SEMANTIC AND TEMPORAL GRAPH NEURAL NETWORKS FOR SUPPLY CHAIN RISK QUANTIFICATION

by

SVETLE MATOVSKI

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN SYSTEM ENGINEERING

2023

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Nasim Nezamoddini, Ph.D., Chair
Sankar Sengupta, Ph.D.
László Lipták, Ph.D.
Huirong Fu, Ph.D.

To my mother, father, wife, and children

ACKNOWLEDGMENTS

I'm extremely grateful to Nasim Nezamoddini, Ph.D., my chair, without her I would never have been able to complete my journey. She kept me focused on the task and encouraged me to not give up. Thank you for advising me during my journey. This endeavor would not have been possible without my committee Sankar Sengupta, Ph.D., László Lipták, Ph.D., and Huirong Fu, Ph.D.. I'm grateful that they chose to stand beside me on this journey. Thank you, Qian Zou, Ph.D., for allowing me to complete my journey.

I would like to extend my sincere thanks to Chris Wagner, Ph.D.. He helped me get through my exams and jumpstarted my research. I want to thank Patrick Dessert, Ph.D., who courage me to start this journey and advised me in the very beginning. Special thanks to Robert Van Til Ph.D. for suggesting Nasim Nezamoddini Ph.D. as my chair when I was in need. Thank you, Maureen Callaghan, and Oakland University Writing Center for helping me out.

Finally, I would like to thank my wife, my children, my parents, and my friends without their support and encouragement I would never have made it this far. I will close with my favorite quote by Brandon Sanderson "Journey before destination" because it was really about the journey to this point in my academic career.

Svetle Matovski

ABSTRACT

SEMANTIC AND TEMPORAL GRAPH NEURAL NETWORKS FOR SUPPLY CHAIN RISK QUANTIFICATION

by

Svetle Matovski

Adviser: Nasim Nezamoddini, Ph.D.

Calculating supply chain risk management values requires a granular set of parameters. Failure risk at each supply chain entity is a dependent value influenced by entities within a supply chain. Predicting future risk values from historical data is based on trends and patterns therein. On the surface, historical data does not show how the data interlink and connects.

To understand the problem this research starts by examining the underlying data structure that a supply chain uses to store data. Mining the data requires understanding its relationship to other data within the structure. This understanding allows the system or user to make better decisions.

The research brings together four different pieces to show a risk value at each node within a supply chain network. This research generates a dataset using seed data and trend data to build a supply chain network. This research then predicts future risk values at a node level using ARMIA and a graph neural network. Node centrality values help show the importance of nodes to a supply chain. The centrality methods this research uses are betweenness, degree, eigenvector, and Katz centrality. Each one gives an importance value based on a different view of centrality. This research then looks for the

influence of a node on a customer. Calculating an influence value by using Bayesian networks. The last value calculated is the profit loss at a customer node.

All these different pieces are brought together in the risk calculation equation to give a final node-per-node risk value. The risk calculations align well with the structure of a graph. This research will show that a graph neural network and a graph database are useful for supply chain problems.

TABLE OF CONTENTS

TABLE OF CONTENTS – Continued

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES—Continued

LIST OF ABBREVIATIONS

GNN	Graph Neural Network
ANN	Artificial Neural Network
LSTM	Long Short Term Memory
MPNN	Message Passing Neural Network
RDMS	Relational Database Management System
DBA	Database Administrator
AI	Artificial Intelligence
RDB	Relational Database
SQL	Structured Query Language
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
W3C	World Wide Web Consortium
OWL	Web Ontology Language
GCN	Graph Convolution Network
SCRM	Supply Chain Risk Management
CART	Classification and Regression Tree
k-NN	k-Nearest Neighbor
RF	Random Forest
MLP	Multilayer Perception
ARIMA	Autoregressive Integrated Moving Average
GAT	Graph Attention Network

ReLU	Rectified Linear Unit
AdaGrad	Adaptive Gradient Algorithm
RMSprop	Root Mean Square Propagation
MSE	Mean Square Error
GCONV	Graph Convolution
GRU	Gated Recurrent Unit
MANOVA	Multi-variate Analysis of Variance
NP-Hardness	Non-deterministic Polynomial-time Hardness

CHAPTER ONE

Introduction

The chapter presents an overview of how data has outgrown traditional data storage systems. The next two subchapters detail the different data storage systems. Subchapter 1.3 states that a larger dataset requires a cognitive style of data storage, retrieval, and analysis. An area where a cognitive data system would be useful is supply chain data management and risk, which subchapter 1.4 details. Subchapters 1.5 and 1.6 detail the problem and the significance of the research topic. The research overview summarizes how the research presents the rest of the information found in this paper.

