AMERICAN UNIVERSITY OF BEIRUT

INFERING UNDERLYING NETWORKS FROM TIME SERIES OF DYNAMICAL SYSTEMS AND EVALUATING GLOBAL BALANCE

by
ALI SHAWKI BADEREDDINE

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Graduate Program in Computational Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
May 2024

AMERICAN UNIVERSITY OF BEIRUT

INFERING UNDERLYING NETWORKS FROM TIME SERIES OF DYNAMICAL SYSTEMS AND EVALUATING GLOBAL BALANCE

by
ALI SHAWKI BADEREDDINE

Approved by:

Dr. Sara Najem, Assistant Professor Physics	Advisor
Dr. Amer E. Mouawad, Assistant Professor Computer Science	Member of Committee
Dr. Izzat El Hajj, Assistant Professor Computer Science	Member of Committee

Date of thesis defense: May 7, 2024

Acknowledgements

Completing this research has been a journey filled with frustration and non-linearities. Fortunately, I found myself among a subset of the partition comprising incredibly supportive individuals and a cohesive community.

I am immensely grateful for the support of my advisor, Dr. Sara Najem, who has guided me towards this exploration and has continually provided valuable advice since I began my master's degree—and even before.

I am also thankful for the opportunity to work with Dr. Amer E. Mouawad and Dr. Izzat El Hajj. Their involvement has not only enriched my knowledge in the field but also challenged me to embrace criticism and strive for the excellence I seek.

I extend special thanks to my father, mother, and sisters, who have supported me from the moment I was born, both literally and figuratively. Their pride in my achievements has been tremendous, motivating me to accomplish even more.

Lastly, I thank my peers and friends, who have eagerly anticipated this moment, for all the belief they have shown in me.

Abstract of the Thesis of

Ali Shawki Badereddine for Master of Science
Major: Computational Science

Title: Infering Underlying Networks from Time Series of Dynamical Systems and Evaluating Global Balance

A complex system's emerging behavior is a result of the interactions of its components. A graph-theoretic representation of it is a network of interactions dictating through differential equations the evolution of the state of the individual components, represented by nodes. These networks can be signed, directed, and weighted. Our first goal is to infer these networks of interactions from time series relying on dynamical systems theory. Our second goal is to characterize these networks, and for this purpose, we rely on multiscale definitions of the frustration indices. We implement algorithms that compute the indices of frustration on multiple levels, explore and address some of the computational bottlenecks, and apply the algorithms to the network inferred from the dynamics.

TABLE OF CONTENTS

Illustrations

Tables

Chapter 1

Introduction to Complex Systems

1.1 Definition of a Complex System

When looking at an exact definition of a complex system, one thing that literature agrees on is that there is no definition. For instance, Binder states that ”defining complexity is frustrating” [1].
However, this does not mean that we do not have symptoms for those complex systems. That is, complexity scientists agree on some signs or indicators that when we observe, there is a high chance that we are looking at a complex system.
For instance, following an example of how a big colony of ants arranges itself to form collective intelligence that researchers do not fully understand, Mitchell defines complex systems as ”an interdisciplinary field of research that seeks to explain how large numbers of relatively simple entities organize themselves, without the benefit of any central controller, into a collective whole that creates patterns, uses information, and, in some cases, evolves and learns” [2].
A treatment of complex system that will prove to be interesting in our context is one that has been followed by Bar-Yam as it follows the emergence of a macroscopic behavior from a series of microscopic behaviors [3].
The provided insights on complex systems should be sufficient to offer a perspective on how complex complex systems are. We might want to summarize complex systems , with some risk of generalization, by saying they are systems with emergent, nonlinear and adaptable behaviors [1]–[4].

1.2 Networks and Interactions

In light of the definition of complex systems provided in the previous section, we can see that interactions between multiple components are a major part of the definition. Mitchell defines a network as a collection of nodes and links [2]. A network is a very commonly used term in our digital age to reference the internet or social media accounts, and it has developed to a verb whereby people ”network” as in they get introduced to more people. However, in our context of complex systems, the nodes

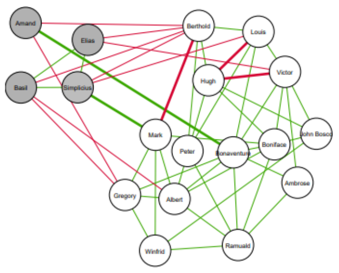
would represent the component (each node represents a component), and the links represent the relations between those components.
In her discussion of network science, Mitchell emphasizes the importance of network thinking as a fundamental approach to complexity science, specifically highlighting the contributions of Barabási in Linked and Watts in Six Degrees. She argues that network thinking prioritizes the analysis of relationships among interacting components over the investigation of the components themselves, providing a more holistic view of complex systems [2], [5]–[7].
Thus, we can see that when we have a system with interacting components, if we find a correlation between those components that models the interaction, we can model the system as a network and consequently be able to study the system. We see in the next section multiple examples of systems that were studied as networks.

1.3 Complex Systems across Disciplines

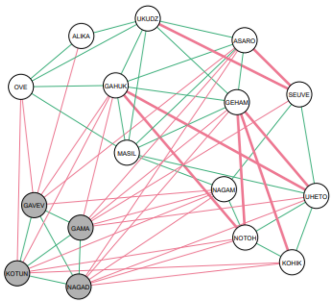
In this section, we explore the different disciplines in which complex systems may be observed. We look at examples from which we could extract useful knowledge by studying those systems.

1.3.1 *Social Science*

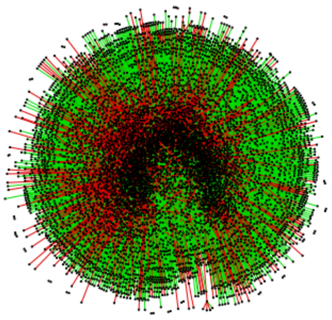
The most natural types of networks to model is the social networks. Those networks would include modeling sociological or political phenomena. The simplest social network one can think about is the Instagram network which models the users as the interacting components, and their means of interactions are the follows or blocks [8].
On a higher level, one can model social interactions as networks like Sampson did with monastery interactions [9], represented in subfigure 1.1a. Furthermore, Read models cultural interactions between the highland tribes of New Guinea as networks [10], represented in subfigure 1.1b.
Networks in social science have been taken to a level to analyze political systems that involve elections. For example, the Wikipedia election dataset involves an interaction between users that vote for other users on whether they approve or disapprove each other as administrators for the page [11], represented in subfigure 1.1c.
Furthermore, Antal, Krapivsky and Redner model the dynamics of friendship and enmity by introducing a marriage and divorce model of social tension to study the evolution of balance in the network [13]. In this model, they consider 2 nodes, one representing a husband and another representing a wife, and a third node would represent a common friend between the two. If the husband and a wife get a divorce, the common friend would have to choose a side to avoid the tension of the resulting divorce.



(a) Representation of the interactions between monks in Sampson's network



(b) Representation of the interactions between highland tribes of New Guinea



(c) Representation of the interactions in the Wikipedia elections network

Figure 1.1: The network representations of multiple social networks in which a green link represents a positive interaction such as friendship while a red link portrays a negative interaction such as enmity [12]

1.3.2 *Finance*

It is very interesting to look at financial examples from a complex system perspective. For example, the stock market has many stocks which can be the interacting components of the system. The correlation between those stocks can exhibit behaviors on the level of the system. For instance, Ferreira et al. model the loss of structural balance in stock markets by establishing a relationship between the correlation between stock returns in several countries [14].

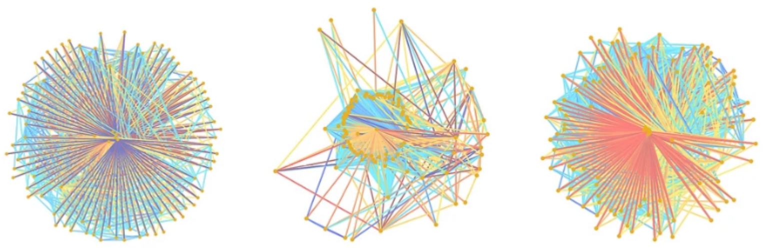


Figure 1.2: Representation of the interaction between stocks in the United states at three different times of economic significance, where the colors of the lines represent the strength of the correlation between the stocks ranging from dark red for most negative to blue for most positive [14]

Another example from finance is the evaluation of financial portfolios. A financial portfolio is a set of investments such as stocks, bonds and cash [15]. Those networks have been modeled by Harary and Kabell in [16] to represent interactions between financial assets based on their correlations.

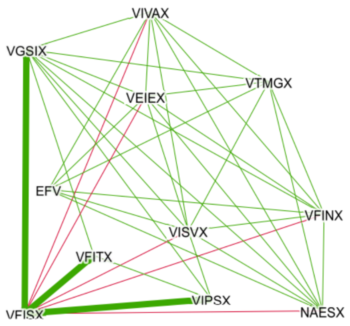


Figure 1.3: Representation of the interaction between financial assets in an Ultimate Buy and Hold portfolio where positive (negative) correlations between the assets are represented in green (red) [12]

1.3.3 *International Relations*

Politics can be viewed as a complex system, in which the interacting components are the countries. For instance, Lai models the conflict in the Middle East as a network of interactions between the Arabs and Israel [17]. Furthermore, Doreian and Mvar use in [18] the Correlates of War dataset (CoW) available in [19] and actively updated in [20] to analyze the balance among the countries.

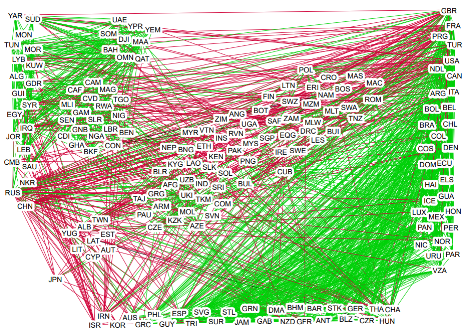
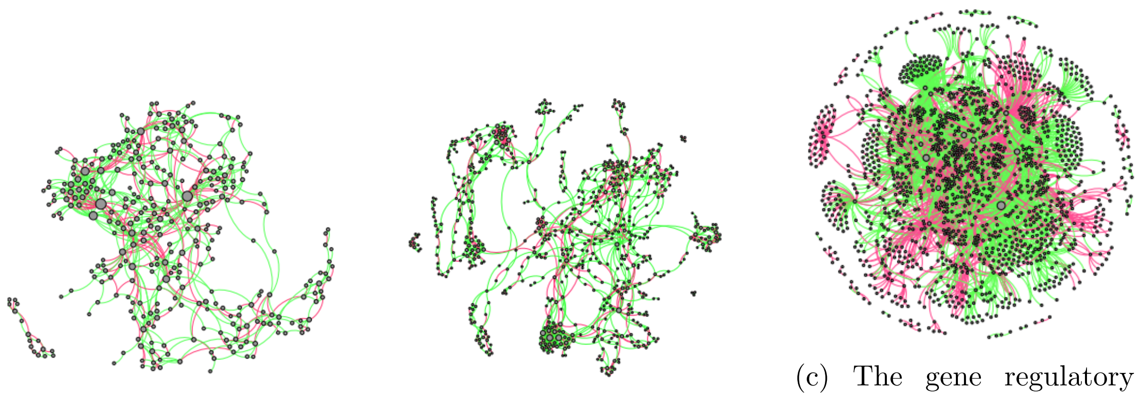


Figure 1.4: Visualization of the Correlates of War Dataset by partitioning the network into two teams [12]

Figure 1.4 demonstrates 2 groups of countries that are on opposite teams: team A on the left and team B on the right. We also see that some countries are somewhat in the center, which means that those countries do not have a clear stand.

1.3.4 *Biology*

Biology is a comprehensive field that involves multiple fields of study, and each of those multiple fields has several instances in which we can appreciate a biological network. As an example, we can model interactions between biological molecules as a network. The biological molecules which would be represented by nodes in our network can be genes, enzymes, proteins and etc... Their interactions denoted by links would be the activation or inhibition [21]. Oda et al. model the epidermal growth factor receptor pathway as a network in [22] and the map of molecular interaction of a macrophage as another network in [23]. Salgado et al. model the gene regulatory network of Escherichia coli as a complex network in [24]. Figure 1.5 show the representations of those networks.



(a) Epidermal growth factor receptor pathway [22]

(b) Molecular interaction map of a macrophage [23]

(c) The gene regulatory network of the Escherichia coli [24]

Figure 1.5: Examples of biological networks in which the nodes are the biological molecules and their interactions are represented by the links in each of the representations [12]

1.3.5 *Physics*

There are many applications for complex systems in physics, because physics has many useful tools that can be used to analyze a system. A very famous example from physics on modeling complex networks is the Ising Model. The Ising Model is a ferromagnetic model that involves the alignment of magnetic material which can either point "up" or "down" [25]. Thus, if we have multiple sites arranged like a grid (check figure 1.6), and each site is inhabited by a certain spin, the interaction between those sites is governed by a magnetic moment, and this would be called an Ising spin glass [26]. In more simple terms, a site would be represented by a node that can either be "up" or "down", and the interaction between the neighboring nodes would be represented by a link. From the Ising model, we can study how local interactions give rise to long range correlations [27]. For instance, if all the nodes were aligned in one direction across the grid, the whole grid would act as one whole magnet. In fact, many physical and non physical systems can be framed as an Ising model. As an example, Bartashevich models collective decision making by means of an Ising Model in which "for" and "against" are the possible states of a node, and the decision is the long range interaction [28]. To visualize the grid as a network, and for even higher dimensions, we can represent the vertices of the grid (i.e. the intersection of a vertical and a horizontal line) as the nodes and the connecting lines as the links. Figure 1.7 shows the network representation of the Ising model, whereby we look at the model from a network thinking perspective.

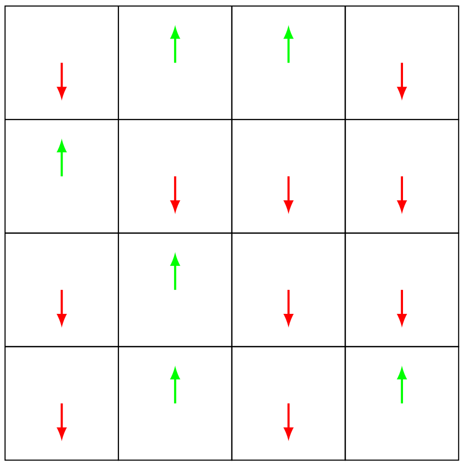
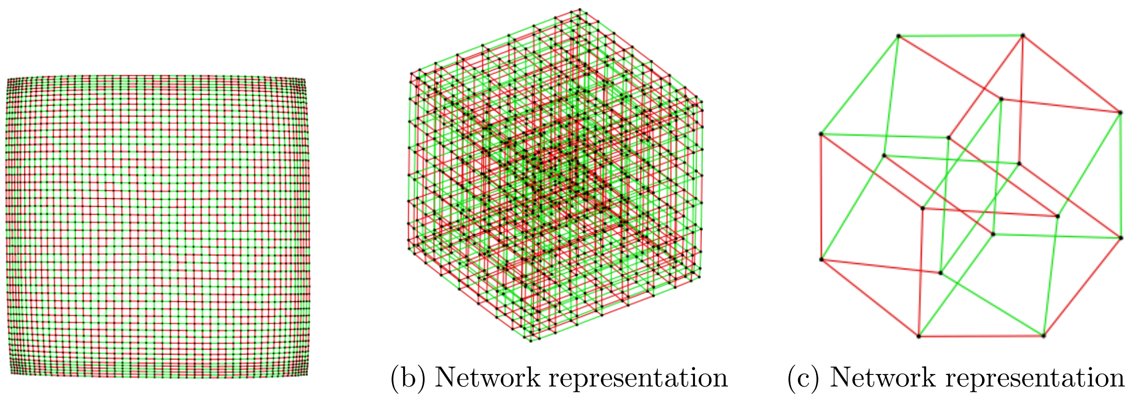


Figure 1.6: 4x4 grid with arrows pointing up or down representing the state



(a) Network representation of the Ising Model in 2D

(b) Network representation of the Ising Model in 3D as a cube

(c) Network representation of the Ising Model in 4D as a hypercube

Figure 1.7: Examples of representations of the Ising Model in multiple dimensions as networks, whereby the color of the edges determines the alignment between two neighboring nodes [12]

We focus on the interactions rather than the nodes themselves, such that if two neighboring nodes have the same alignment, they would have a link that represents a positive correlation and negative otherwise.

Chapter 2

Graph Representations of Networks

Motivation

In this chapter, we characterize networks using notions from graph theory. We then define useful characteristics and storage representations of graphs and that will be relevant throughout our work.

2.1 Definition of a Graph

The most basic definition of a graph given by West states that a graph, denoted as $\mathcal{G}$, is a triple consisting of a vertex set $V(\mathcal{G})$, and edge set $E(\mathcal{G})$ and a relation that associates with each edge two vertices which would constitute its endpoints [29]. We can represent a graph by points/circles (or any other geometrical shape) connected by lines. Each vertex in the vertex set would be represented by some geometrical shape, most commonly a point or a circle, and each two vertices which constitute the endpoints of an edge would be connected with a line. Figure 2.1 represents 3 nodes and 3 vertices. The vertex set of this graph would be $V(\mathcal{G}) = \{A, B, C\}$, while the edge set would be $E(\mathcal{G}) = \{e_1, e_2, e_3\}$. The relation between the vertex and edge sets would be to say that $e_1 = AB$, $e_2 = AC$ and $e_3 = BC$, for which we would say that A and B, A and C, and B and C are the endpoints of edges e_1, e_2 and e_3 respectively.

