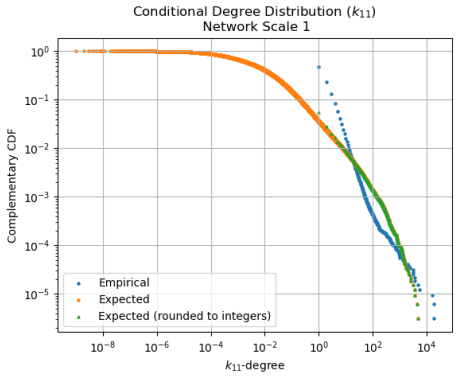
7.4.1 Network Scale 1



(a) *Cumulative expected conditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional in-degree distribution (blue) on a double log scale.*

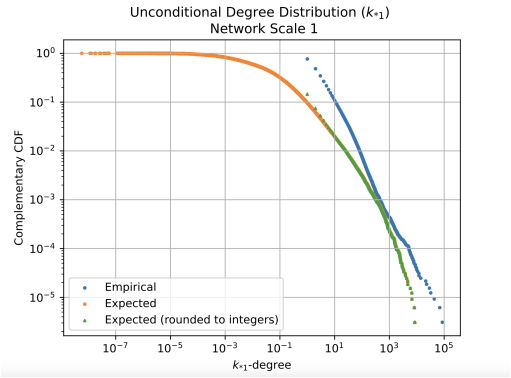


(b) *Cumulative expected conditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional out-degree distribution (blue) on a double log scale.*

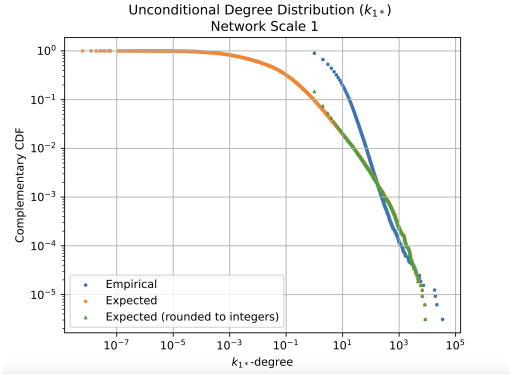


(c) *Cumulative expected conditional reciprocated-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional reciprocated-degree distribution (blue) on a double log scale.*

Figure 7.7: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 1.*



(a) *Cumulative expected unconditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional in-degree distribution (blue) on a double log scale.*



(b) *Cumulative expected unconditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional out-degree distribution (blue) on a double log scale.*

Figure 7.8: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 1.*

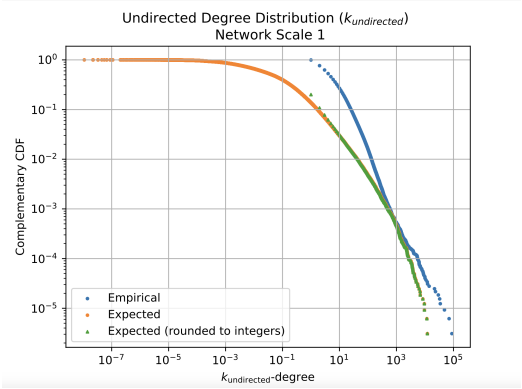
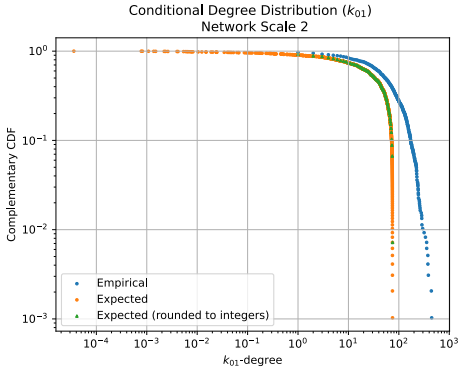
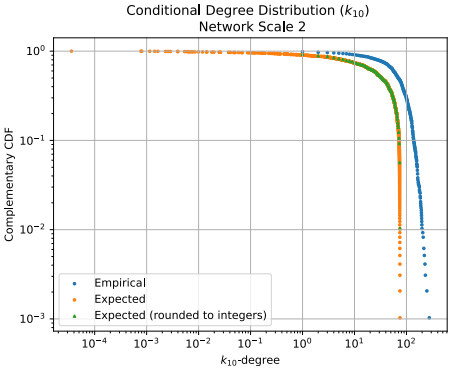