1.1 Overview

We as a civilization create a sizeable amount of data every day, and data scientists believe they can mine data for useful information. In 2020, the world has created 59 zettabytes of data and we are on track to reach 74 zettabytes by 2021 (Holst, 2021). The supply chain industry is experiencing similar growth in the amount of data being created. A supply chain's data structure is growing because every product, machine, worker, etc. can now create data. Studying supply chain data can show where to build the next warehouse, fast delivery routes, and what items should be stocked based on location and season. Risk data is a subset of supply chain data that is very relationship rich and needs to be processed correctly so knowledge can augment the decision-making tasks. Large companies have supply chain data, but it is only part of the bigger data picture. Large companies generate purchase orders, product requirements, product testing reports, customer reports, etc. On the outside, this data looks to be separate and not dependent on

one another, but that is a mistake. Decisions on parts, captured in a bill of materials and product requirements, affect the ability of a supply chain to function, which is quantified as risk. Outside social, environmental, and political factors can add or reduce supply chain risk. Gone are the days when simple supply and demand models can predict when to build a product. Many factors are linking entities together, but their connections are not visible to the untrained eye. Data is only recently being brought together and viewed as a larger set. The act of bringing the data together is usually a manual process and finding the connections is being augmented by deep learning algorithms but solutions are unique. The research describes a more generic approach for combining traditional datasets in a more interconnected model, usable for finding dependencies and causation relationships. Helping to predict values when data depends on predecessor values.

Making the data above useful involves storing the data in a computer system. A computer system will use a database as a repository for the data. The stored data will have a data model applied to it. The data model will define how to represent the data and the methods used to extract and manipulate the data. A database administrator (DBA) will add data to the database or create a system for adding data. In extracting and analyzing the data, they will need a DBA to accomplish the task. Data usually needs to be migrated to different systems or warehoused for analysis. The migration could miss key data elements and disrupt the flow of data. The research describes a data model that is dynamic and difficult to disrupt.

Traditional databases usually analyze a data snapshot or offline record. Recently, artificial intelligence algorithms analyze the data for better understanding but

understanding how an AI builds its results is a task of its own. The research explores a theory of crafting a database that may more closely represent the structures used by an artificial intelligence algorithm giving a window into the build process so we can better understand the results and its thought process.

Interconnecting datasets, viewing data plus connection together, embedding artificial intelligence processes, and building a more dynamic database system are the key points that the research is exploring and describes the building blocks of a cognitive database system. The next section discusses computer databases as a foundation for a new database model.

1.2 Databases

A database is a computer system used to store electronic information and data. The data can spread across multiple hard drives and computer systems. The database management system will access all data locations. All databases represent their data within a structure or schema. A schema is a data model of how to store information on a computer system. The first databases were flat file systems. A flat file system is a text file that organizes data by using delimiters such as tabs or commas, storing only one record per file.

Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

Figure 1.1 Flat-File Database from U.S. Department of Transportation

A flat file is not a true database next came hierarchy and network map databases. Hierarchical databases follow a tree structure with root parent and child leaves.

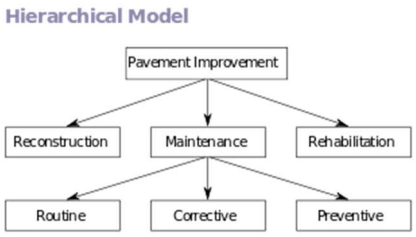

Figure 1.2 Hierarchical Database from U.S. Department of Transportation

A parent or child represents one record in the database. A parent can have multiple children, but a child can only have one parent. Traversing the database requires starting at the root and moving down the records. A network database is more complex than a hierarchical database because it allows children to have multiple parents.

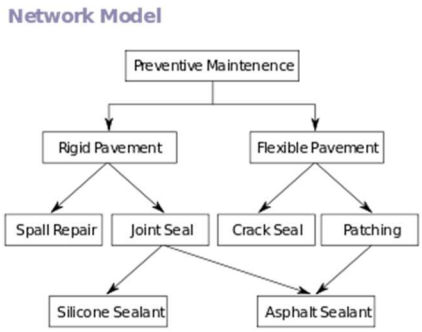

Figure 1.3 Network Database Model from U.S. Department of Transportation

The structure of the network database represents a generalized graph linking records by relationships, but the relationships within a hierarchical or network database