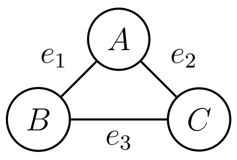


Figure 2.1: A representation of a basic graph that has 3 vertices and 3 edges

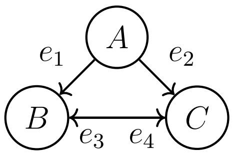


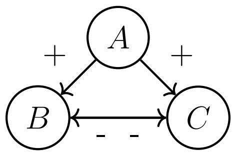
Figure 2.2: A representation of a basic graph that has 3 vertices and 4 edges

2.1.1 *Directions*

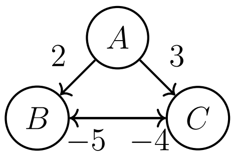
In some cases, we might want the relation between the edges not to be symmetric. That is, in figure 2.1, saying that A and B are the endpoints of the edge e_1 makes no distinction between whether A is linked to B or B is linked to A. This raises the need to define a directed graph. A directed graph, also known as a digraph, $\mathcal{G}$ is a triple consisting of a vertex set $V(\mathcal{G})$, an edge set $E(\mathcal{G})$, and a function assigning each edge an ordered pair of vertices [29]. In this case, the endpoints of an edge would be called a head or a source, and a tail or a destination. For a directed edge, the edge would connect the source/head to the destination/tail. Figure 2.2 makes it clear that A is linked to B but the opposite is not true. However, B and C are linked to one another in both directions. This means that the vertex set is the same for figures 2.1 and 2.2, however the edge set is not the same.

2.1.2 *Signs and Weights*

Sometimes we can associate values with each of the edges in the edge set $E(\mathcal{G})$. We define a function σ that maps edges to a value. If the values that σ maps the edge to are $\{-1, +1\}$, then this would mean that the edge is associated with either a positive or a negative sign, as represented in subfigure 2.3a [30]. If the function σ maps the edges to any set of values which is a subset of $\mathbb{R}$, then we say the edges are associated with weights as shown in subfigure 2.3b [31].



(a) Signed graph



(b) Weighted graph

Figure 2.3: A representation of a signed and a weighted graph

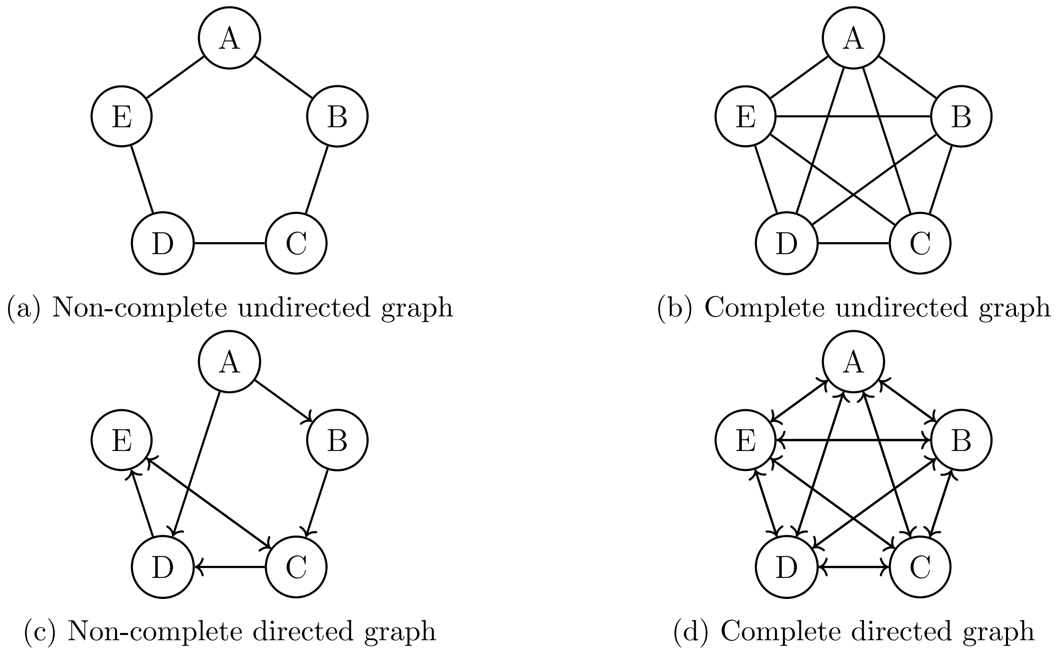
2.1.3 *Degree*

Consider a vertex $v \in V(\mathcal{G})$ of a graph $\mathcal{G}$. We define the degree of $\mathcal{G}$, in case the graph is undirected, denoted as $d(v)$, as the number of edges for which v is an endpoint [29]. For example, in figure 2.1, the degree would be $d(A) = d(B) = d(C) = 2$ because each of the vertices is an endpoint of two edges.
In case the graph is directed, we distinguish between the indegree and the outdegree

of the vertex. The outdegree of the vertex v, denoted as $d^+(v)$ is the number of edges for which v is the tail/destination, whereas the indegree of the vertex v, denoted as $d^-(v)$ is the number of edges for which v is the head/source [29]. In figure 2.2, $d^+(A) = 0$, $d^-(A) = 2$, $d^+(B) = 2$, $d^-(B) = 1$, $d^+(C) = 2$, and $d^-(C) = 1$.

2.1.4 *Complete Graphs*

A graph $\mathcal{G}$ is said to be complete if for any $u, v \in V(\mathcal{G})$ there exists an edge $e \in E(\mathcal{G})$ such that u, v are its endpoints. For a directed graph, the edge must exist in both directions [32].



(a) Non-complete undirected graph

(b) Complete undirected graph

(c) Non-complete directed graph

(d) Complete directed graph

Figure 2.4: Complete vs non-complete graphs for the cases of undirected and directed graphs

In Figure 2.4, we see that the column on the left does not depict complete graphs, because if we take nodes B and E, there is no edge between them, unlike the graphs in the column on the right.

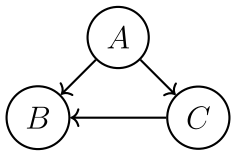
2.2 Graph Theoretic Formulation of Networks

Consider a signed and directed graph $\mathcal{G} = (V, E, \sigma)$ where V is the set of vertices. We also consider that $|V| = n$ and E is the set of edges such that $|E| = m$. Furthermore, for $e \in E$, we have $\sigma(e) = -1, 1$ which maps an edge to a sign. It follows that the set of positive edges is denoted by E^+ and that of negative edges is denoted by E^-, such that $E = E^+ \cup E^-$. Consequently, we denote by m^+ the number of positive edges $|E^+|$ and by m^- the number of negative edges $|E^-|$, such that $m = m^+ + m^-$. In this formulation, each vertex in the graph represents an entity in the network, or a variable in the dynamical system, and each edge between nodes A and B represents

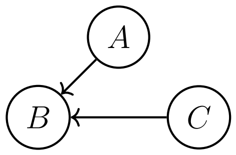
the interaction between A and B if it exists, and the sign of the edge represents the nature of the interaction. In other words, referring to section 1.2, the nodes of a network would be represented by vertices in a graph, and the links of a network would be represented by its edges.
All the examples of networks in section 1.3 can be represented as graphs, where the correlations between the entities of the network are represented by edges. However, we will focus on social networks because we will exploit generalizable properties of this type of networks.

2.2.1 *Triads*

We define a triad as a set of three vertices such that each of the nodes has at least one directed edge between them [30]. To illustrate the definition, consider three nodes A, B and C. For example, subfigure 2.5a is an example of a triad while subfigure



(a) An example of a valid triad because each two of the three nodes are connected by at least one edge



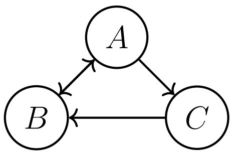
(b) An example of an incomplete triad because A and C are not connected

Figure 2.5: Examples of one valid and one invalid triad in directed graphs

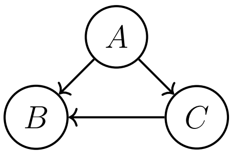
2.5b is not an example of a triad because it is missing a connection between A and C.

2.2.2 *Semicycles*

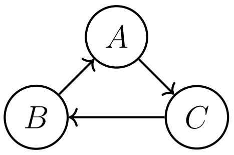
Given a triad, if there are 3 edges incident on its nodes such that for every pair of nodes, there is one edge, then those three edges form a semicycle [30]. Figure 2.6 demonstrates a triad that has 2 semicycles, and breaks it down to each of its semicycles.



(a) A complete triad with two semi-cycles



(b) First semi-cycle of the triad in 2.6a



(c) Second semi-cycle of the triad in 2.6a

Figure 2.6: An example of a triad broken down to its semicycles

2.2.3 *Transitive Semicycles*

Define a binary relation $\mathscr{R}$ such that for 2 nodes A and B (third in the triad is C), $A\mathscr{R}B \leftrightarrow (A, B) \in E$ [30]. A semicycle is said to be transitive if the relation is transitive over the set of the semicycle's edges. That is,

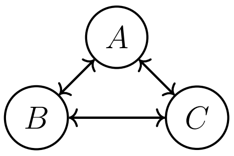
$$A\mathscr{R}B \& B\mathscr{R}C \to A\mathscr{R}C \tag{2.1}$$

This relation would translate to given a vertex in a semicycle, if we follow the directed edges starting from this vertex, we would not reach to the same vertex again. Figure 2.7 represents a triad with 8 semicycles. After breaking it down into its semicycles, we see that only 6 out of the 8 semicyles are transitive, while the other two are intransitive because they do not satisfy the relation in 2.1. For instance, if we take the semicycle in subfigure 2.7b, if we start at vertex A, the outgoing edge leads us to B, then the outgoing edge leads us to C, and from C, the edge leads us to A again, which is the starting edge.

2.3 Representations of Graphs

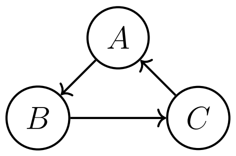
In this section, we explore various representations of graphs that can make the storage of those graphs suitable for computational purposes. For demonstration purposes, consider the graph in figure 2.8, which we will use to represent in each of the representations we present in this section. In this sample graph, we have:

- Vertex Set $V = \{A, B, C, D, E\}$, and $n = 5$
- Positive Edge Set $E^+ = \{AB, BC, CB, CD, DE\}$, and $m^+ = 5$
- Negative Edge Set $E^- = \{AE, EC, BD\}$, and $m^- = 3$
- Edge set has size $m = m^+ + m^- = 8$

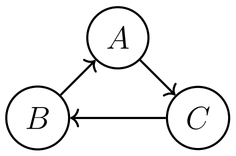


(a) A triad with eight semicycles

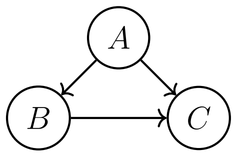
Break down of the triad into its semicycles



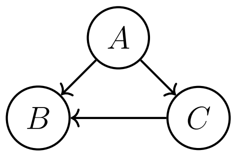
(b) Not a transitive semicycle



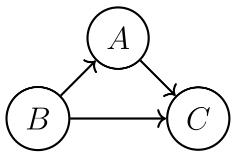
(c) Not a transitive semicycle



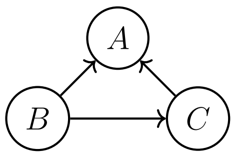
(d) A transitive semicycle



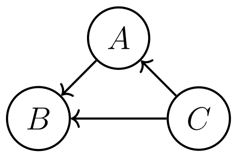
(e) A transitive semicycle



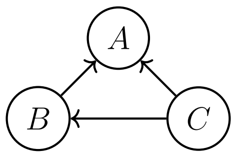
(f) A transitive semicycle



(g) A transitive semicycle



(h) A transitive semicycle



(i) A transitive semicycle

Figure 2.7: Detailed breakdown of a triad into its constituent semi-cycles, and classifying them as transitive or intransitive

2.3.1 *Matrix Representation*

A graph can be represented using an adjacency matrix $\mathbf{A}$, such that $\mathbf{A}$ is an $n \times n$ matrix where each row corresponds to a node i and each column corresponds to a node j. The element of the matrix that falls on entry $\mathbf{A}_{ij}$ represents the edge (i, j) connecting node i to node j [16]. There are three possible values for $\mathbf{A}_{ij}$.

$$\mathbf{A}_{ij} = \begin{cases} 0, & \text{if } (i,j) \notin E \\ +1, & \text{if } (i,j) \in E^{+} \\ -1, & \text{if } (i,j) \in E^{-} \end{cases} \tag{2.2}$$

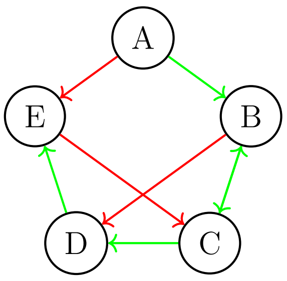


Figure 2.8: A sample directed graph to be represented in multiple representations. Green edges are positive while red edges are negative.

Those values apply to unweighted graphs only. In the case of weighted graphs, $\mathbf{A}_{ij}$ can take any value in $\mathbb{R}$ that represents that sign and weight of the given edge. The graph in figure 2.8 graph would have the following adjacency matrix $\mathbf{A}$

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix} \end{array} \tag{2.3}$$

2.3.2 *Coordinate Format (COO)*

The coordinate format (COO) format is widely used for representing sparse matrices because of its simplicity and direct approach. In COO, we store the adjacency matrix of a graph in three arrays, which represent the non-zero elements of the matrix:

- **Row Indices:** This array stores the indices corresponding to the rows of each non-zero element in the matrix. For graphs, these indices represent the nodes that act as the source of an edge.
- **Column Indices:** Similar to Row Indices, this array stores the indices corresponding to the columns of each non-zero element. For graphs, these represent the destination nodes for each edge.
- **Values:** This array stores the actual non-zero values found at the corresponding row and column indices. This straightforward representation makes it easy to iterate over non-zero elements, which is advantageous in many numerical and graph algorithms.

The COO format is particularly beneficial for applications where the matrix needs to be built incrementally since it allows the easy addition of non-zero entries without reorganizing the entire data structure. However, it is less efficient for operations that require frequent row or column slicing compared to CSR or CSC formats [33].

The representation of the adjacency matrix given in (2.3) in COO format is shown in table (2.1).

Row Indices	0	0	1	1	2	2	3	4
Column Indices	1	4	2	3	1	3	4	2
Values	1	-1	1	-1	1	1	1	-1

Table 2.1: COO representation of the adjacency matrix given in (2.3)

2.3.3 *Compressed Sparse Row Format (CSR)*

This format is commonly used to store sparse matrices efficiently. In the Compressed Sparse Row (CSR) format, we store the adjacency matrix of a graph using three arrays that capture the structure and values of the nonzero elements:

- **Row Pointers:** An array where the i^{th} entry denotes the index in the Values array where the i^{th} row starts. The size of this array is one more than the number of rows in the matrix, with the last element storing the total number of nonzero elements, providing a quick way to determine the number of elements in any row.

- **Column Indices:** Corresponds to the column indices of the elements in the Values array. For graphs, this would represent the nodes that act as the destination for an edge originating from the node represented by the row.

- **Values:** Stores the nonzero values of the adjacency matrix in the order they appear in the matrix, row-wise.

CSR is particularly useful for matrix-vector multiplications and is better suited than COO for row-oriented operations. This format allows efficient access to rows, which is beneficial for algorithms that primarily require row-wise traversal of the matrix [33].

The representation of the adjacency matrix given in (2.3) in CSR format is presented in table (2.2).

Row Pointers	0	2	4	6	7	8		
Column Indices	1	4	2	3	1	3	4	2
Values	1	-1	1	-1	1	1	1	-1

Table 2.2: CSR representation of the adjacency matrix given in (2.3)

2.3.4 *Compressed Sparse Column Format (CSC)*

This format is commonly used to store sparse matrices efficiently, particularly well-suited for column-wise operations. In the Compressed Sparse Column (CSC) format, we store the adjacency matrix of a graph using three arrays that capture the structure and values of the nonzero elements:

- **Column Pointers:** An array where the i^{th} entry denotes the index in the Values array where the i^{th} column starts. This array is one entry longer than the number of columns, with the last element indicating the total number of nonzero elements, facilitating quick determination of the number of elements in any column.

- **Row Indices:** Corresponds to the row indices for the elements in the Values array. For graphs, this would represent the nodes that are the origin of an edge directed towards the node represented by the column.

- **Values:** Stores the nonzero values of the adjacency matrix as they appear in the matrix, column-wise.

CSC is particularly useful for matrix operations that require efficient access to columns, such as certain types of matrix factorizations and solving systems of linear equations where column pivoting is necessary [33].

The representation of the adjacency matrix given in (2.3) in CSC format is presented in table (2.3).

Column Pointers	0	0	2	4	6	8		
Row Indices	0	2	1	4	1	2	0	3
Values	1	1	1	-1	-1	1	-1	1

Table 2.3: CSC representation of the adjacency matrix given in (2.3)

2.3.5 *Hybrid Formats*

Hybrid COO-CSR Format

The Hybrid COO-CSR format combines the strengths of the coordinate format (COO) and Compressed Sparse Row (CSR) formats to offer more flexibility and efficiency in certain sparse matrix operations. This hybrid format can leverage the benefits of both the simplicity of the COO format and the row-access efficiency of the CSR's row-access format:

- **Row Indices (COO component):** Similar to the pure COO format, the hybrid format maintains an array of row indices that store the positions of non-zero elements row-wise. This component is particularly useful for incremental matrix building and easy iteration over non-zero elements.

- **Column Indices and Values (CSR component):** From the CSR format, the hybrid method adopts the approach of storing column indices and corresponding values together, segmented by rows. This allows for quick access and operations specific to rows of the matrix.

- **Row Pointers (CSR component):** This array is retained from CSR, indicating the start of each row's data in the Column Indices and Values arrays. It enhances the ability to jump quickly to specific rows, thus facilitating efficient row-based processing.

This hybrid approach is beneficial in scenarios where both row-wise and element-wise access are frequently required, combining the quick insertions and flexibility of COO with the structured and efficient row access of CSR. It is particularly effective in iterative algorithms that need to modify matrix structure or values dynamically during computation.