Figure 7.9: *Cumulative expected undirected degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical undirected degree distribution (blue) on a double log scale for the transaction network at scale 1.*

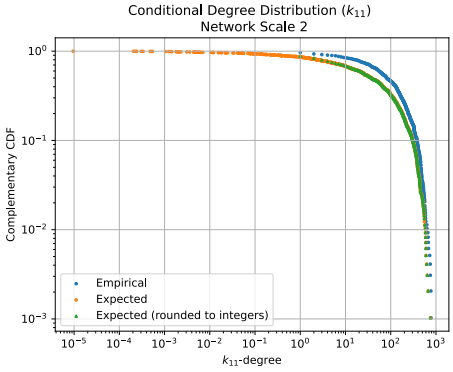
7.4.2 Network Scale 2



(a) *Cumulative expected conditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional in-degree distribution (blue) on a double log scale.*

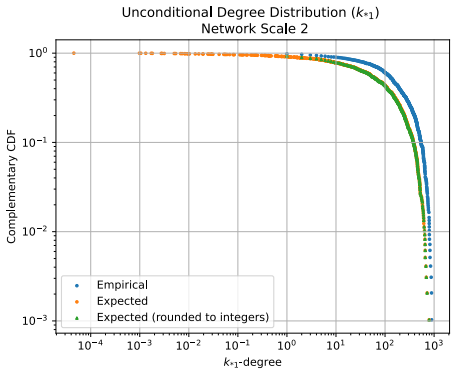


(b) *Cumulative expected conditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional out-degree distribution (blue) on a double log scale.*

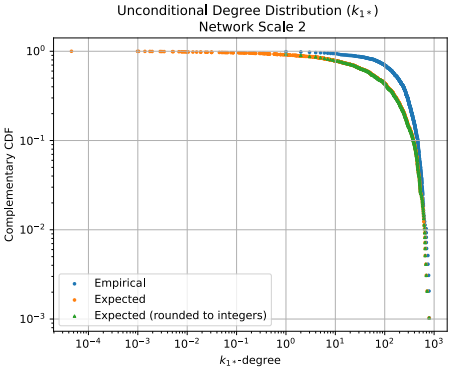


(c) *Cumulative expected conditional reciprocated-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional reciprocated-degree distribution (blue) on a double log scale.*

Figure 7.10: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 2.*



(a) *Cumulative expected unconditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional in-degree distribution (blue) on a double log scale.*



(b) *Cumulative expected unconditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional out-degree distribution (blue) on a double log scale.*

Figure 7.11: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 2.*

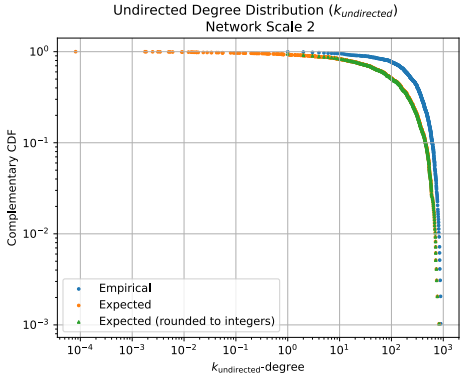
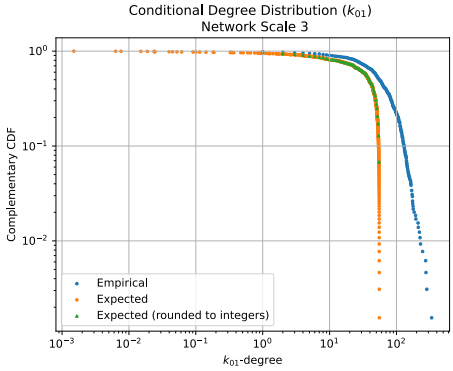
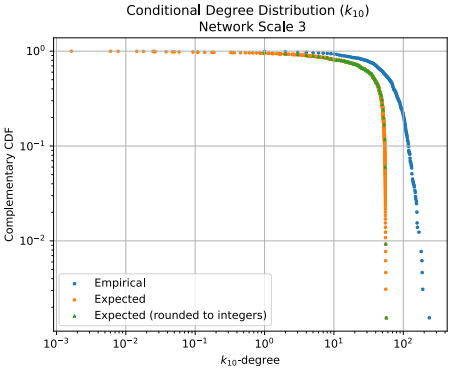