hold no data and only link records together. Relationships without meaning or purpose lose their value quickly. Network databases are supposal easier to model because of the graph structure, but IBM, a major database developer, never adopted the idea, and the relational database displaced both in the 1970s. Almost every organization uses relational databases (RDB). Retrieving and analyzing data is the job of a relational database management system (RDBMS). A human database administrator (DBA) accomplishes all the work. The reason relational databases have a DBA is that an RDBMS and an RDB schema need to be learned. Using an RDBMS and RDB efficiently requires training and skills an average user does not possess. The relational model does not handle big data efficiently, this has forced organizations to create alternative data models (Kuznetsov et al., 2014). One of these new database systems is a graph database. The schema of a graph database is a graph theory graph. A graph theory graph, G, is a set of vertices, V, and edges, E, G = (V, E). Vertices, also known as Nodes, represent data objects, and edges represent relationships between objects. Edges can carry meaning and value, then extracting and analyzing the data follows graph theory methodologies. The research shall explore the idea that a graph is similar to the way humans cognitively store and structure data. In the next subchapters, the research details RDBs and graph databases.

1.2.1 Relational Databases

Storing data in relational databases (RDB) began in the 1970s. First proposed by E. F. Codd (1972, 1983) as the relational data model and relational algebra is the basis for data extraction and manipulation. The data structure housed in an RDB and the methods to access that data follows E. F. Codd's relational data model closely.

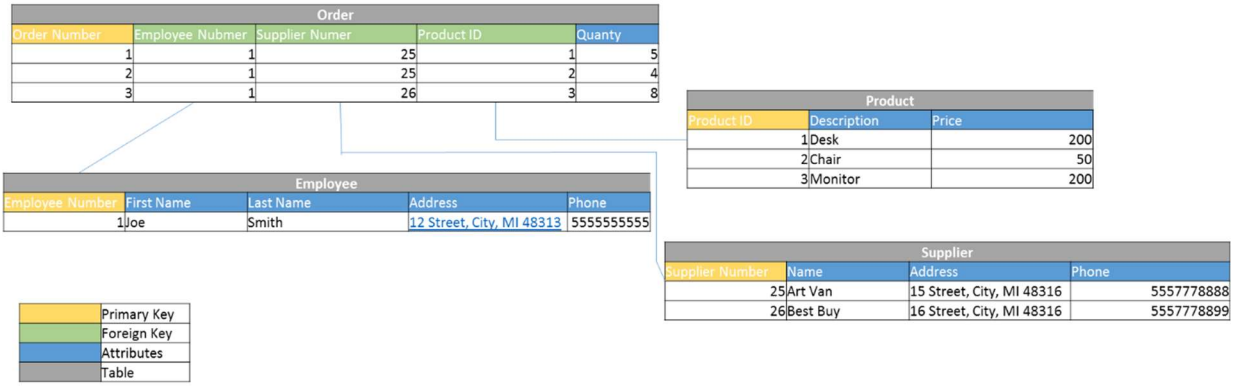

Figure 1.4 Relation Database Data Model

Figure 4 shows an RDB in a visual format. The gray boxes represent database entities. Each entity in an RDB is a unique table. An RDB table has columns of attributes describing that entity. One or more columns in each table are a unique identifier or primary key. The primary key is used to reference that row in other entities (foreign key) and used as a reference for functions. Storing foreign keys in an entity's columns is one way of storing relationships within an RDB. Each row in an RDB table represents an instance of that entity. A table by itself is a relation since it is a group of related instances. Relationships within an RDB can be one-to-one, one-to-many, or many-to-many. A recursive many-to-many-to-many and so forth relationship is possible with multiple foreign keys. One idea to keep in mind is relationships in an RDB are virtual. An RDB relationship does not directly show the relationship meaning between tables or columns within a table. Users infer RDB relationship meaning through context or past knowledge. Increasing the number of relationships between tables creates confusion, takes up space, and could create redundant data. Ideally, minimizing relationships will help to speed up

processes. Minimizing repetitive data between tables is another good practice, but merging RDBs and migrating data can create repetitive data. The building, maintaining, changing, and extracting of information requires a human database architect (DBA). A DBA will take a human request and use a relational database management system (RDBMS) to extract the requested data. Data accessing uses eight basic functions, union, intersection, difference, Cartesian product, selection/restriction, projection, join, and relational division, all come from relational algebra. Union merges two or more tables and removes duplicates. Intersection creates an entity of shared attributes between two tables. Difference creates an entity with attributes found in the first table but not in the second table. The Cartesian product joins two tables together without removing duplicate data. Selection/restriction returns specific attributes from a table. Projection extracts attributes from two tables using common attributes as the selection criteria. The relational division is the opposite of the Cartesian product. Structure Query Language (SQL) is the interface that most RDBMSs execute for information extraction, manipulation, and analysis. Case in point if a user needs to know what products each employee has ordered. A DBA would write a SQL program that would extract data from “Employee”, “Order”, and “Product” tables using Relational Algebra functions. The reason so many tables need to be referenced is that a DBA must follow the path from “Employee” to “Product” tables. Writing an SQL program requires training and education in the mathematical theory of relational algebra. Relational algebra is the foundation of SQL and RDBs. RDBMS implements some axioms of relational algebra, but not all.