The representation of the adjacency matrix given in (2.3) using a hybrid COO-CSR format can be illustrated as follows (this table would be conceptual since hybrid formatting would depend on implementation specifics):

Row Pointers	0	2	4	6	7	8		
Row Indices	0	0	1	1	2	2	3	4
Column Indices	1	4	2	3	1	3	4	2
Values	1	-1	1	-1	1	1	1	-1

Table 2.4: Conceptual illustration of the hybrid COO-CSR representation of the adjacency matrix given in (2.3)

Hybrid COO-CSC Format

The Hybrid COO-CSC format merges the advantages of the coordinate format (COO) and Compressed Sparse Column (CSC) formats to provide a versatile framework for handling sparse matrices. This format is particularly useful in scenarios where both flexibility in building the matrix and efficiency in column-based operations are required:

- **Column Indices (COO component):** In this hybrid format, we maintain a list of column indices similar to the COO format. This aspect helps in dynamically building the matrix and efficiently iterating over elements based on their column positioning.
- **Row Indices and Values (CSC component):** Drawing from the CSC format, row indices and their corresponding values are stored together, allowing for quick access and manipulation of data organized by columns.
- **Column Pointers (CSC component):** From the CSC, this format retains the column pointers array that indicates the start of each column in the Row Indices and Values arrays. This structure ensures efficient access and operations on columns, facilitating algorithms that require frequent column traversal and manipulation.

This hybrid setup is ideal for applications involving matrix transformations, fast column accesses, and algorithms requiring rapid modifications and queries based on columnar data. It combines the ease of element insertion from COO with the structured access provided by CSC.

The representation of the adjacency matrix given in (2.3) using a hybrid COO-CSC format can be conceptually illustrated as follows (note that the actual table format would depend on specific implementation details):

Column Pointers	0	0	2	4	6	8		
Row Indices	0	2	1	4	1	2	0	3
Column Indices	1	1	2	2	3	3	4	4
Values	1	1	1	-1	-1	1	-1	1

Table 2.5: Conceptual illustration of the hybrid COO-CSC representation of the adjacency matrix given in (2.3)

2.3.6 ***Modeling a Social Network***

In a social network, an individual A is represented by a vertex in $\mathcal{G}$. If individual A knows another individual B, then there exists a directed edge $e_{A\to B}$ that connects A to B. An edge $e_{B\to A}$ would connect B to A in the opposite direction only if B knows A. If B is a friend of A, then $\sigma(e_{A\to B}) = +1$. The sign of this edge $\sigma(e_{B\to A}) = -1$ in case B is an enemy of A.

Chapter 3

Multiscale Measures of Global Balance in Signed Directed Graphs

Motivation

In this chapter, we explore and define measures of balance on multiple scales that we will use to characterize our complex networks. We also introduce the algorithms that are used to compute those measures and comment on possible limitations.

3.1 Definition of Balance on the Microscale

To define balance on the microscale, we set criteria from social theory that deem a semicycle as balanced or not:

- The friend of my friend is my friend
- The enemy of my enemy is my friend

To formulate the above using graph theoretic formulations, consider a signed and directed graph $\mathcal{G} = (V, E, \sigma)$. Assigning a positive sign to an edge that corresponds to friendship, and a negative sign to an edge that corresponds to enmity, we realize that we need the product of the edges in a transitive semicycle to be positive. It is notable that the definition based on social theory explains the intuition behind the need for transitive semicycles. As such, if we have 3 positive edges or if we have two negative edges and one positive edge in a transitive semicycle, then the transitive semicycle is balanced. The transitive semicycle is said to be negative otherwise.
To place the definition above in a mathematical framework, Balance theory suggests that a transitive semicycle $< V_i, V_j, V_k >$ is balanced if

$$\prod_{i,j;i\neq j} \sigma_{ij} > 0$$

On the scale of the whole network, we require all transitive semicycles to be balanced so that the network is globally balanced.
We thus define the microscale index of frustration of a graph $\mathcal{G}$, if T^+ is the number of balanced transitive semicycles and T^- is the number of unbalanced transitive semicycles as

$$T(\mathcal{G}) = \frac{T^+}{T^+ + T^-}$$

We note that the graph would be globally balanced if $T(\mathcal{G}) = 1$, and unbalanced otherwise.
It is worth noting that negative edges are the source of frustration in a graph. For example, if $\forall e \in E, \sigma(e) = +1$, i.e. all the edges of the graph $\mathcal{G}$ are positive, then the product of all edges of all transitive semicycles have a positive sign, which means that all the transitive semicycles are balanced. However, it is only when a graph has negative edges that it would have a tendency for imbalance as it would introduce the possibility of imbalanced transitive semicycles.

3.1.1 *Significance of 3-cycles*

It is known that in a network of any type, it is likely that interactions may occur beyond the dyadic or triadic level. In social networks, we might have interactions between four individuals or even more. As such, the question that arises is, why do we limit our microscale index of frustration to transitive semicycles of length 3 instead of transitive semicycles of any length? In other words, it might be the case that all transitive semicycles of length 3 are balanced, while cycles of length 4 are imbalanced, and thus while the microscale index of frustration indicates a value of 1 as in the network is globally balanced, the index does not reflect reality given that frustration exists in cycles of longer lengths. It is worth noting that literature has recently started exploring balance in a directed graph, and as such the notion of transitive semicycle is not clear for cycles of longer lengths for directed graphs.
To make the discussion more meaningful, we first start by generalizing the definition of balance for cycles of any length. A cycle is said to be balanced if the product of the signs of its edges is positive, and imbalanced otherwise. For a cycle of length k, denote the total number of cycles of length k by O_k. Denote the number of balanced cycles of length k by O_k^+ and the number of imbalanced cycles of length k by O_k^-. This being said, $O_k = O_k^+ + O_k^-$.
The first measure of balance that emerges as a result of this definition [34] is

$$D(\mathcal{G}) = \frac{\sum_{i=3}^{n} O_k^+}{\sum_{i=3}^{n} O_k} \tag{3.1}$$

in which we measure balance as the fraction of balanced cycles of any length to that of the total number of cycles of any length.
Another definition that can arise, is

$$D_k(\mathcal{G}) = \frac{O_k^+}{O_k} \tag{3.2}$$

which is an evaluation of balance based on the fraction of the balanced cycles of length k to the total number of cycles of the same length. A special instance of $D_k(\mathcal{G})$ is

$$D_3(\mathcal{G}) = \frac{O_3^+}{O_3} = \frac{T^+}{T^+ + T^-} = T(\mathcal{G})$$

which is our original definition of the microscale index of frustration.
However, in a complete graph, if every 3-cycle is balanced then the graph is globally balanced. Similarly, if there exists a cycle of any length that is imbalanced in a complete graph, then there exists a 3-cycle that is imbalanced [35], [36]. This results in making it sufficient to study $T(\mathcal{G})$ only when the graph is complete. In the context of our problem, the networks we infer will always complete.

3.2 Definition of Balance on the Macroscale

We are interested in characterizing balance by looking at the network as a whole instead of a group of triads. As such, we seek a measure of how far a graph is from global balance. For this purpose, we define the macroscale index of frustration as the minimum number of edges that we need to remove from the set of edges of a graph to render a graph balanced. Denote by E_D the set of edges of the minimum size that we delete from E. Define $L(\mathcal{G}) = |E_D|$. Then, the macroscale index of frustration is obtained by a normalization [30] of $L(\mathcal{G})$ as

$$F(\mathcal{G}) = 1 - \frac{2L(\mathcal{G})}{m} \tag{3.3}$$

We see that when $L(\mathcal{G}) = 0$, $F(\mathcal{G}) = 1$ which means that the graph is fully balanced. In other words, we do not have to delete any edge from the set of edges E of $\mathcal{G}$ to obtain balance.
Since negative edges are the source of frustration in a graph, we can say that removing all negative edges from a graph would result in a globally balanced graph. In other words, taking E^- as our set of deleted edges is a feasible solution. However, given that we might find a deletion set of a smaller size that achieves global balance, we cannot always take $E_D = E^-$. However, we can set an upper bound to $L(\mathcal{G})$ as the number of negative edges in the graph

$$L(\mathcal{G}) \leq m^- \tag{3.4}$$

Nevertheless, Aref and Wilson argue that the normalization given by equation 3.3 is arbitrary [34] and thus provide other alternatives for this normalization. An alternative we discuss is based on the upper bound of $L(\mathcal{G})$ given in 3.4, in which we compute a macroscale index of frustration by normalizing using this upper bound. That is if we define this macroscale index of frustration to be $X(\mathcal{G})$, then

$$X(\mathcal{G}) = 1 - \frac{L(\mathcal{G})}{m^-} \tag{3.5}$$

3.3 Definition of Balance on the Mesoscale

On the Mesoscale, we seek a partition of the set of nodes V of a graph $\mathcal{G}$ into two sets X and $V \backslash X$ such that if we take two nodes i and j:

- if $i, j \in X$, then $\sigma(e_i) = +1$
- if $i, j \in V \backslash X$, then $\sigma(e_i) = +1$
- if $i \in V, j \in V \backslash X$, then $\sigma(e_i) = -1$
- if $i \in V \backslash X, j \in V$, then $\sigma(e_i) = -1$

In other words, nodes within the same set need to be connected with a positive edge, and nodes across the sets need to be connected with a negative edge. Furthermore, if we have nodes within the same set connected by a negative edge or nodes across the sets connected by a positive edge, those scenarios introduce frustration into the graph. However, we cannot guarantee that this partition always exists. Therefore, we seek a partition that minimizes the number of positive edges across the sets, and the number of negative edges within a set.

3.3.1 *Node Coloring Formulation*

We can formulate the problem as a node coloring problem as suggested by [31]. Let $\mathcal{G}$ be a signed graph, and X be a coloring. The frustration count of $\mathcal{G}$ under X is given by

$$f_{\mathcal{G}}(X) = \sum_{(i,j) \in E} f_{ij}(X)$$

where

$$f_{ij}(X) = \begin{cases} 0, & \text{if } x_i = x_j \text{ and } (i,j) \in E^+ \\ 1, & \text{if } x_i = x_j \text{ and } (i,j) \in E^- \\ 0, & \text{if } x_i \neq x_j \text{ and } (i,j) \in E^- \\ 1, & \text{if } x_i \neq x_j \text{ and } (i,j) \in E^+ \end{cases} \tag{3.6}$$

What this formulation is proposing is finding a coloring that minimizes the frustration count. The formulation assigns a cost of 1 for any undesired edge (i.e. positive across or negative within). Then, the frustration count sums all of those costs, and the coloring would look for a partition that minimizes this frustration count.
In figure 3.1, we see an example of frustrated positive and negative edges. The coloring would be given by whether a vertex belongs to set A or set B. The label of the set would be the color of the vertex. In subfigure 3.1a, we see that a positive edge that connects vertices in the same set is not frustrated (colored in green), while a positive edge that connects vertices across sets would be frustrated. In this case, $f_{AB}(X) = 0$, while $f_{CD}(X) = 1$. Following the same logic for negative edges demonstrated in subfigure 3.1b, we see that the negative edge connecting vertices in the same set is frustrated, and that connecting vertices across the sets is not frustrated. Thus, $f_{AB}(X) = 1$ and $f_{CD}(X) = 0$

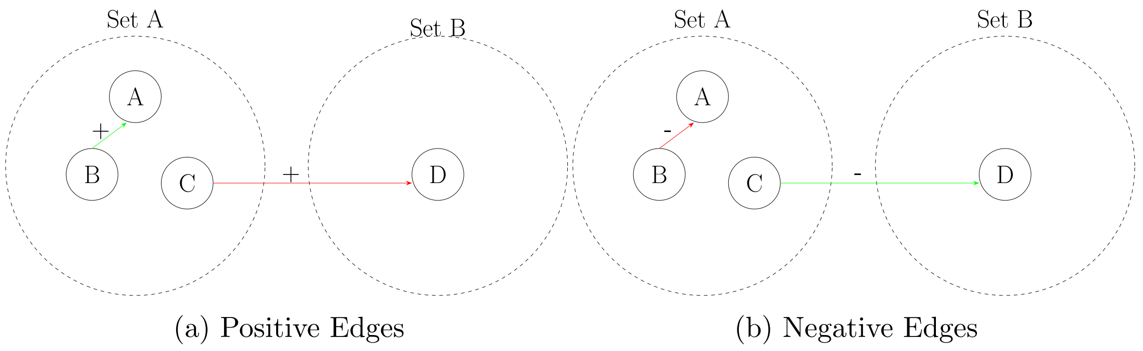


Figure 3.1: An example of frustrated edges in the partition. An edge colored in green is an edge that gets assigned a value of zero, and that in red gets assigned a value of one.

3.3.2 *Cohesiveness and Divisiveness*

We can introduce 2 measures of balance on this scale that evaluate the cohesiveness and divisiveness of a graph.
Cohesiveness is the ratio of the number of positive edges that are within the sets (i.e. do not introduce frustration into the graph) to the total number of positive edges.
Similarly, divisiveness is the ratio of the number of negative edges that are across edges (i.e. do not introduce frustration) to the total number of negative edges.
To formulate the definition better, we define sets of internal and external edges.
For a partition $P = X, V \backslash X$, the set of internal edges is defined as $E_i^P = \{(i,j) \in E | i,j \in X \text{ or } i,j \notin X\}$ that is the set of edges that are not across the sets. The set of external edges is defined as $E_e^P = \{(i,j) \in E | i \in X, j \notin X \text{ or } i \notin X, j \in X\}$.
Thus, we can define the cohesiveness of a partition of a graph as

$$C(P) = \frac{|E_i^P \cap E^+|}{|E^+|}$$

and its divisiveness as

$$D(P) = \frac{|E_e^P \cap E^-|}{|E^-|}$$

We note that $C(P) = 1$ indicates that we do not have any positive edge that connects nodes in different sets of the partition. Similarly, $D(P) = 1$ indicates that we do not have any negative edges connecting nodes within the same set of the partition. If we have $C(P) = D(P) = 1$, then the graph is globally balanced with no frustration.

3.3.3 *Formulating the size of edge deletion in terms of the frustration count*

We seek the partition P that minimizes the frustration count, and we know that the frustration count is a total cost function that penalizes edges that introduce frustration. We can thus write $L(\mathcal{G})$ as

$$L(\mathcal{G}) = \min_{X \subseteq V} f_{\mathcal{G}}(X)$$

since $L(\mathcal{G})$ is the edge deletion set of minimum size, and the coloring problem minimizes the number of edges that are penalized. In other words, if we remove the edges that introduce frustration in the best partition, we would obtain the minimum number of edges that we need to remove to achieve global balance because those are the edges that are impeding us from getting the desired partition of the graph.

3.4 Computing Balance on the Microscale

Based on the definition of the index of frustration on the microscale, we can use a triangle counting algorithm to compute this index. We need two counts to be able to compute this index:

- The number of balanced transitive semicycles in a graph
- The number of total transitive semicycles in a graph

To compute the numbers above, we exploit the fact that a transitive semicycle can be detected by only one of its three edges, which is the edge such that both of its nodes are sources for the two other edges in the transitive semicycle. For instance, in

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[rr] & & C \end{tikzcd}$$

we have edge AB such that A is the source of edge AC and B is the source of edge BC. Neither AC nor BC satisfies this property. As such, by using the edge that satisfies this property in a transitive semicycle, we can detect it and none of the other edges will detect it, so we can avoid multiple counting this way. It follows that we can loop over the list of edges of the graph, and for each identify the key transitive semicycles that it participates in.
We also exploit another fact which is the one that tells us that a transitive semicycle is balanced if it has exactly 0 or 2 negative edges. Similarly, a transitive semicyle is imbalanced if it has exactly 1 or 3 negative edges.

3.4.1 *Serial Algorithm*

The algorithm would loop over the list of (src, dst) pairs in the positive COO array, identify the outgoing neighbors of the src and dst nodes, and find the cardinality of the intersection of the two sets of outgoing neighbors. That is, if the endpoints of an edge are (u, v) in a directed definition, and if $N^+(u)$ are the nodes that are connected to u through an outgoing positive edge, and $N^-(u)$ are those connected to u through an outgoing negative edge, we seek $|N^{\sigma_1}(u) \cap N^{\sigma_2}(v)|$ to get the number

of triangles, where σ_i is some sign.
The following table summarizes how the intersections guide us to determine the numbers of balanced and unbalanced transitive semicycles.

$(u,v) \in E^+$	$N^+(u)$	$N^-(u)$
$N^+(v)$	balanced	unbalanced
$N^-(v)$	unbalanced	balanced

$(u,v) \in E^-$	$N^+(u)$	$N^-(u)$
$N^+(v)$	unbalanced	balanced
$N^-(v)$	balanced	unbalanced

As such, this would enable us to loop over the positive edges, increment the number of balanced transitive semicycles by finding the 2 intersections that correspond to balance, and then loop over the negative edges, and perform the other 2 intersections that would lead to balance. The hybrid COO-CSR matrix is what enables us to find the outgoing neighbors of a given node u.
The algorithm is described in Algorithm 1.

Algorithm 1: Serial triangle counting algorithm to compute microscale index of frustration

```
1  function microscaleIndex (COOCSR+,COOCSR-);
   Input  : A hybrid COO-CSR representation of the positive edges
            A hybrid COO-CSR representation of the negative edges
   Output: $T(\mathcal{G})$ the microscale index of frustration
2  $N_{balanced} \leftarrow 0$
3  $N_{unbalanced} \leftarrow 0$
4  for $(u,v) \in E^+$ do
5      $N_{balanced} \leftarrow N_{balanced} + |N^+(u) \cap N^+(v)| + |N^-(u) \cap N^-(v)|$
6      $N_{unbalanced} \leftarrow N_{unbalanced} + |N^+(u) \cap N^-(v)| + |N^-(u) \cap N^+(v)|$
7  end
8  for $(u,v) \in E^-$ do
9      $N_{balanced} \leftarrow N_{balanced} + |N^+(u) \cap N^-(v)| + |N^-(u) \cap N^+(v)|$
10     $N_{unbalanced} \leftarrow N_{unbalanced} + |N^+(u) \cap N^+(v)| + |N^-(u) \cap N^-(v)|$
11 end
12 $T(\mathcal{G}) \leftarrow {}^{N_{balanced}}/_{(N_{balanced} + N_{unbalanced})}$
13 return $T(\mathcal{G})$
```

We notice that there are 4 intersections that need to be computed per iteration, and those intersections may be computed independently.