Figure 7.12: *Cumulative expected undirected degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical undirected degree distribution (blue) on a double log scale for the transaction network at scale 2.*

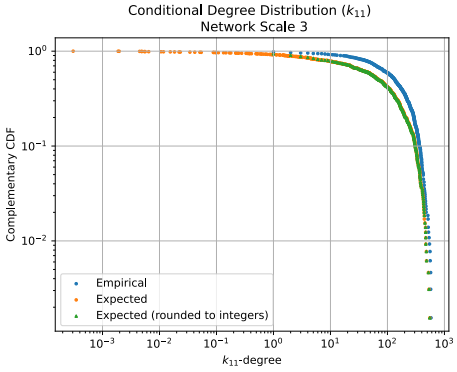
7.4.3 Network Scale 3



(a) *Cumulative expected conditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional in-degree distribution (blue) on a double log scale.*

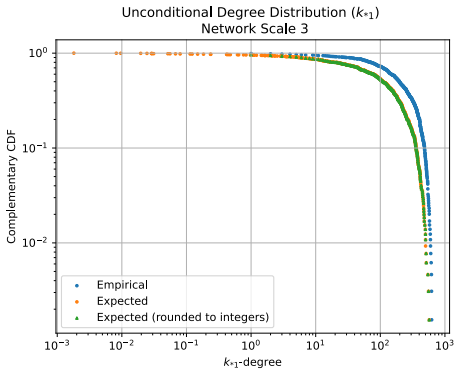


(b) *Cumulative expected conditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional out-degree distribution (blue) on a double log scale.*

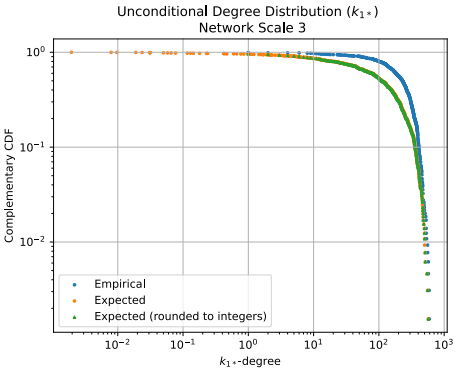


(c) *Cumulative expected conditional reciprocated-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional reciprocated-degree distribution (blue) on a double log scale.*

Figure 7.13: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 3.*



(a) *Cumulative expected unconditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional in-degree distribution (blue) on a double log scale.*



(b) *Cumulative expected unconditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional out-degree distribution (blue) on a double log scale.*