3.4.2 *Parallel Algorithm*

Exploiting the fact that the four intersections may be computed independently, we propose a parallel algorithm that computes those intersections in parallel. In order to do so, we can assign 4 threads to each edge, and each thread would be responsible to compute one of the intersections that correspond to an edge as demonstrated in figure 3.2.

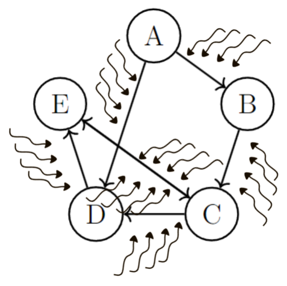


Figure 3.2: Thread assignments to compute the microscale index of frustration on a GPU

3.5 Computing Balance on the Macroscale and Mesoscale

Computing balance on those scales is not an easy problem. The problem is one in combinatorial optimization such that finding the global solution is an NP-hard problem. However, in some formulations, we can compute the macroscale index of frustration, the cohesiveness and the divisiveness for the mesoscale using the same algorithm. We also explore algorithms that only achieve balance on cycles of length 3, while balance might not be achieved for cycles of a longer length.

3.5.1 *Integer Linear Programming*

We need to define the Integer Linear Programming problem (ILP) because the macroscale formulation is mostly formulated as an ILP. Integer Linear Programming is a type of a combinatorial optimization (maximization/minimization) problem that has a linear form and subject to linear constraints. Another condition for ILP problems is to limit the values of the decision variables to integers. ILP are mostly used in fields such as operation research, and are known to NP-hard [37]. The problem may be put in a mathematical form as defined in 3.7.

$$\begin{aligned} &\text{minimize} \quad \mathbf{c}^T\mathbf{x} \\ &\text{subject to} \\ &\mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ &x_i \in \mathbb{Z} \quad \text{for all } i. \end{aligned} \tag{3.7}$$

such that **A** is a matrix of constants, **c** and **b** are vectors of constants, **x** is the vector of integer decision variables.

3.5.2 *The Coloring Problem*

In this section, we explore 3 binary linear programming formulations of equation 3.6 provided by [31]. The reference argues that the three formulations are equivalent, however depending on the graph, one formulation might behave better than the other.

3.5.2.1 The AND Model

The following formulation of the ILP assigns a variable for each edge and node in the graph. All of those variables are binary, and they either indicate the color of the node or whether an edge is penalized or not. There are two colors for the nodes, and each color indicates whether a node should belong to the first or second set of the partition. Based on the partition, the statement checks whether the two nodes of an edge belong to a frustrated state or not. A frustrated state occurs when the endpoints (nodes) of a positive edge belong to different sets of the partition, or when those of a negative edge belong to the same set of the partition. The value of the objective indicates the number of edges that are in a frustrated state. Given that the partition is an optimal partition, it means that in the best case, we need to remove those edges so that we obtain a globally balanced graph. In other words, the value of the objective corresponds to the size of the set of deleted edges that has a minimum size.
By starting with the objective

$$f_{ij} = \begin{cases} x_i + x_j - 2x_ix_j & \forall (i,j) \in E^+ \\ 1 - (x_i + x_j - 2x_ix_j) & \forall (i,j) \in E^- \end{cases} \tag{3.8}$$

We notice that the objective is not linear in this case because of the x_ix_j term. However, we notice that we can introduce a new binary variable instead of this term such that $x_{ij} = x_ix_j$, such that

$$x_ix_j = \begin{cases} 1 & \text{if } \mathrm{AND}_{x_i,x_j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

which explains the name of the model as the AND Model. So this means that the objective would evaluate to 0 when (i,j) belong to the same set of the partition if they are connected with a positive edge or when (i,j) belong to different sets and are connected with a negative edge. The objective would evaluate to 1 for each edge otherwise indicating frustration. We add constraints that indicate those configurations.

$$\begin{aligned} \min_{x_i : i \in V, x_{ij} : (i,j) \in E} \quad & Z = \sum_{(i,j)\in E^+} x_i + x_j - 2x_{ij} + \sum_{(i,j)\in E^-} 1 - (x_i + x_j - 2x_{ij)} \\ \text{s.t.} \quad & x_{ij} \le x_i \quad \forall (i,j) \in E^+, \\ & x_{ij} \le x_j \quad \forall (i,j) \in E^+, \\ & x_{ij} \ge x_i + x_j - 1 \quad \forall (i,j) \in E^-, \\ & x_i \in \{0,1\} \quad \forall i \in V, \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in E \end{aligned} \tag{3.10}$$

We see that the model has $n + m$ variables and $2m^+ + m^-$ constraints.

3.5.2.2 The XOR Model

The following model is formulated by noticing that a frustrated edge is a positive edge (i,j) such that $\text{XOR}_{(x_i,x_j)} = 1$ or a negative edge such that $1 - \text{XOR}_{(x_i,x_j)} = 1$.

$$\begin{aligned}
\min_{x_i : i \in V, f_{ij} : (i,j) \in E} \quad & Z = \sum_{(i,j)\in E} f_{ij} \\
\text{s.t.} \quad & \\
& f_{ij} \geq x_i - x_j \quad \forall (i,j) \in E^+, \\
& f_{ij} \geq x_j - x_i \quad \forall (i,j) \in E^+, \\
& f_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E^-, \\
& f_{ij} \geq 1 - x_i - x_j \quad \forall (i,j) \in E^-, \\
& x_i \in \{0,1\} \quad \forall i \in V, \\
& f_{ij} \in \{0,1\} \quad \forall (i,j) \in E
\end{aligned} \tag{3.11}$$

Thus, we notice that the XOR model has $n+m$ variables and $2m$ constraints. This model captures more elements of the original statement defined in equation 3.6.

3.5.2.3 The ABS Model

We notice that a positive frustrated edge satisfies $|x_i - x_j| = 1$ and a negative frustrated edge satisfies $|x_i + x_j - 1| = 1$. To make use of those properties, we need to linearize the absolute value. As such, we assign two binary variables to each edge e_{ij} and h_{ij}.

$$\begin{aligned}
\min_{x_i : i \in V, e_{ij}, h_{ij} : (i,j) \in E} \quad & Z = \sum_{(i,j)\in E} e_{ij} + h_{ij} \\
\text{s.t.} \quad & \\
& x_i - x_j = e_{ij} - h_{ij} \quad \forall (i,j) \in E^+, \\
& x_i + x_j - 1 = e_{ij} - h_{ij} \quad \forall (i,j) \in E^-, \\
& x_i \in \{0,1\} \quad \forall i \in V, \\
& e_{ij} \in \{0,1\} \quad \forall (i,j) \in E, \\
& h_{ij} \in \{0,1\} \quad \forall (i,j) \in E
\end{aligned} \tag{3.12}$$

We note that for a frustrated edge, either e_{ij} or h_{ij}, and not both, has to be 1. If both take a value of 0, then the edge is not frustrated. In this model, we have $n+2m$ variables and $m = m^+ + m^-$ constraints.
Each of the formulations above are equivalent. Each attempts to find an optimal partition that minimizes the frustrated edges and colors the nodes of the vertices to indicate the set of the partition to which they belong.

3.5.3 ***The 3-Hitting Set Problem***

The models defined in the previous section allow us to evaluate global balance on any graph, whether complete or incomplete. Yet, the graphs that we handle in our

work are complete graphs. We make use of the completeness by making use of the property that the source of frustration in a complete graph is the 3-cycles. The reason behind this is because if the frustration comes from a cycle of a longer length, we can trace down the origin of the frustration to a frustrated transitive semi-cycle. We consider variations of an algorithm that would only balance transitive semicycles of length 3, which we consider to be the source of frustration on the microscale. In this subsection, we provide some formulations that we considered exploring, however there was no need to use them.
To define the hitting set problem, consider a collection $\mathcal{S} = S_1, S_2, \ldots, S_m$ of subsets of a finite set U, find the smallest possible subset H of U such that H contains at least one element from each subset in $\mathcal{S}$. In other words, find a set H of a minimum size such that $H \cap S_i \neq \emptyset$ for all $i \in 1, 2, \ldots, m$ [38].
The d-hitting set problem refers to the instance of the problem in which all the subsets S_i in $\mathcal{S}$ in the collection have a cardinality of d, i.e. $|S_i| = d \quad \forall i \in 1, 2, \ldots, m$. This definition makes it clear that the 3-hitting set problem refers to the problem in which all the subsets of the collection have size 3.
What makes the 3-hitting set problem relevant to our problem, is that we aim to explore algorithms that would remove the edges that are the source of frustration in triads.
Indeed, we want to minimize the number of edges to be removed to achieve this purpose. As such, $\mathcal{S}$ would be the collection of subsets of the set of edges E, such that each of the subsets S_i represents a triplet of edges that constitute a frustrated transitive semicycle.
For example, referring to the benchmark graph, a subset of $\mathcal{S}$ would be the edge triplet $\{e_{13}, e_1, e_5\}$ because those edges constitute an imbalanced transitive semicycle. Therefore, by deleting at least one edge from this subset, we would destroy an imbalanced transitive semicycle and thus have 1 less imbalanced transitive semicycle in the graph. Extending this picture to all the subsets of $\mathcal{S}$, by removing at least one edge from each of the subsets, we would destroy all imbalanced transitive semicycles and thus achieve global balance for all triads. It happens that we can minimize the size of the set of edges $H \subset E$ that we delete in the graph to achieve balance by removing the least number of edges from the graph.

3.5.3.1 Greedy Algorithm

A greedy algorithm is a type of algorithmic paradigm that makes locally optimal choices at each step in the hope of finding a global optimum solution. In other words, at each step of the algorithm, the choice that appears to be the best is made without considering the possible consequences of that choice in the future steps. The basic idea of a greedy algorithm is to repeatedly make the locally optimal choice at each step, which will eventually lead to the global optimal solution. However, this is not always the case, and there are instances where a greedy algorithm fails to find the optimal solution.
The disadvantage of using a greedy algorithm is that the solution it gives is not guaranteed to be the optimal solution, and in most cases it is not. However, an

advantage of using a greedy algorithm is its relatively fast speed of execution, compared to exact algorithms. As such, using a greedy algorithm in our case can give us an upper bound of the exact solution.
The description of the algorithm is given by Algorithm 2.

Algorithm 2: Greedy Algorithm for solving the 3-hitting set problem

```
1  function greedySet (COOCSR+,COOCSR-,COOCSC+,COOCSC-);
   Input  : A hybrid COO-CSR representation of the positive edges
            A hybrid COO-CSR representation of the negative edges
            A hybrid COO-CSC representation of the positive edges
            A hybrid COO-CSC representation of the negative edges
   Output: Size of greedy edge deletion set $|E_G|$
            Greedy solution for edge deletion $E_G$
2  Find the number $N_i$ of frustrated triangles that each edge $e_i \in E$
     participates in
3  while $\sum_{i=1}^{m} N_i \neq 0$ do
4      $e^* \leftarrow e_i \quad s.t. \quad N_i = max_i\{N_i\}$
5      $E_G \leftarrow E_G \quad \cup \quad \{e^*\}$
6      Mark  $e^*$ as deleted
7      Update  $N_i$ accordingly
8  end
9  return $|E_G|, E_G$
10
```

The greedy algorithm starts by counting the number of frustrated transitive semicycles that each edge participates in. Then, the algorithm would identify the edge that participates in the highest number of frustrated transitive semicycles, and add this edge to the solution. After deleting this edge, we find the edges that share a frustrated triangle with the deleted edge, and reduce the corresponding count of the frustrated transitive semicycles that the edge participates in by 1. The algorithm then repeatedly adds the edge that participates in the constantly updating highest number of frustrated transitive semicycles to the solution. The algorithm stops when none of the edges participate in a frustrated triangle, and thus no more frustrated transitive semicycles exist in the graph. In other words, we would obtain a microscale index of frustration of 1 to indicate that all transitive semicycles are balanced. The algorithm returns the indices of the edges that have been added to the solution, and the size of the returned set would be the greedy size of the edge deletion set.

3.5.3.2 Branching Algorithm

We are interested in finding a hitting set that has a minimum set, since the definition of the macroscale index of frustration involves the size of the minimum number of edges that need to be removed and not just any number of edges. As such, we explore a branching algorithm that recursively explores the combinations of edges that might result in a hitting set of a minimum size. It is worth noting that the solution of the problem does not have to be unique. That is, we might find several distinct combinations of edges such that the set has a minimum size, yet any of those solutions is acceptable. The branching algorithm is described in Algorithm 3, and it calls the recursive hitting set function defined in Algorithm 4.

Algorithm 3: Hitting Set Branching Algorithm for solving the 3-hitting set problem

```
1 function branchingHSet (COOCSR+,COOCSR-,COOCSC+,COOCSC-);
  Input  : A hybrid COO-CSR representation of the positive edges
           A hybrid COO-CSR representation of the negative edges
           A hybrid COO-CSC representation of the positive edges
           A hybrid COO-CSC representation of the negative edges
  Output: Size of minimum edge deletion set $|H|$
           An exact solution for edge deletion $H$
2 $|E_G| \leftarrow$ greedySet(COOCSR+,COOCSR-,COOCSC+,COOCSC-)
3 $U \leftarrow edgeSets(COOCSR+, COOCSR-)$
4 hittingSet($U$, { }, $|E_G|$)
5 return $|H|, H$
```

We use the solution obtained from the Greedy algorithm as an upper bound. Whenever a branch exceeds the solutions of the Greedy algorithm, we can directly decide that this branch will not return an optimal solution and thus break it. We also find a set U that contains subsets of the edges that participate in frustrated transitive semicycles. That is, the elements of U are sets of size 3, and the elements of these sets are the indices of 3 edges that determine a frustrated transitive semicycle.
The hittingSet function is a recursive function in which the problem is solved, and its scheme is described in

Algorithm 4: Hitting Set recursive function

```
Input: a universe $U$, a list of sets $H$, and a greedy parameter $greedy$
Output: a hitting set

Global: a list of solutions $solutions$
Global: a current minimum $current\_min$
$H$.sort(); if $len(H) == 0$ then
    $ind\_max \leftarrow$ dictMax($U$); $S \leftarrow U[ind\_max]$; for $s$ in $S$ do
        hittingSet($U$, $\{s\}$);
    end
end
else
    if $len(H) \leq greedy$ and $len(H) < current_min$ then
        for $S$ in $U$ do
            if not intersect($S$, $H$) then
                for $s$ in $S$ do
                    hittingSet($U$, $H + \{s\}$);
                end
            end
        end
        if superIntersect($U$, $H$) then
            if $len(H) < current\_min$ then
                $current\_min \leftarrow$ len($H$); append $H$ to $solutions$;
            end
        end
    end
end
```

The first step in the branching algorithm is to count the number of times that each edge appears in the subsets of $\mathcal{S}$. We then add the obtained number of times that each edge in a subset appears in all the subsets for each of the subsets. We choose the subset that has the highest sum of frequencies of its edges appearing in all subsets, and we branch on its edges. We branch as long as the size of the current solution is less than the size of the greedy solution or the size of any smaller solution that the algorithm has hit.
We demonstrate the branching on the following example. Let $\mathcal{S} = \{1, 2, 3, 4, 5, 6, 7\}$ assuming that $\mathcal{S}$ is the set of edges of some graph. Let U be a collection of subsets such that each subset contains three edges that determine a frustrated transitive semicycle, and take $U = \{\{1, 2, 3\}, \{1, 4, 6\}, \{2, 3, 6\}, \{3, 5, 6\}\}$. We notice that the frequency at which each element appears is as in table 3.2.

Table 3.1: The frequency at which each edge appears in the subsets of the collection

edge	**frequency**
1	2
2	2
3	3
4	1
5	1
6	3
7	0

Now, associate with each of the subsets the sum of frequency of its elements as in

Table 3.2: The total frequency at which the edges in a subset appear

subset	**total frequency**
{1,2,3}	7
{1,4,6}	6
{2,3,6}	8
{3,5,6}	7

We notice that the set $\{2, 3, 6\}$ has the highest total frequency of edges. Thus, we use it as our initial branching set.
The diagram demonstrates the branching in each direction. We notice that the size of the minimum set is 2 because if we branch further, we would find a larger solution on the unfinished branches. The diagram also illustrates how the solution does not have to be unique.

Chapter 4

Non-linear Dynamics and Compartmental Models

Motivation

At this point, we have defined complex systems, their relevance, mathematical and algorithmic frameworks for their representation, and a method for characterizing their balance. In this chapter, we introduce non-linear dynamics and emphasize compartmental models.

4.1 Dynamical Systems

A dynamical system is a system that has quantities, known as the dynamical variables, that are variable with time according to predefined rules. While those rules can have stochastic components, we will focus on deterministic dynamic systems [39]. To put a dynamical system in mathematical terms, consider a vector of n dynamical variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, and a vector field $f(\cdot)$ that maps $f : \mathbb{R}^n \to \mathbb{R}^n$. A continuous dynamical system is defined as a system of n coupled differential equations and has the following mathematical form [40]

$$\dot{\mathbf{x}} = f(\mathbf{x}(t)) \tag{4.1}$$

where $\dot{\mathbf{x}}$ represents the time derivative of $\mathbf{x}$.
A discrete dynamical system is defined as n sequences that have the following mathematical form [39]

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) \tag{4.2}$$

where the subscript t represents the time indexing of the sequence.

4.1.1 *Linear Dynamical Systems*

A linear dynamical system is a system of n coupled differential equations that has the following form [41]

$$\begin{cases} \dot{x_1} = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + c_1 \\ \dot{x_2} = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + c_2 \\ \vdots \\ \dot{x_n} = a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n + c_n \end{cases} \tag{4.3}$$

where $x_1, x_2, \cdots, x_n$ are the dynamical variables of the system, c_i and $a_{ij} \in \mathbb{R}$ for $i, j = 1, 2, \cdots, n$ are real constant coefficients, and $\dot{x}_i$ represents the time derivative of x_i.
This system can be written in matrix form in the form of (4.1) as

$$\dot{\mathbf{x}} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{c} \tag{4.4}$$

or the form of (4.2) as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) = \mathbf{A}\mathbf{x}_t + \mathbf{c} \tag{4.5}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

When a dynamical system does not have the form described in 4.4, this dynamical system would be known is a non-linear system.

4.1.2 *Fixed Points*

We are generally interested in the long-term behavior of a dynamical system. As such, we look for variables for which the system is observing a steady state, and thus no change with respect to time.
Translating the statement above to a mathematical statement means solving for the following equation for continuous systems [40]

$$\dot{\mathbf{x}} = 0 \tag{4.6}$$

or, alternatively, for discrete systems

$$\mathbf{x}_{t+1} = \mathbf{x}_t \tag{4.7}$$

To obtain the solution for those equations, we need to solve a system of simultaneous equations which has the form

$$f(\mathbf{x}^*) = 0 \tag{4.8}$$

where $\mathbf{x}^*$ is the solution of the system and thus the set of fixed points of the system.

4.1.3 *The Jacobian and Stability Analysis*

We are interested in determining the significance of this fixed point. When we speak of the significance, we are particularly addressing the question of what happens to a point that is in the vicinity of this fixed point.
To illustrate our goal, let us consider a small perturbation $\tilde{\mathbf{x}}$ and a point

$$\mathbf{x} = \mathbf{x}^* + \tilde{\mathbf{x}}$$

Exploiting the time derivative of $\mathbf{x}$

$$\dot{\mathbf{x}} = \dot{\tilde{\mathbf{x}}}$$

We can linearize around the fixed point $\mathbf{x}^*$ by doing a Taylor expansion [41]

$$\dot{\mathbf{x}} = \dot{\tilde{\mathbf{x}}} = f(\mathbf{x}^* + \tilde{\mathbf{x}}) = f(\mathbf{x}^*) + \mathbb{J}(\mathbf{x}^*) \cdot \tilde{\mathbf{x}} + \cdots \tag{4.9}$$

where $\mathbb{J}$ is the Jacobian matrix defined as

$$\mathbb{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{4.10}$$

where $\frac{\partial f_i}{\partial x_j}$ is the partial derivative of the i-th component of the vector function f with respect to the j-th state variable x_j. The Jacobian matrix is a very important matrix because it captures how the dynamical variables vary with one another, and thus it captures their interactions.
Using the fact that $f(\mathbf{x}^*) = 0$, we can substitute in 4.9 to get

$$\dot{\tilde{\mathbf{x}}} = \mathbb{J}(\mathbf{x}^*) \cdot \tilde{\mathbf{x}} \tag{4.11}$$

Since $\mathbb{J}(\mathbf{x}^*)$ is a constant matrix, we can say

$$\mathbf{A} = \mathbb{J}(\mathbf{x}^*)$$

and this would remind us of (4.4).Thus, we can linearize the system around the fixed point by means of a Jacobian matrix for a small perturbation. This small perturbation means that we are allowed to ignore the non-linear behavior of the system to a certain extent.
To characterize the dynamics of the system, we would need to find the eigenvalues of this Jacobian matrix evaluated at the fixed point $\mathbf{x}^*$ by solving the eigenvalue problem

$$\det(\mathbf{J}(\mathbf{x}^*) - \lambda\mathbf{I}) = 0$$

where λ is the eigenvalue(s) and $\mathbf{I}$ is the identity matrix. For each eigenvalue λ_i, we would have an eigenvector $\mathbf{e}_i$. This set of eigenvalues with its corresponding set of eigenvectors allows us to solve the linearized system by expanding $\tilde{\mathbf{x}}$ on a basis of eigenvector in the following form

$$\tilde{\mathbf{x}}(t) = \sum_{i=1}^{n} d_i \mathbf{e}_i \exp(\lambda_i t) \tag{4.12}$$

The sign of each of the λ_i in the set of eigenvalues determines what happens to the whole exponential term at $t = \infty$. There is no guarantee that $\lambda_i \in \mathbb{R}$, as such we consider the more general case where $\lambda_i \in \mathbb{C}$.
If the real part $Re(\lambda_i) < 0$, then as $t \to \infty$, the exponential term $\exp(\lambda_i t) \to 0$. If $Re(\lambda_i) < 0 \forall i$, then as $t \to 0$, we have $\tilde{\mathbf{x}} \to 0$ and thus

$$\mathbf{x} = \mathbf{x}^* + \cancel{\tilde{\mathbf{x}}} = \mathbf{x}^*$$

Making sense of the mathematics, it means if all the exponentials for the eigenvalue expansion decay (because all eigenvalues are negative), when the system is in a state that is very close to the fixed point, the system will converge to the fixed point. In this case we call $\mathbf{x}^*$ is a stable fixed point. Alternatively, if at least one of the eigenvalues $\lambda_{i'} > 0$, then there is a direction along which the exponential will blow up to ∞, and thus pull the system away from this fixed point. In this case, the fixed point would be an unstable fixed point [39].
The process of linearizing the system around a fixed point and studying the eigenvalues of the resultant Jacobian is called stability analysis because it determines whether a fixed point is stable or unstable.

4.2 Time Series Analysis

A time series denoted $\{x_t\}$ is a sequence of a variables measured at different times. A time series can be discrete as in

$$x_t = \{x_0, x_1, \cdots, x_T\}$$

or continuous

$$x(t) = \{x(t=0), x(t=t_1), \cdots, x(t=T)\}$$

[42].
Time series are widely used for forecasting and modeling of the relationship that might exist between different variables. It so happens that real life data is an example of time series, as the data would be discrete measurements of a certain quantity with time.

4.2.1 *Standard Linear Vector Autoregressive Model (VAR)*

As mentioned, forecasting is of interest in time series because it helps us predict the time series for times for which we do not have observations or measures. One of the methods that are used for forecasting is the Standard Linear Vector Autoregressive Model which is abbreviated as VAR. The concept behind VAR is that the next

observation of a time series is a linear combination of the current and all previous observations of this time series [43].
That is to say, consider a time series $\{\mathbf{x}_t\}$ where $\mathbf{x}$ is an n-dimensional vector of variables. We can write

$$\mathbf{x}_t = \mathbf{A}_1\mathbf{x}_{t-1} + \mathbf{A}_2\mathbf{x}_{t-2} + \cdots + \mathbf{A}_p\mathbf{x}_{t-p} + \mathbf{c} \tag{4.13}$$

where $\mathbf{A}_i$ is a set of $n \times n$ matrices of coefficients, and $\mathbf{c}$ is an n-dimensional vector of constants. Indeed, we can include an irreducible error term ϵ.
The model above, as mentioned in the introduction of the section, is a linear model. Cenci, Sugihara and Saavedra view this linearity as a limitation in the model because if the time series evolves in a non-linear fashion, this model cannot capture its trends accurately [44].

4.2.2 *Sequential Locally Weighted Global Linear Map*

To resolve the limitation of VAR as introduced in the previous section, Cenci, Sugihara and Saavedra proposed a method known as the S-Map, which is short for Sequential Locally Weighted Global Linear Map, which they demonstrate to work better for forecasting time series generated by non-linear systems and they argue that it is a non-linear extension to VAR [44]. The idea behind the S-Map is that the S-map considers the position of the current data point relative to an attractor in state space rather than the temporal proximity of the point [45].
On the mathematical end, the S-map represents an SVD solution to the linear equation which has the form

$$\mathbf{B} = \mathbf{A} \cdot \mathbf{C} \tag{4.14}$$

where, after setting $Y_i = x_i(t_k + 1)$,

$$B_k = w_k x_i(t_k + 1) = w_k Y_i$$

and

$$A_{kj} = w_k X_j(t_k)$$

and the weight is defined as

$$w_k = e^{-\frac{\theta\|x(t_k)-x(t^*)\|}{\bar{d}}} (4.15)$$

In (4.15), knowing that $x(t^*)$ is called the predictee variable, we have

$$\bar{d} = \frac{1}{n}\sum_i \|x(t_i) - x(t^*)\|$$

as the average distance from the target point and θ is a parameter that controls the local weighting [46]–[48].
We notice that (4.14) is equivalent to solving an optimization problem that has the form

$$\hat{c} = \underset{c\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_j w_j(y_j - x_j c)^2 = \min_{c\in\mathbb{R}^d} \frac{1}{n}(Y - Xc)^T W(Y - Xc) \tag{4.16}$$

and its solution is

$$\hat{c} = (\hat{V}\Sigma^{-1}\hat{U}^T)WY$$

in which we are getting **C** row by row [44].
Cenci, Sugihara and Saavedra also provide a regularized version of the solution for which (4.16) is replaced with

$$\hat{c} = \underset{c\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}(Y - Xc)^T W(Y - Xc) + \lambda\|c\|_2^2, \tag{4.17}$$

where λ is a regularization parameter, and the solution becomes

$$\tilde{c} = (\mathbf{X}^T\mathbf{W}\mathbf{X} + \lambda n\mathbf{I})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{Y} \tag{4.18}$$

in which again we are solving for **C** row by row [44], [49].
Putting everything together, we can forecast using an S-map by following the following relation

$$x_i(t+1) = c_0 + \sum_{j=1}^{d} J_{ij}(t-1)x_j(t) \tag{4.19}$$

where J_{ij} is the set of interaction coefficients given by (4.18), and c_0 is a constant.
In concluding this section, the S-Map gives us a way to linearize our time series by computing the coefficients of the interaction matrices.

4.3 Compartmental Models

This section is dedicated to introducing compartmental models, which are models that, as their name indicates, associate compartments with each of the dynamical variables of a system and links between those compartments by the means of Markov chains. Then, we introduce a type of compartmental model that are epidemiological in nature, which we have chosen as our focus.

4.3.1 *Definition and Network Representation*

A compartmental model is a method of modeling a system in which the system is partitioned into several compartments from or to which the quantities can flow to one another. That is, for each variable x_i of the system, there would be a compartment associated with it, and we would measure the level of quantity in each of the compartments at different times $x_i(t)$ [50]. What is interesting in compartmental modeling is that we can look at the model from a network perspective and represent it as a directed graph, in which the nodes/vertices are the compartments, and the flows from and to each of the compartments are the edges/links between those compartments [50]. From this lens, we can see how compartmental models serve as a great candidate (although not the only one) for modeling complex systems.
In figure 4.1, we have two compartments A and B, and the direction of the arrows indicates the direction of the flow. If an arrow is incoming to the compartment, we say there is an inflow, and if it is outgoing from the compartment, we say there is

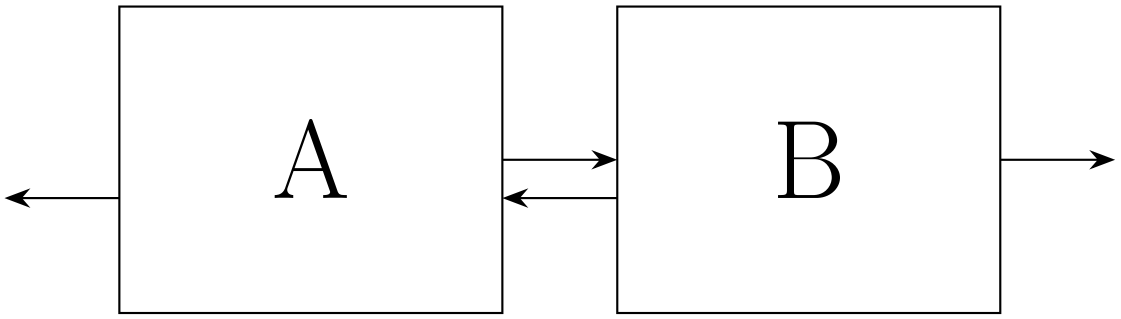


Figure 4.1: An example of an open compartmental model with 2 compartments A and B

an outflow. So here, we have an outflow from A and B to the exterior of the system, and we have an exchange between A and B. This type of system is known as an open compartmental model because $x_A + x_B := g(t)$ which means the total quantity available in all compartments is variable with time. In this example, we can expect $x_A + x_B$ to be decreasing with time.

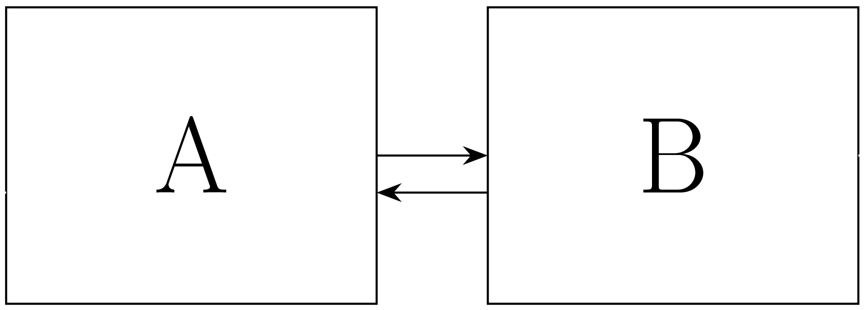


Figure 4.2: An example of a closed compartmental model with 2 compartments A and B

In figure 4.2, we have two compartments A and B, and we see that there is an exchange between the two compartments. However, there is no inflow/outflow from the exterior of the system, which makes it a closed system. In this case, we expect $x_A + x_B = cst$, and summing up the time derivatives of all variables gives zero. Indeed, we can generalize to as many compartments as we want, and we can define the inflows/outflows from any compartment as we please.

4.3.2 *System of Differential Equations*

Now that we have set up the problem, we can define the general form of the system of differential equations. The differential equation that governs the $i-th$ compartment of the compartmental model would have the form [51]

$$\underbrace{\dot{x_i}}_{\text{rate of change of } i} = \underbrace{\sum_k f_{0i} + f_{ki}}_{\text{inflows to } i} - \underbrace{\sum_j f_{i0} + f_{ij}}_{\text{outflows from } i} \tag{4.20}$$

where f_{ij} represents the flow rate from compartment i to compartment j. f_{0i} and f_{i0} describe the rates of flow from and to the external environment for open systems, and we set them to zero for closed systems. There are several ways in which we can define the flow rates in (4.20) in a compartmental model [50] as follows

$$f_{ij} = \begin{cases} c_{ij} & \text{constant flow rate} \\ a_{ij}x_i & \text{donor controlled} \\ b_{ij}x_j & \text{recipient controlled} \\ d_{ij}x_ix_j & \text{donor-recipient controlled or Lotka-Volterra} \\ \frac{\alpha_{ij}x_i}{(\beta_{ij}+x_i)} & \text{chemostat} \end{cases} \tag{4.21}$$

We define the flow rates in our compartmental model depending on the problem. Before taking a particular class of compartmental model, we summarize the steps involved in defining a compartmental model:

- Identify the compartments of the system
- Identify the inflows and outflows of each compartment
- Assign the flow rates for each flow
- Assign an initial quantity for each compartment (initial conditions)

4.3.3 *Epidemiological Models*

We defined compartmental models from a network and a mathematical perspective, and now we are ready to construct a compartmental model. We have chosen epidemiological models due to their extensive connection to the real world. The National Library for Medicine (NLM) defines epidemiology as ”the study of the determinants, occurrence, and distribution of health and disease in a defined population” [52].
We will start by setting up the problem with the most basic SI-model from epidemiology, and we will increase the compartments step by step.

SI Model

In the SI model, we have two compartments: S and I. S stands for Susceptible and I stands for Infected. The SI model considers the disease spread in a population. The flow rate is βSI from S to I. Figure 4.3 demonstrates this system [53]. Applying the flow balance equation in 4.20, we get a system of coupled differential equations

$$\begin{cases} \dot{S} = -\beta SI \\ \dot{I} = \beta SI \end{cases} \tag{4.22}$$

We see that the flow rate is defined as a Lotka-Volterra rate, because the spread of disease depends on how large the number of susceptible and infected individuals is,

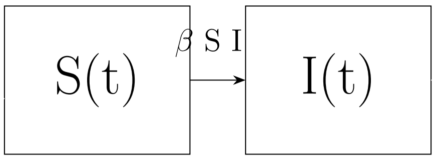


Figure 4.3: The SI model has two compartments, such that the flow goes from compartment S to compartment I with rate βSI.

and thus β is a measure of transmission. An important observation is that summing up $\dot{S} + \dot{I} = 0$. If we denote

$$N(t) = S(t) + I(t)$$

then,

$$\dot{N}(t) = \dot{S} + \dot{I} = 0$$

which means that

$$N(t) = N = cst$$

and thus it is a closed system.
On the long run, the system stabilizes when either $S = 0$ or $I = 0$. Since $I = 0$ is only increasing due to the absence of outflows from $I(t)$, we expect the system to stabilize either when all susceptible individuals get infected, or if there is no infection to spread from the beginning, i.e. $I(0) = 0$.

SIR Model

The SIR Model is an extension of the SI Model, in which we have 1 extra compartment, which is the R compartment. The R stands for recovered. So the journey across the SIR Model is that an individual is susceptible, then they get infected, and then they recover. When they recover, they build up immunity such that they are neither infected nor susceptible. Figure 4.26 represents the SIR system. We realize

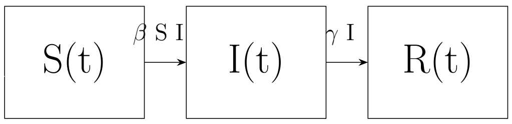


Figure 4.4: The SIR model has two compartments, such that the flow goes from compartment S to compartment I with rate βSI, and from I to R with rate γI.

that the flow rate from I to R is a donor controlled rate, because the number of recovered individuals purely depends on the current number of infected individuals. In this case, γ is a recovery rate. Now, we can write the differential equations

$$\begin{cases} \dot{S} = -\beta SI \\ \dot{I} = \beta SI - \gamma I \\ \dot{R} = \gamma I \end{cases} \tag{4.23}$$

Similar to the SI model, we notice that the SIR model is an example of a closed system, and we notice that $S(t)$ should decrease with time while $R(t)$ should increase with time. As for $I(t)$, the behavior is determined by a competition between β and γ, in which we are looking at the strength of the inflow rate compared to that of the outflow. We can expect that for low levels of R, the inflow rate is higher so I increases, and then at some point the outflow rate is higher so I starts decreasing. As such, the ratio $\alpha = \frac{\gamma}{\beta}$ characterizes I, whereby if $\alpha < 1$, then the inflow rate is higher, $\alpha > 1$, then the outflow rate is higher, and if $\alpha = 0$, this should represent the peak in the $I(t)$ curve.