Figure 7.14: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 3.*

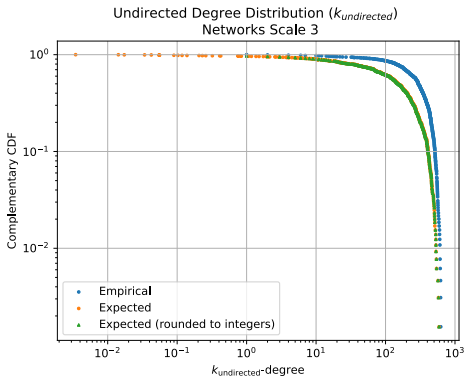
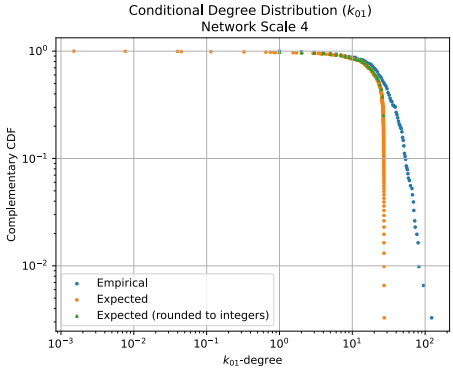
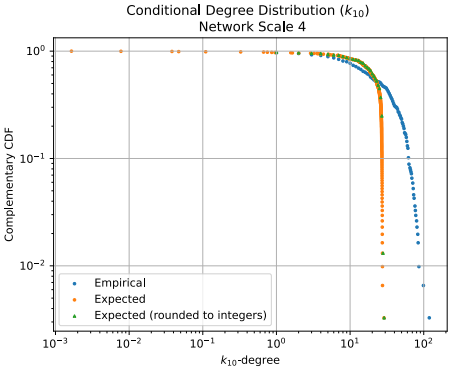


Figure 7.15: *Cumulative expected undirected degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical undirected degree distribution (blue) on a double log scale for the transaction network at scale 3.*

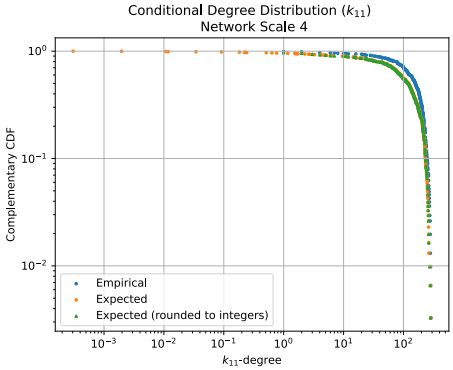
7.4.4 Network Scale 4



(a) *Cumulative expected conditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional in-degree distribution (blue) on a double log scale.*

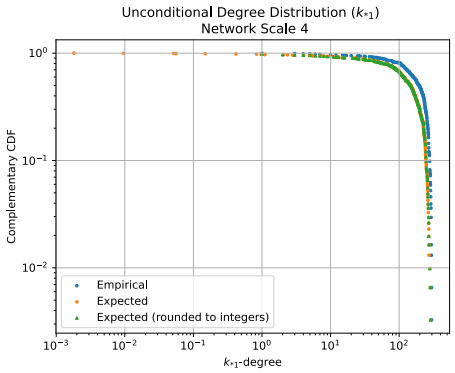


(b) *Cumulative expected conditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional out-degree distribution (blue) on a double log scale.*

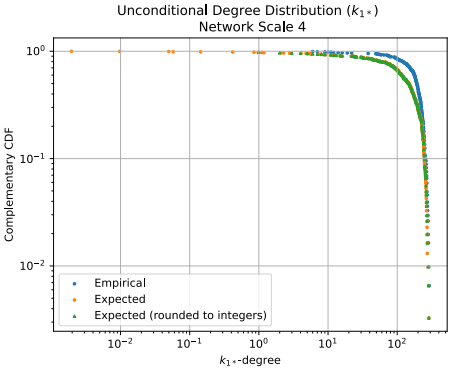


(c) *Cumulative expected conditional reciprocated-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical conditional reciprocated-degree distribution (blue) on a double log scale.*

Figure 7.16: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 4.*



(a) *Cumulative expected unconditional in-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional in-degree distribution (blue) on a double log scale.*



(b) *Cumulative expected unconditional out-degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical unconditional out-degree distribution (blue) on a double log scale.*

Figure 7.17: *Expected (orange and green) and empirical (blue) conditional degree distributions on double log scales for the transaction network at scale 4.*