SEIR Model

When one gets the trend of compartmental models, it becomes easy to generalize the epidemiological models to more compartments to get it closer to reality. The last compartment we will introduce is the exposed compartment, represented by E, and we represent it graphically in figure 4.27. and following the same process, the

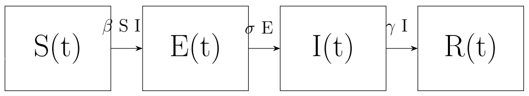


Figure 4.5: The SEIR model has two compartments, such that the flow goes from compartment S to compartment E with rate βSI, from E to I with rate σE, and from I to R with rate γI.

system of differential equations would be

$$\begin{cases} \dot{S} = -\beta SI \\ \dot{E} = \beta SI - \sigma E \\ \dot{I} = \sigma E - \gamma I \\ \dot{R} = \gamma I \end{cases} \tag{4.24}$$

The system in (4.27) holds the same analysis as that defined in (4.26), and it becomes clear when we compute $\dot{E} + \dot{I} = \beta SI - \gamma I$, which is exactly the $\dot{I}$ term in the SIR model.

SIS, SIRS and SEIRS Models

We only explored models so far in which no loop exists. However, all of the models that have been already introduced can be modified to go into a loop. That is, after recovery, the individual can become susceptible again. Those models have an S added to their names to indicate that the journey of the individuals can loop. Figures 4.6, 4.7 and 4.8 represent the new flow balance diagrams. The system of differential equations for the SIS model becomes

$$\begin{cases} \dot{S} = -\beta SI + \gamma I \\ \dot{I} = \beta SI - \gamma I \end{cases} \tag{4.25}$$

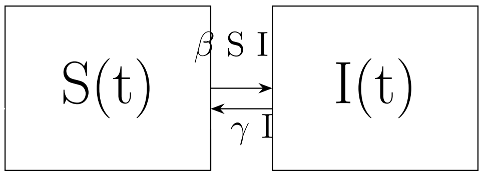


Figure 4.6: The SIS model is very similar to the SI model, and it has an outflow from I to S at a rate γI.

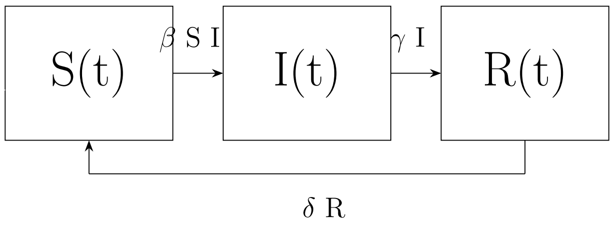


Figure 4.7: The SIRS model is very similar to the SIR model, and it has an outflow from R to S at a rate δR.

The system of differential equations for the SIRS model are

$$\begin{cases} \dot{S} = -\beta SI + \delta R \\ \dot{I} = \beta SI - \gamma I \\ \dot{R} = \gamma I - \delta R \end{cases} \tag{4.26}$$

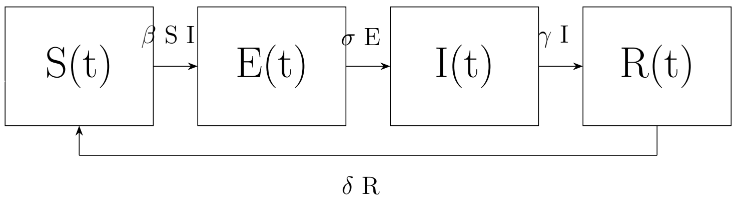


Figure 4.8: The SEIRS model is very similar to the SEIR model, and it has an outflow from R to S at a rate δR.

This translates to the system of differential equations

$$\begin{cases} \dot{S} = -\beta SI + \delta R \\ \dot{E} = \beta SI - \sigma E \\ \dot{I} = \sigma E - \gamma I \\ \dot{R} = \gamma I - \delta R \end{cases} \tag{4.27}$$

Coupled SIR Models

We are interested in a system that has a correspondence with real life scenarios. Consider a set of K populations such that for each population we have compartments for S_k, I_k and R_k that represent the number of susceptible, infected and recovered individuals for the $k-th$ population. Thus, for each population, we would have an SIR system like the one in (4.26). Figure 4.7 demonstrates multiple disconnected SIR models.

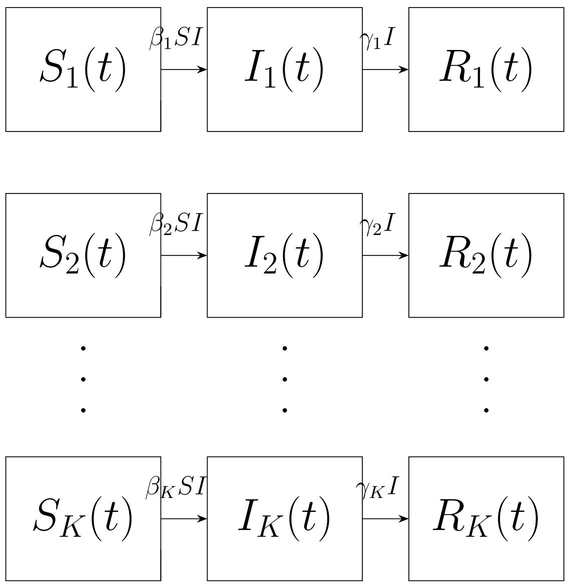

$$\begin{tikzcd}
S_1(t) \arrow[r, "\beta_1 SI"] & I_1(t) \arrow[r, "\gamma_1 I"] & R_1(t) \\
S_2(t) \arrow[r, "\beta_2 SI"] & I_2(t) \arrow[r, "\gamma_2 I"] & R_2(t) \\
\vdots & \vdots & \vdots \\
S_K(t) \arrow[r, "\beta_K SI"] & I_K(t) \arrow[r, "\gamma_K I"] & R_K(t)
\end{tikzcd}$$

Figure 4.9: This figures depicts multiple SIR models for several populations.

The model becomes interesting when we allow individuals from each population to migrate from one to another. That is, on the scale of the population, the system is an open system, but on the global scale of the whole differential equation, the system is closed. This means we are in control of the whole system. Figure 4.10

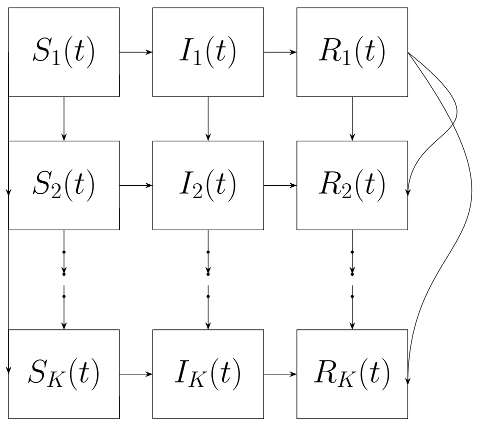

$$\begin{tikzcd}
S_1(t) \arrow[r] \arrow[d] \arrow[dd, bend right] & I_1(t) \arrow[r] \arrow[d] & R_1(t) \arrow[d] \arrow[d, bend left] \arrow[dd, bend left] \\
S_2(t) \arrow[r] \arrow[d, dotted] & I_2(t) \arrow[r] \arrow[d, dotted] & R_2(t) \arrow[d, dotted] \\
S_K(t) \arrow[r] & I_K(t) \arrow[r] & R_K(t)
\end{tikzcd}$$

Figure 4.10: Multiple SIR models for K populations coupled by means of migration effects.

does not reflect all the links to avoid the messiness, however the links must exist between all compartments of the same variable. That is, all S compartments are connected, so are I and R compartments. Thus, the system of differential equations becomes for the $i - th$ population

$$\begin{cases} \dot{S}_i & = -\beta_i \cdot \frac{S_i I_i}{N} + \sum_{j=1}^{K} \left(m_{ji}^S \cdot S_j - m_{ij}^S \cdot S_i \right) \\ \dot{I}_i & = \beta_i \cdot \frac{S_i I_i}{N} - \gamma_i \cdot I_i + \sum_{j=1}^{K} \left(m_{ji}^I \cdot I_j - m_{ij}^I \cdot I_i \right) \\ \dot{R}_i & = \gamma_i \cdot I_i + \sum_{j=1}^{K} \left(m_{ji}^R \cdot R_j - m_{ij}^R \cdot R_i \right) \end{cases} \tag{4.28}$$

where the m_{ij}^V represents the migration rate from i to j for the variable $V = \{S, I, R\}$. The flow rate here is either donor controlled or recipient controlled, depending on the origin. The system would have $3K$ compartments.

Chapter 5

Computational Experiments and Results

Motivation

As we have introduced complex systems, their graph representations, a method to characterize them alongside with their algorithms, in addition to dynamical systems theory with some models, this chapter is dedicated to put the pieces together. We start by doing computational explorations for the serial vs. parallel algorithms. After this, we explore the relation between dynamical stability and the frustration indices. We finally make a conclusion.

5.1 Algorithmic Contribution

5.1.1 *Runtimes of Computation of Microscale Index*

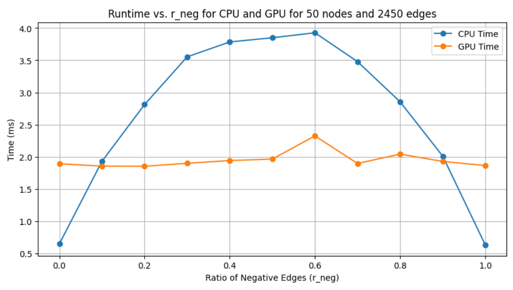
We consider the runtimes for the computation of the microscale index for the serial vs the parallelized schemes. For comparing the runtimes, we consider three parameters:

- The number of nodes in the graph
- The number of edges in the graph
- The ratio of negative to total edges of the graphs

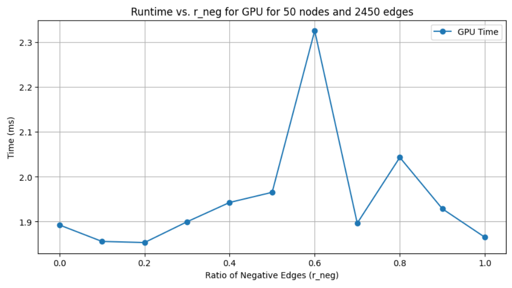
As such, we generate random graphs that satisfy a set of values for the parameters above. For each of the generated graphs, we run the serial and the parallel codes and measure the time it takes for the function to complete its run.

Variable Ratio of Negative to Total Edges

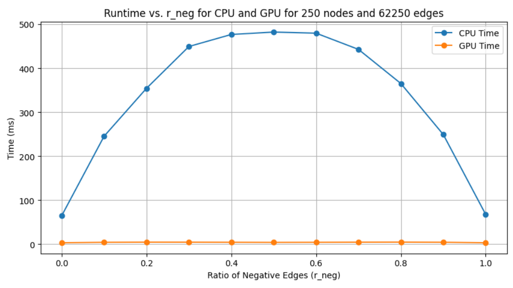
In this experiment, we fix the number of nodes and we generate a complete directed graph that has a variable ratio of negative edges to total edges. We vary this ratio between 0 (fully positive) and 1 (fully negative) with a step of 0.1. The plots in figure 5.1 show the runtime of the functions for computing the microscale index of frustration for 3 complete graphs vs. the ratio of negative to total edges for fixed n = 50, 250, 350 nodes, and the number of edges is that for a complete directed graph. Figure 5.1 shows that the GPU acceleration is significant only when the graph is



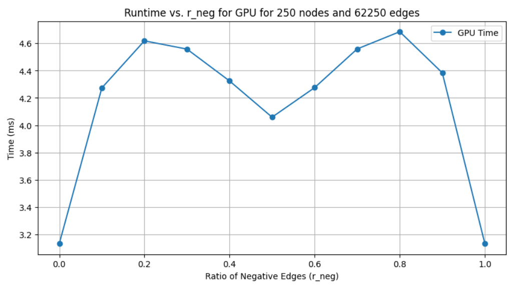
(a) Directed complete graph for 50 nodes



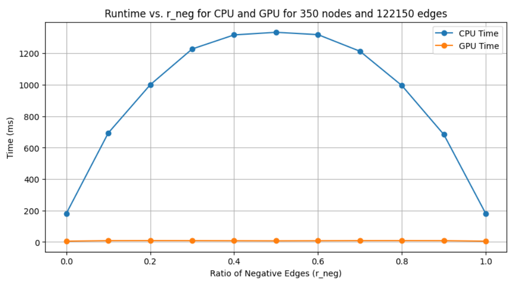
(b) Directed complete graph for 50 nodes for the GPU



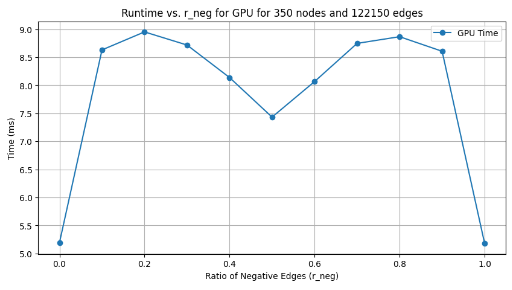
(c) Directed complete graph for 250 nodes



(d) Directed complete graph for 250 nodes for the GPU



(e) Directed complete graph for 350 nodes



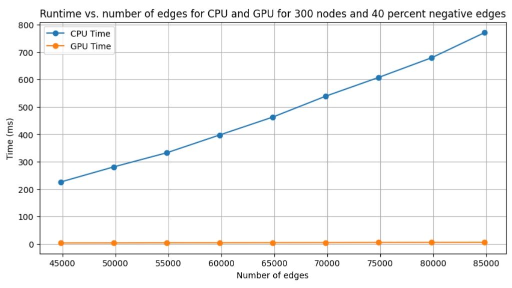
(f) Directed complete graph for 350 nodes for the GPU

Figure 5.1: Comparative analysis of complete directed graphs with 50, 250 and 350 nodes for variable ratio of negative to total edges

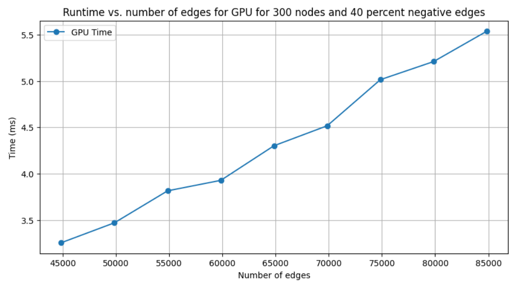
large. We also realize that the GPU runtime grows in variation non-monotonously as a function of the ratio of negative to total edge. The GPU runtimes space out more as the size of the graph increases. As for the comparison with the CPU runtime, we realize that the GPU speedup can reach up to 100 times compared to the CPU runtime.

Variable Number of Edges

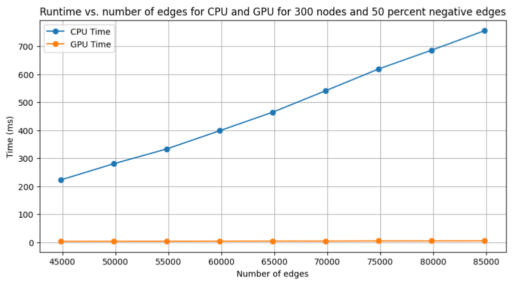
In this experiment, we fix the number of nodes and we generate directed graphs with a variable number of edges for 3 different ratios. In this part, we are not only exploring complete graphs. Figure 5.2 shows that the CPU runtime scales up as the



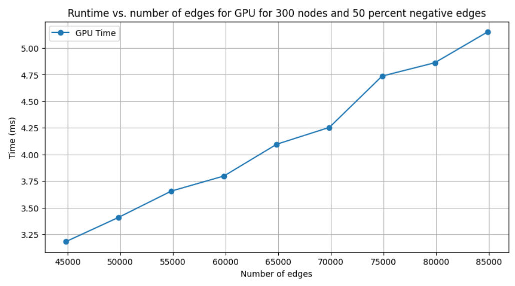
(a) Directed complete graph with ratio = 0.4 negative to total edges



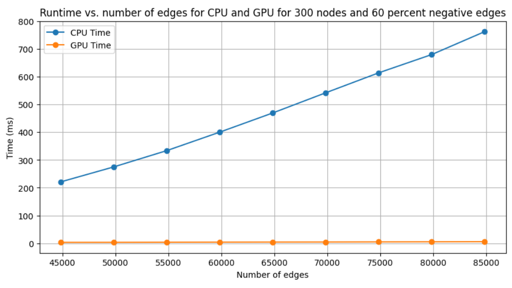
(b) Directed complete graph with ratio = 0.4 negative to total edges for the GPU



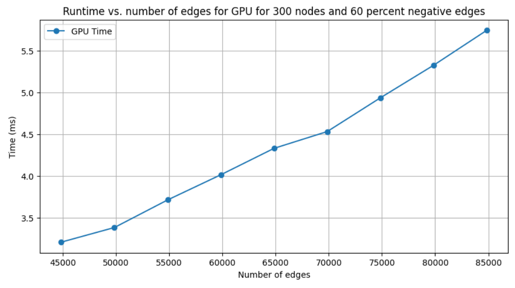
(c) Directed complete graph with ratio = 0.5 negative to total edges



(d) Directed complete graph with ratio = 0.5 negative to total edges for the GPU



(e) Directed complete graph with ratio = 0.6 negative to total edges



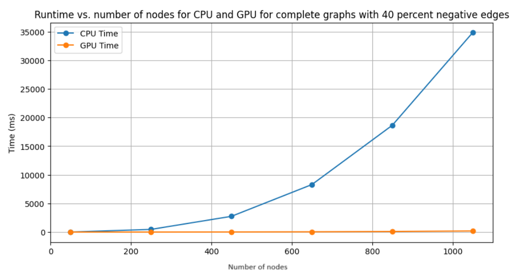
(f) Directed complete graph with ratio = 0.6 negative to total edges for the GPU

Figure 5.2: Comparative analysis of complete directed graphs with 300 nodes and varied number of edges for 40%, 50%, and 60% negative edges

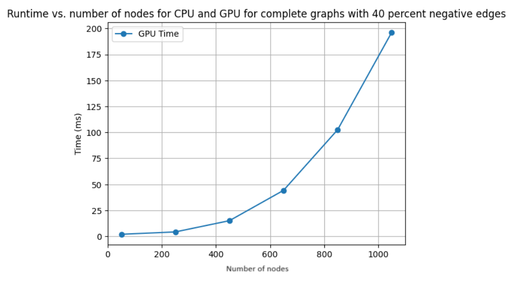
number of edges in the graph grows, and so does the GPU runtime, however the rate of growth of the GPU runtime is slower than that of the CPU. The CPU is always slower in the scenarios that we have considered.

Variable Number of Nodes

In this experiment, we vary the number of nodes and choose the number of edges such that the graph is complete for the given number of nodes. We fix the ratio of negative to total number of edges to be 0.4, because we see in the previous experiments that around this value for the ratio the CPU computation time peaked. Figure 5.3 shows that as the number of nodes increases in a graph, the CPU runtime



(a) Directed complete graph with ratio = 0.4 negative to total edges



(b) Directed complete graph with ratio = 0.4 negative to total edges for the GPU

Figure 5.3: Comparative analysis of complete directed graphs with variable nodes for 40% negative to total number of edges

also increases, at an apparent exponential way. The GPU runtime also increases as the size of the graph increases, but we can see that the rate of increase is way slower than that of a CPU.

5.2 Time Series Analysis Contribution

In this section, we examine closely a dynamical system which has multiple variables and deduce the properties that are encoded in measuring an index of balance.

5.2.1 *Data Generation*

For the purpose of validating that there is a relation between balance and dynamical stability, we explore the compartmental model introduced in (4.10). We start by solving this system numerically for 10 populations, each of which has a compartment for S, I and R, which adds up to a total of 30 compartments. We use ODE solvers provided by the SciPy library [54].
To make the system as close as possible to reality, we generate initial conditions for each of the S and I variables of the system sampled from a uniform random distribution, and for each SIR triplet, $R_0 = N - S_0 - I_0$, where N is the initial size of each of the population, initialized to 100000.
For the flow rates between the compartments of a population, we sample from a normal distribution:

- $\beta_i \sim N(\mu_\beta, \sigma_\beta)$, such that $\mu_\beta = 0.3$ and $\sigma_\beta = 0.05$
- $\gamma_i \sim N(\mu_\gamma, \sigma_\gamma)$ such that $\mu_\gamma = 0.1$ and $\sigma_\gamma = 0.02$

For setting up the migration effects across the population, we define 3 matrices that contain the migration rates from each of the $S_i \to S_j$, $I_i \to I_j$ and $R_i \to R_j$. We

set up those rates asymmetrically such that the migration from $S_i \rightarrow S_j \neq S_j \rightarrow S_i$. We sample those matrices from a uniform random variable that is at most 1% of the population in a compartment.
Figure 5.4 represents the variation of the population sizes for each time instant. We observe that the population size varies with time due to the migration across the compartments.

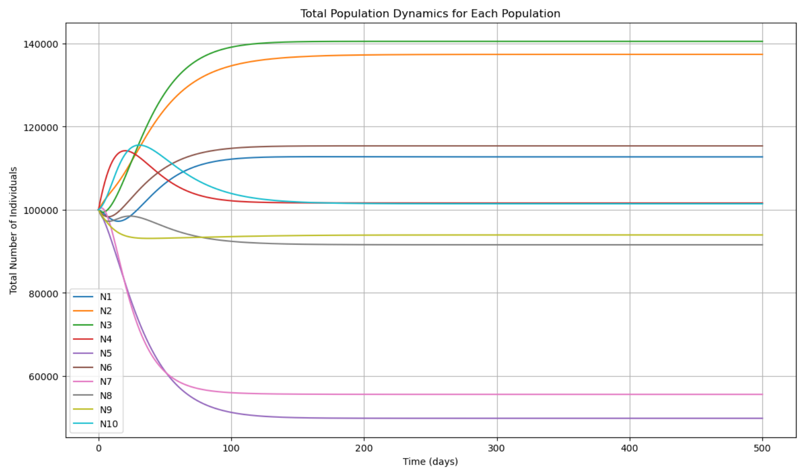


Figure 5.4: A plot of the variation of each population with time

As for the time series, we can get an idea of its behavior, although not with clarity of its behavior with time due to the difficulty of representing 30 variables on 1 graph, in figure (5.5), which looks like multiple SIRs that are not in exact sync if we look closely. Since we expect to handle an index that has a value between 0 and 1, we normalize the time series by dividing by the total population so that we get to visualize the index on the same plot as the time series later. We also round up to 3 decimal places to avoid the numerical fluctuations from the numerical solver from impacting our analysis.

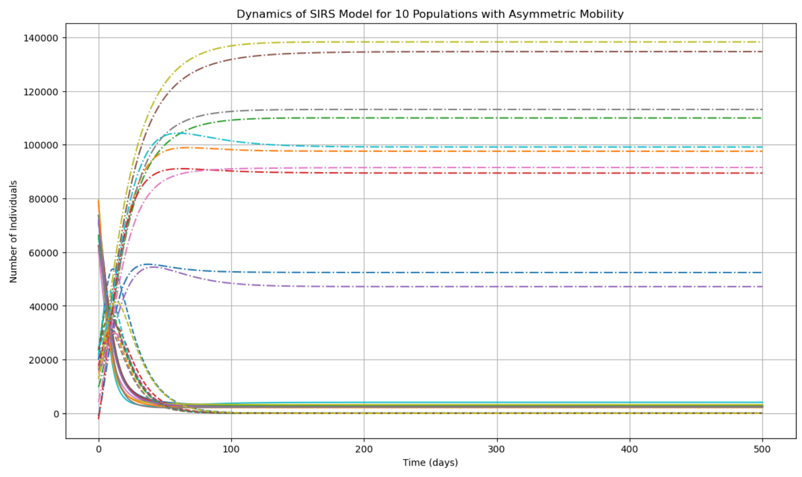


Figure 5.5: A plot of the variation of the time series and the variation of its variables with time

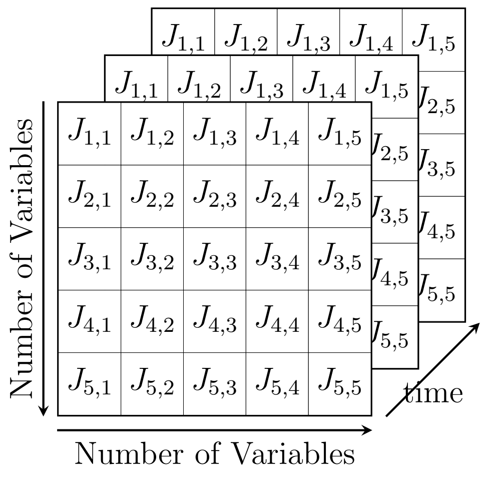

$J_{1,1}$ $J_{1,2}$ $J_{1,3}$ $J_{1,4}$ $J_{1,5}$
$J_{2,1}$ $J_{2,2}$ $J_{2,3}$ $J_{2,4}$ $J_{2,5}$
$J_{3,1}$ $J_{3,2}$ $J_{3,3}$ $J_{3,4}$ $J_{3,5}$
$J_{4,1}$ $J_{4,2}$ $J_{4,3}$ $J_{4,4}$ $J_{4,5}$
$J_{5,1}$ $J_{5,2}$ $J_{5,3}$ $J_{5,4}$ $J_{5,5}$
Number of Variables
Number of Variables
time

Figure 5.6: Stacked interaction matrices at different time instants, each matrix labeled with $J_{i,j}$.

5.2.2 *Inference of Interaction Matrices*

After preprocessing the data, we move to inferring the matrices of the interaction coefficients by utilizing the S-Map method introduced in 4.2.2. This method is available to us through the pyEDM package through the smap function [55].

Figure 5.6 represents a piece of the multiple interaction matrices that we infer for multiple time instants. Given that in our context self-loops do not have any significance, we replace the diagonal entries with zero.

5.2.3 *Frustration Index Computation*

For every set of time instants, we have an interaction matrix that encodes its interactions locally. At this point, we assume that the interaction matrix at a certain instant is the adjacency matrix of the graph that represents the interactions of the system at this instant. As such, we compute the frustration indices for this graph at a given instant. Figure 5.7 shows how the microscale index of frustration varies with time at multiple instants in the graph. It is important to mention that the index of frustration is actually a measure of balance. That is, an index of frustration that has a value of 1 indicates an underlying fully balanced graph. We see that as a trend, the system moves from low levels of balance and increases all the way to total balance.
We are also interested in the macroscale index of frustration which is depicted in figure 5.8, in which we see a similar trend as the microscale index. The interesting point is that the microscale and macroscale indices have a value of 1 at the exact instants, indicating total balance.
We also look at the cohesiveness and divisiveness determined by the mesoscale measure of frustration, through which we can tell how good the partition is in terms of separating nodes connected with negative edges and grouping those connected with positive edges. We see in figure 5.9 that the quality of the partition (which is the

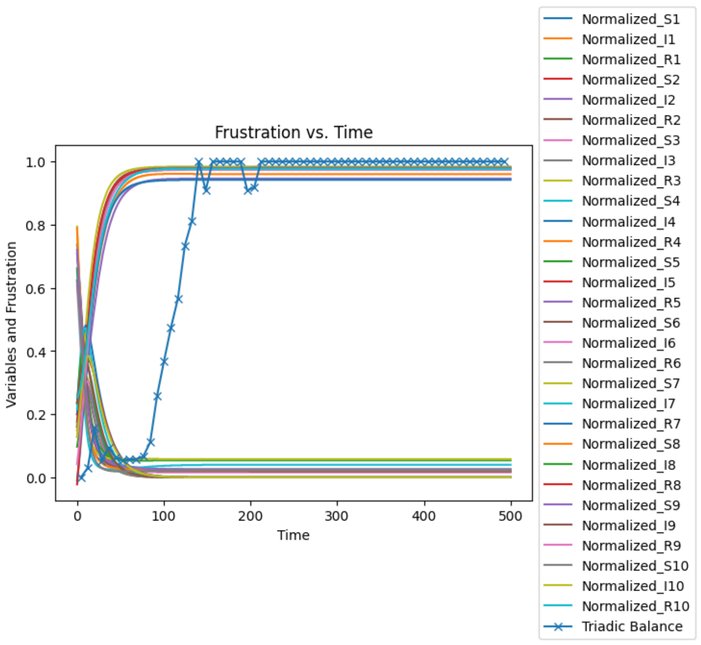


Figure 5.7: The variation of the microscale index of frustration as a function of time

best interpretation of the mesoscale index of frustration) increases with time, meaning that we are having less nodes that are connected with negative edges within the same set, and likewise positive edges across the sets.

5.2.4 ***COVID-19 Real Data***

In this experiment, we explore time series that originate from real data. From the mood of epidemiological modeling, we look at the the COVID-19 dataset available on ourworldindata.org [56]. We take the number of daily confirmed cases, and we take the cumulative sum for those cases for days distributed over 4 years from March 1st, 2020 till February 29th, 2024. We choose 12 countries which have significant COVID-19 cases per population, and among which there is a noticeable mobility. Those countries are Australia, Brazil, China, France, Germany, India, Italy, Japan, Mexico, Russia, Spain and the United States. We normalize by dividing by the maximum number of cases to maintain our values between 0 and 1.

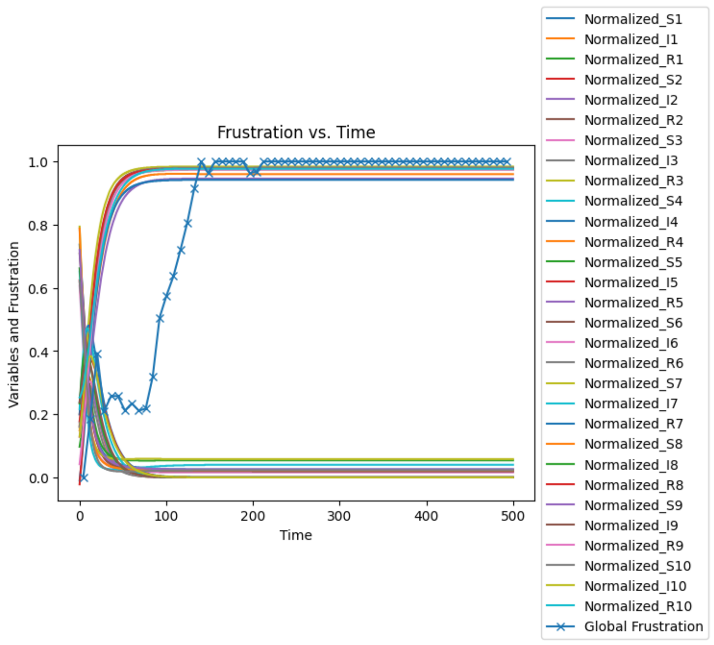


Figure 5.8: The variation of the macroscale index of frustration as a function of time

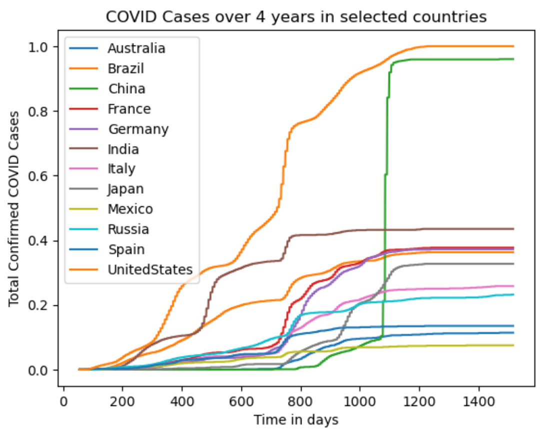


Figure 5.10: The cumulative sum of COVID-19 cases with time for 12 countries with significant interaction and infectivity

Figure 5.10 shows the normalized time series for the countries that have been chosen. The results for evaluating the frustration for the real data is depicted in figure 5.11. We see that for real data, we can still interpret balance using our method, especially from the microscale and macroscale indices of frustration plotted in subfigures 5.11a and 5.11b. We can say that we have achieved a state of balance for the state of the COVID-19 pandemic. We realize that there are some fluctuations in all indices after achieving an index of 1, and there are multiple reasons that can explain this fluctuation. It is possible that with the introduction of new strains and variants of the virus, the equilibrium gets perturbed slightly before recovering. It is also possible that real data has a lot of uncertainties and thus the interaction

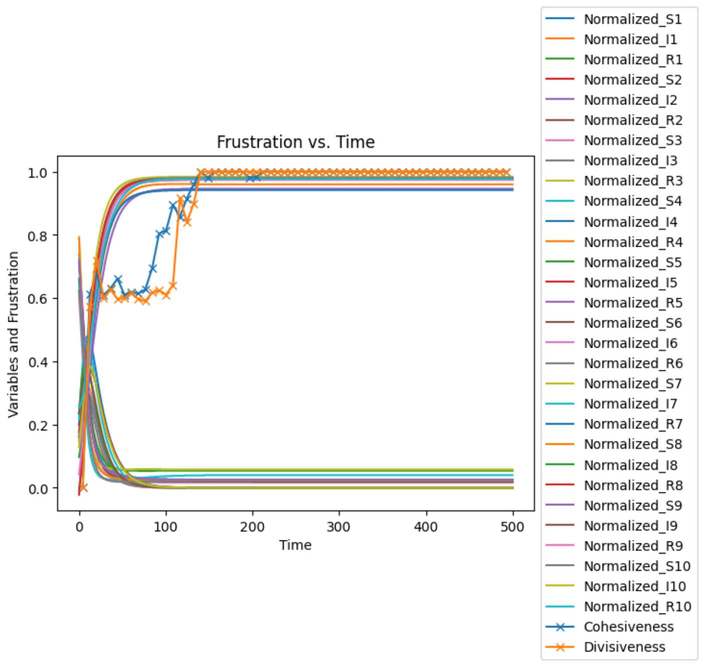


Figure 5.9: The variation of the mesoscale index of frustration represented by the measures of cohesiveness and divisiveness as a function of time

coefficients may be challenged by the smoothness of the data (and the resulting issues with differentiability). Another reason can be that we are not taking a full network of interacting countries, and thus we are missing out on some minor interactions that might be reasonable. Using a full scale network from real data requires large datasets, because generating the SMap requires a minimum number of instants that is related to the number of variables we have in the system, which stems from the constraints we have from solving linear regression problems.
To illustrate the maximum number of data points for which we can compute the index of frustration, we processed the system for a full network of countries and computed the microscale index in figure 5.12.