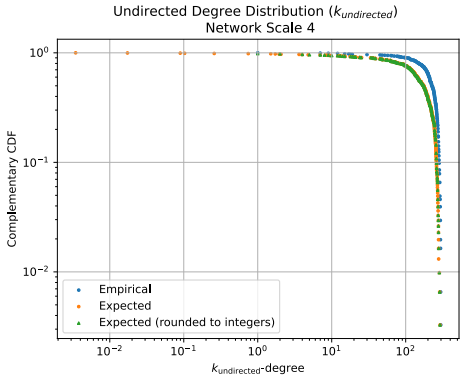


Figure 7.18: *Cumulative expected undirected degree distribution (orange) rounded to integer valued degrees (green) and cumulative empirical undirected degree distribution (blue) on a double log scale for the transaction network at scale 4.*

7.5 Empirical and Expected Weight Distribution Plots

In this section we provide the figures containing the cumulative weight distributions plots described in chapter 5. Subsections 7.5.1, 7.5.2, 7.5.3

and 7.5.4 provide the plots for the networks at scales 1, 2, 3 and 4 respectively.

7.5.1 Network Scale 1

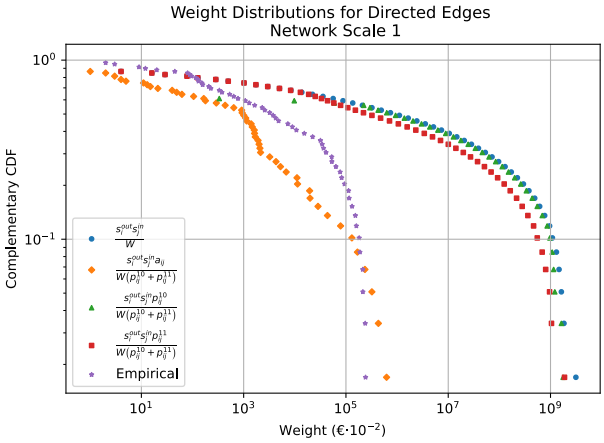


Figure 7.19: *Expected (orange diamonds, green triangles and red squares), empirical (purple stars) and average (blue dots) weight distribution (in units of $€\cdot10^{-2}$) for directed edges for the transaction network on scale 1 on a double log scale.*

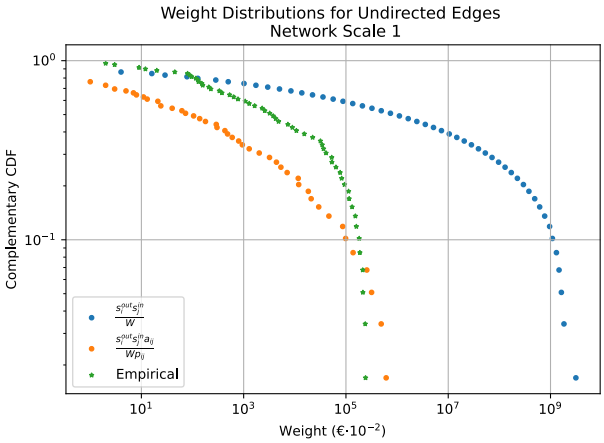


Figure 7.20: *Expected (orange dots), empirical (green stars) and average (blue dots) weight distribution (in units of $€\cdot10^{-2}$) for undirected edges for the transaction network on scale 1 on a double log scale.*

Version of June 10, 2024– Created June 10, 2024 - 13:16

7.5.2 Network Scale 2

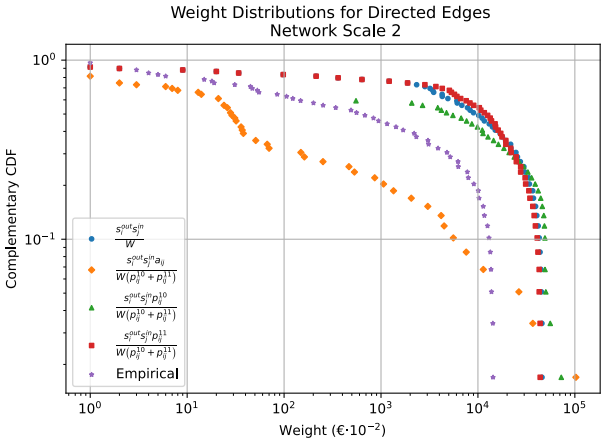


Figure 7.21: *Expected (orange diamond, green triangle and red squares), empirical (purple stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for directed edges for the transaction network on scale 2 on a double log scale.*