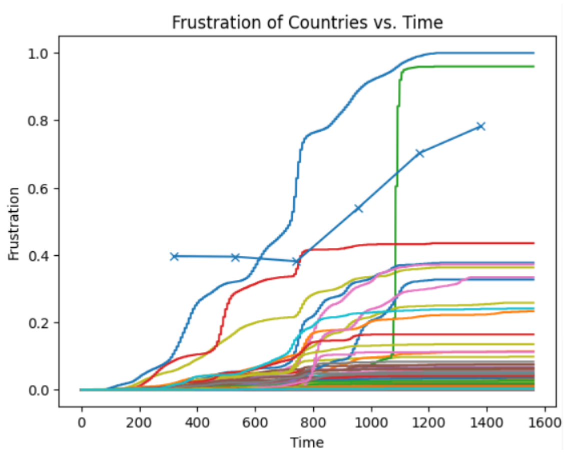
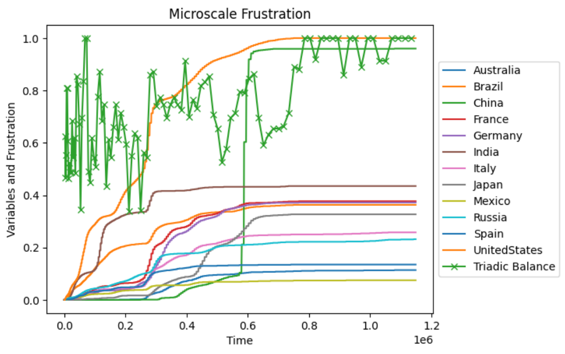
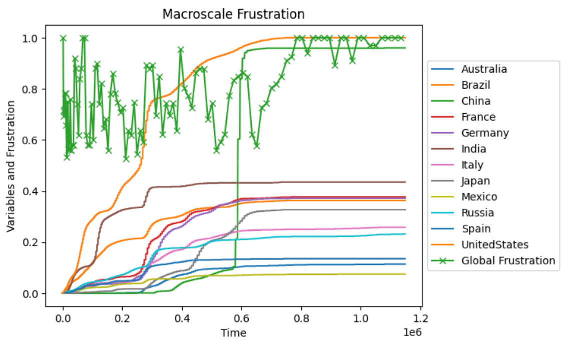


Figure 5.12: The microscale index of frustration for a network of 210 countries

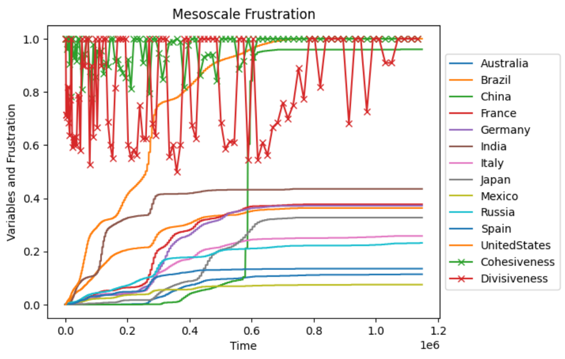
In figure 5.12, we see that despite the scarcity of the points for which we can compute the microscale index of frustration, we can see that as a trend the index is moving towards balance, and we hope that when with more data that gets recorded, to be able to capture this balance.



(a) The microscale index of frustration



(b) The macroscale index of frustration



(c) The mesoscale index of frustration

Figure 5.11: The indices of frustration on multiple scales for real data of COVID-19

Chapter 6

Conclusion and Future Work

Throughout the work, we provided an interesting observation for a field that is massively growing and is becoming present all across the disciplines. We introduced in chapter 1 complex networks with a decent portion of their applications in multiple fields ranging from Physics to Biology to Social Science, International Relations, Politics and Finance. The applications may also extend to multiple other disciplines. We also saw that through network thinking, if one can properly represent a system as a set of nodes having some links as interactions, we can represent the complex system as a network. Moving to chapter 2, we explored graph theory and we used the notion of graphs to represent networks. We explored all the relevant properties of graphs, in addition to storage formats that enable us to represent those graphs in a computation-friendly framework. Then, in chapter 3, we explored multiple algorithms derived from social theory that evaluate the frustration and balance of a social network. The algorithms were graph algorithms at heart, and they ranged from triangle counting to ILP problems and hitting set formulation. We also suspected that the size of the graph that represents our network influences our runtimes considerably, and thus we proposed a parallelization for the algorithm that computes the microscale index of frustration. In chapter 4, we investigated dynamical systems theory, stability and mathematical notions that get us to describe a dynamical system. We also looked into time series and explored the SMap method to infer the matrices of interaction coefficients for the linearized dynamical system that governs a time series. We then introduced compartmental models, and emphasized epidemiological models that are used for modeling the spread of a disease in a population. Finally, in chapter 5, we explored our results on 2 ends.
On the algorithmic end, in which we looked closely at the speedup that has been offered by the parallelization of the algorithm that computes the microscale index of frustration, we observed that the parallel version of our algorithm that computes the microscale index of frustration outperformed the serial version for graphs that have a huge size (whether in terms of edges or nodes). We also realized that our runtime increases when the number of positive edges in a graph, and that of negative edges get close to one another.
The other end was the dynamical systems end for which we took a time series, inferred its matrices of interaction coefficients at multiple instants, assumed that they

constitute the adjacency matrix of a time-evolving graph that characterizes our underlying dynamical system, and evaluated the time-evolving multilevel indices of frustration.

We realized that as the time series stabilizes, the indices of frustration indicate a state of balance by admitting an index of 1. We explored this state for a system that has been modeled as a compartmental model, and for time series that have been taken from real COVID-19 data. We also explored the limitations of our method and the impact of the number of dynamical variables on our method. For the sake of increased accessibility, we combined the code for our method in a library called `FrustrationDynamiX` that was explained in Appendix A.

As a general reflection, we realize that the macroscale index has the most information, yet it is the most computationally expensive one to compute. We realize that the microscale index of frustration can be less expensive, and it captures some important phenomena for complete graphs. As such, one might want to aim to compute the macroscale index of frustration of their system to capture more phenomena in the system. This motivates the future work to explore parallelizations of the macroscale index of balance, ILP problems or alternative formulations.

We can also suspect that edge weights should have an impact on the measures of frustrations, and as such we should explore the weighted analogs of all the proposed algorithms and study the impact of weights on our dynamical systems.

Our work essentially contributed to a massive field that is constantly growing with the growth we are witnessing the data-driven modelling, and further explorations appear to be rewarding to the community.

Appendix A

FrustrationDynamiX Library Documentation

This appendix serves the purpose of giving the reader a basic tutorial on calling the FrustrationDynamiX library in Python to generate results using the tool.

Installing the Package

To install the package and use it in Python locally, in the command prompt, call

```
pip install FrustrationDynamiX
```

If the package is to be used in Google Colab or Jupyter Notebook, in a coding block call

```
!pip install FrustrationDynamiX
```

Importing the data

The data must have the following format:

time	X_1	X_2	$\cdots$	X_n
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table A.1: The format of the data of the input timeseries

When the format of the data is like the one described in A.1, we can use Pandas to import it. If it is stored in a CSV file we would

```
import pandas as pd
df = pd.read_csv(...)
```

Initializing the FrustrationDynamiX Object

When we have the data ready, we can initialize an instance of FrustrationDynamiX

```
from FrustrationDynamiX import *
fdx = FrustrationDynamiX(time_series_df, is_normalized=False
    )
#time_series_df is the pandas dataframe with the data
#is_normalized is an optional argument initialized as False
    by default, but may be initialized as True in case the
    data is already normalized between 0 and 1
```

Data Preprocessing

If the data is not normalized, there are two options to normalize the time series between 0 and 1.
First option is to divide all values by the absolute value of the maximum across all values of the timeseries

```
fdx = fdx.normalize_series_max()
#No argument is needed
```

Second option is to allow the user to provide a value that divides all values of the time series. This value must be real and positive

```
fdx = fdx.normalize_series_constant(value)
#value is the user-defined real and positive number to
    normalize the series
```

We can also choose to round the timeseries to a certain number of decimal places, and this feature is recommended to be used after normalizing

```
fdx = fdx.round_series(number_figures)
#number_figures is a positive integer that determines the
    number of decimal places to be taken
```

Frustration Computation

We can compute the frustration indices for a time series by calling the following methods.
For the microscale index of frustration, call

```
time_vector, triadic_balance, balanced_triangles,
    unbalanced_triangles = fdx.compute_triadic_evolution(
    window_size = None, showPlots = True)
#window_size is the size of the interval to be taken,
    initialized to number of variables + 2 by default
```

```
#showPlots (True by default) allows the user to view the
   SMap fitness plots
#time_vector contains the values of time for which balance
   was evaluated
#triadic_balance contains the microscale indices of balance
   computed as balanced_triangles./(balanced_triangles +
   unbalanced_triangles)
```

For the macroscale and mesoscale indices of frustration, call

```
time_vector, F, Z, C, D = compute_frustration_evolution(
   method, window_size = None, showPlots = True)
#method is either "XOR", "ABS" or "AND"
#window_size and showPlots as previously defined
#time_vectors contains the values of time for which balance
   was evaluated
#F contains the macroscale index
#Z contains the size of the edge deletion set
#C and D contain the cohesiveness and divisiveness
   respectively
```

Plotting

We can either plot the raw timeseries, or the time series overlapped by one of the computed indices.
To plot the raw timeseries, call

```
fdx.plot_series(xlabel = "Time", ylabel = "Variables", title
    = "Time series", save_plot = None)
#Where the parameters determine the plot settings
#save_plot takes a string that saves the plot in the name of
    the string if given
```

To plot the timeseries with an index, one must have called the Frustration computation function corresponding to the desired scale of the index before calling the following function

```
fdx.plot_frustration_series(method, xlabel = "Time", ylabel
   = "Variables and Frustration", title = "Frustration vs.
   Time", save_plot = None)
#Where the method is "MIC" for microscale, "MAC" for
   macroscale, and "MES" for cohesiveness and divisivess
#save_plot takes a string that saves the plot in the name of
    the string if given
```

Note that a warning will be issued in case the timeseries was neither normalized using one of the preprocessing functions nor normalized before instantiation. The warning is due to the fact that the frustration index may not appear properly on the plot due to the variation in the scales.

Remarks

The package includes many methods that can compute the index of frustration for a single graph. In fact those methods were used as helper methods to call the functions outlined above. However, this appendix only serves the purpose of providing the user with a documentation for using the package to produce results similar to the one produced in the research.

Bibliography

[1] P.-M. Binder, "Frustration in complexity," *Science*, vol. 320, no. 5874, pp. 322–323, 2008.

[2] M. Mitchell, *Complexity: A guided tour*. Oxford university press, 2009.

[3] Y. Bar-Yam, *Dynamics of complex systems*. CRC Press, 2019.

[4] P. Érdi, *Complexity explained*. Springer, 2008.

[5] M. Mitchell, "Complex systems: Network thinking," *Artificial Intelligence*, vol. 170, no. 18, pp. 1194–1212, 2006, Special Review Issue, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2006.10.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000437020600083X.

[6] A. Barabási, *Linked: The New Science of Networks* (Business book summary). Perseus Pub., 2003, ISBN: 9780452284395. [Online]. Available: https://books.google.com.lb/books?id=rydKGwfs3UAC.

[7] D. Watts, *Six Degrees: The Science of a Connected Age* (Six Degrees: The Science of a Connected Age). Vintage, 2004, ISBN: 9780099444961. [Online]. Available: https://books.google.com.lb/books?id=Qc9LtrmkrIgC.

[8] L. Manikonda, Y. Hu, and S. Kambhampati, "Analyzing user activities, demographics, social network structure and user-generated content on instagram," *arXiv preprint arXiv:1410.8099*, 2014.

[9] S. F. Sampson, *A novitiate in a period of change: An experimental and case study of social relationships*. Cornell University, 1968.

[10] K. E. Read, "Cultures of the central highlands, new guinea," *Southwestern Journal of Anthropology*, vol. 10, no. 1, pp. 1–43, 1954.

[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2010, pp. 1361–1370.

[12] S. Aref and M. C. Wilson, "Balance and frustration in signed networks," *Journal of Complex Networks*, vol. 7, no. 2, pp. 163–189, 2019.

[13] T. Antal, P. L. Krapivsky, and S. Redner, "Social balance on networks: The dynamics of friendship and enmity," *Physica D: Nonlinear Phenomena*, vol. 224, no. 1-2, pp. 130–136, 2006.

[14] E. Ferreira, S. Orbe, J. Ascorbebeitia, B. Álvarez Pereira, and E. Estrada, "Loss of structural balance in stock markets," *Scientific Reports*, vol. 11, no. 1, p. 12 230, 2021.

[15] Z. Bodie, A. Kane, and A. Marcus, *Ebook: Investments-global edition*. McGraw Hill, 2014.

[16] F. Harary, M.-H. Lim, and D. C. Wunsch, "Signed graphs for portfolio analysis in risk management," *IMA Journal of management mathematics*, vol. 13, no. 3, pp. 201–210, 2002.

[17] D. Lai, "Alignment, structural balance, and international conflict in the middle east, 1948-1978," *Conflict Management and Peace Science*, vol. 18, no. 2, pp. 211–249, 2001.

[18] P. Doreian and A. Mrvar, "Structural balance and signed international relations," *Journal of Social Structure*, vol. 16, no. 1, pp. 1–49, 2015.

[19] J. PEVEHOUSE, T. NORDSTROM, and K. WARNKE, "The correlates of war 2 international governmental organizations data version 2.0," *Conflict Management and Peace Science*, vol. 21, no. 2, pp. 101–119, 2004, ISSN: 07388942, 15499219. [Online]. Available: http://www.jstor.org/stable/26273548 (visited on 04/23/2024).

[20] Z. Maoz, P. L. Johnson, J. Kaplan, F. Ogunkoya, and A. Shreve, "The dyadic militarized interstate disputes (mids) dataset version 3.0: Logic, characteristics, and comparisons to alternative datasets," *Journal of Conflict Resolution*, 2019, forthcoming.

[21] B. DasGupta, G. A. Enciso, E. Sontag, and Y. Zhang, "Algorithmic and complexity results for decompositions of biological networks into monotone subsystems," *Biosystems*, vol. 90, no. 1, pp. 161–178, 2007.

[22] K. Oda, Y. Matsuoka, A. Funahashi, and H. Kitano, "A comprehensive pathway map of epidermal growth factor receptor signaling," *Molecular systems biology*, vol. 1, no. 1, pp. 2005–0010, 2005.

[23] K. Oda, T. Kimura, Y. Matsuoka, A. Funahashi, M. Muramatsu, H. Kitano, *et al.*, "Molecular interaction map of a macrophage," *AfCS Research Reports*, vol. 2, no. 14, pp. 1–12, 2004.

[24] S. Gama-Castro, H. Salgado, M. Peralta-Gil, *et al.*, "Regulondb version 7.0: Transcriptional regulation of escherichia coli k-12 integrated within genetic sensory response units (gensor units)," *Nucleic acids research*, vol. 39, no. suppl_1, pp. D98–D105, 2010.

[25] B. A. Cipra, "An introduction to the ising model," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987.

[26] G. Facchetti, G. Iacono, and C. Altafini, "Computing global structural balance in large-scale signed social networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 52, pp. 20 953–20 958, 2011.

[27] B. A. Cipra, "The ising model is np-complete," *SIAM News*, vol. 33, no. 6, pp. 1–3, 2000.

[28] P. Bartashevich and S. Mostaghim, "Ising model as a switch voting mechanism in collective perception," in *Progress in Artificial Intelligence: 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3–6, 2019, Proceedings, Part II 19*, Springer, 2019, pp. 617–629.

[29] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2, pp. 1–3.

[30] S. Aref, L. Dinh, R. Rezapour, and J. Diesner, "Multilevel structural evaluation of signed directed social networks based on balance theory," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[31] S. Aref, A. J. Mason, and M. C. Wilson, "A modeling and computational study of the frustration index in signed networks," *Networks*, vol. 75, no. 1, pp. 95–110, 2020.

[32] A. Benjamin, G. Chartrand, and P. Zhang, *The fascinating world of graph theory*. Princeton University Press, 2017.

[33] W. H. Wen-Mei, D. B. Kirk, and I. El Hajj, *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 2022.

[34] S. Aref and M. C. Wilson, "Measuring partial balance in signed networks," *Journal of Complex Networks*, vol. 6, no. 4, pp. 566–595, 2018.

[35] D. Cartwright and F. Harary, "Structural balance: A generalization of heider's theory.," *Psychological review*, vol. 63, no. 5, p. 277, 1956.

[36] D. Easley, J. Kleinberg, *et al.*, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge university press Cambridge, 2010, vol. 1.

[37] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[38] S. Storandt, "Approximation algorithms in the successive hitting set model," in *Algorithms and Computation: 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings 26*, Springer, 2015, pp. 453–464.

[39] G. Datseris and U. Parlitz, *Nonlinear dynamics: a concise introduction interlaced with code*. Springer Nature, 2022.

[40] S. H. Strogatz, *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[41] L. Perko, *Differential equations and dynamical systems*. Springer Science & Business Media, 2013, vol. 7.

[42] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[43] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[44] S. Cenci, G. Sugihara, and S. Saavedra, "Regularized s-map for inference and forecasting with noisy ecological time series," *Methods in Ecology and Evolution*, vol. 10, no. 5, pp. 650–660, 2019. DOI: https://doi.org/10.1111/2041-210X.13150. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13150. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13150.

[45] C.-h. Hsieh, S. M. Glaser, A. J. Lucas, and G. Sugihara, "Distinguishing random environmental fluctuations from ecological catastrophes for the north pacific ocean," *Nature*, vol. 435, no. 7040, pp. 336–340, 2005.

[46] E. R. Deyle, R. M. May, S. B. Munch, and G. Sugihara, "Tracking and forecasting ecosystem interactions in real time," *Proceedings of the Royal Society B: Biological Sciences*, vol. 283, no. 1822, p. 20 152 258, 2016.

[47] P. A. Dixon, M. J. Milicich, and G. Sugihara, "Noise and nonlinearity in an ecological system," in *Nonlinear dynamics and statistics*, Springer, 2001, pp. 339–364.

[48] G. Sugihara, "Nonlinear forecasting for the classification of natural time series," *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, vol. 348, no. 1688, pp. 477–495, 1994.

[49] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[50] G. G. Walter and M. Contreras, *Compartmental modeling with networks*. Springer Science & Business Media, 1999.

[51] M. Chappell, "Compartmental modelling," ES4A4 Biomedical Systems Modelling Seminar, University of Warwick, 2023.

[52] P. S. Brachman, "Epidemiology," in *Medical Microbiology*, S. Baron, Ed., 4th ed. Galveston, TX: University of Texas Medical Branch at Galveston, 1996, ch. 9. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK7993/.

[53] L. J. Allen, "Some discrete-time si, sir, and sis epidemic models," *Mathematical biosciences*, vol. 124, no. 1, pp. 83–105, 1994.

[54] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[55] J. a. Park, *jpyEDM*. [Online]. Available: https://github.com/SugiharaLab/jpyEDM.

[56] E. Mathieu, H. Ritchie, L. Rodés-Guirao, *et al.*, "Coronavirus pandemic (covid-19)," *Our World in Data*, 2020, https://ourworldindata.org/coronavirus.