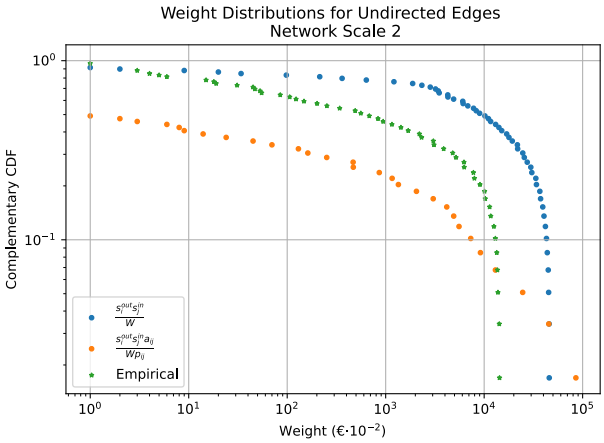


Figure 7.22: *Expected (orange dots), empirical (green stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for undirected edges for the transaction network on scale 2 on a double log scale.*

7.5.3 Network Scale 3

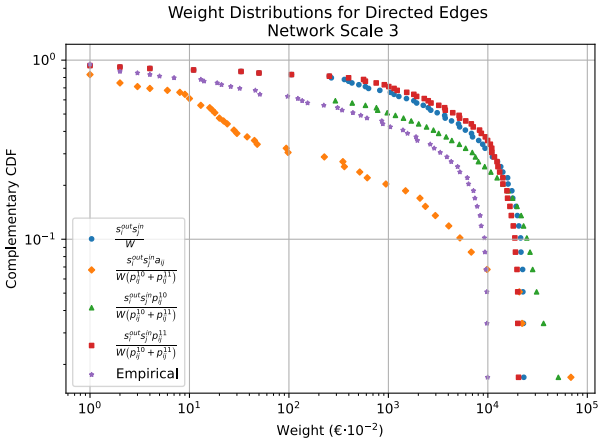


Figure 7.23: *Expected (orange diamond, green triangle and red squares), empirical (purple stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for directed edges for the transaction network on scale 3 on a double log scale.*

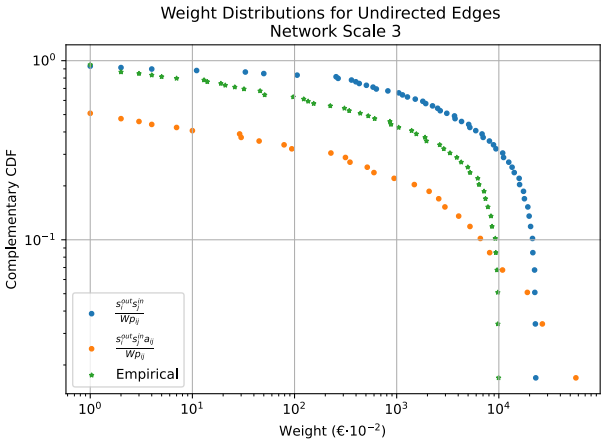


Figure 7.24: *Expected (orange dots), empirical (green stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for undirected edges for the transaction network on scale 3 on a double log scale.*

7.5.4 Network Scale 4

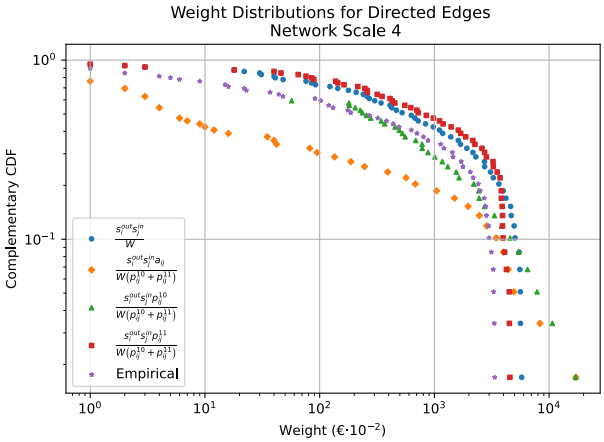


Figure 7.25: *Expected (orange diamond, green triangle and red squares), empirical (purple stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for directed edges for the transaction network on scale 4 on a double log scale.*

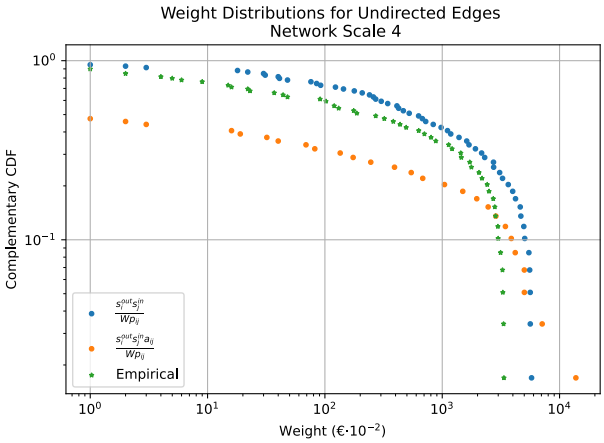


Figure 7.26: *Expected (orange dots), empirical (green stars) and average (blue dots) weight distribution (in units of €·10^{-2}) for undirected edges for the transaction network on scale 4 on a double log scale.*

Bibliography

[1] "Homepage ING Bank N.V." https://www.ing.com/Home.htm. Accessed: 01-2024.

[2] L. N. Ialongo, C. de Valk, E. Marchese, F. Jansen, H. Zmarrou, T. Squartini, and D. Garlaschelli, "Reconstructing firm-level interactions in the dutch input–output network from production constraints," *Scientific reports*, vol. 12, no. 1, p. 11847, 2022.

[3] N. Goldenfeld, *Lectures on phase transitions and the renormalization group*. CRC Press, 2018.

[4] E. Garuccio, M. Lalli, and D. Garlaschelli, "Multiscale network renormalization: scale-invariance without geometry," *Physical Review Research*, vol. 5, no. 4, p. 043101, 2023.

[5] "Homepage CBS." https://www.cbs.nl/. Accessed: 01-2024.

[6] M. Di Vece, F. P. Pijpers, and D. Garlaschelli, "Commodity-specific triads in the dutch inter-industry production network," *Scientific Reports*, vol. 14, no. 1, p. 3625, 2024.

[7] D. Garlaschelli and M. I. Loffredo, "Patterns of link reciprocity in directed networks," *Physical review letters*, vol. 93, no. 26, p. 268701, 2004.

[8] M. Lalli and D. Garlaschelli, "Geometry-free renormalization of directed networks: scale-invariance and reciprocity," *arXiv preprint arXiv:2403.00235*, 2024.

[9] L. P. Kadanoff, *Statistical physics: statics, dynamics and renormalization*. World Scientific, 2000.

[10] "ING Wholesale Banking Advanced Analytics." https://www.ingwb.com/en/insights/empowering-with-advanced-analytics/what-does-ing-wholesale-banking-advanced-analytics-do. Accessed: 02-2024.

[11] "NAICS-codes." https://www.census.gov/programs-surveys/economic-census/year/2022/guidance/understanding-naics.html#:~:text=The%20North%20American%20Industry%20Classification,to%20the%20U.S.%20business%20economy. Accessed: 02-2024.

[12] W. C. Thacker, "The role of the hessian matrix in fitting models to measurements," *Journal of Geophysical Research: Oceans*, vol. 94, no. C5, pp. 6177–6196, 1989.

Acknowledgement

In this section I would like to thank Dr. Diego Garlaschelli for the amazing opportunity this project was. I am very gratefully he went out on a limb to setup a research project outside of the university.

I also want to thank Dr. Fabian Jansen and the rest of the wholesale banking data science team. They made me feel welcome at ING which encouraged me to learn as much as possible during my time there at the office. Dr. Jansen especially invested a lot of time to supervise me on a daily basis. Thanks to him I learned so much about the financial industry and data science which overall resulted in a fantastic learning experience during the coarse of this project.