SQL is difficult for an average database user to understand. The information above shows how the data structure in an RDB is not intuitive to a human user since the data organization is foreign to a user without training in the data structure. SQL programming can be simple for a user with the correct training, but an average user would not know where to start. Two other problems RDBs face are relational algebra does not handle large data sets efficiently and tables are rigid structures. Defining a table is a one-time activity. If a table's definition changes, then a new table must replace the old one. The research shall show that graph databases are a more intuitive and dynamic data structure. Graph databases can manage large data sets with ease and change existing objects by adding relationships.

1.2.2 Graph Database

Graph databases use a graph structure as the data model for storing data. The underlying concept is N nodes and E edges together make a G graph. Nodes are a place where edges converge. Edges connect one or more nodes. Graph Databases have many graph styles, which include property graphs, hyper-graphs, and resource description framework (RDF). One of the most generic graph database structures is a property graph model. Nodes represent data objects that hold attributes and values, and edges are relationships that connect data objects. Edges can hold their own values, allowing edges to affect nodes. Labels for nodes and edges are a requirement. The right labels help to create a semantic network of objects. A semantic network is a collection of concepts linked using semantic relationships. In computer science, a semantic network is also a knowledge representation or ontology. Querying the graph for information is a semantic

process. One of the primary advantages of a graph database is you can query for related information multiple layers deep without a large processing cost. For example, say you want to know Bob's friends. The database would look up Bob and all the friends' relationships with Bob. The database could also look up a friend of a friend type of relationship with Bob going multiple layers deep.

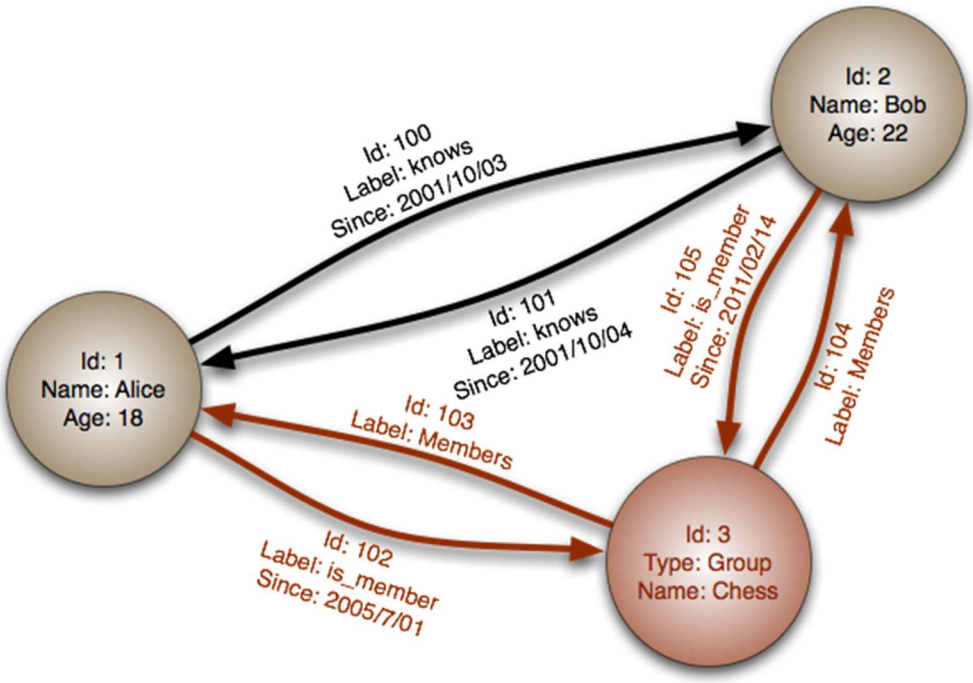

Figure 1.5 Graph Database Using a Property Graph Model

A specific query language called Cypher, which is a declarative graph language, is used to retrieve and manipulate data stored in the Neo4j graph database. When using Cypher, a user declares what he needs the graph to accomplish. Other graph databases use different query languages. The graph database industry is currently trying to unify the query language. HypergraphDB is exactly what the name implies a hypergraph database

model, very similar to property graphs with one added feature. Hypergraphs are a collection of nodes and edges, but one edge may point to multiple nodes. Figure 7 is an example of a hypergraph. Each graph can be a node with an edge connecting it to another graph. Edges can branch and merge to reach different nodes and graphs. A property graph could mimic a hypergraph by having special nodes that only branch or merge relationships before heading to other nodes.

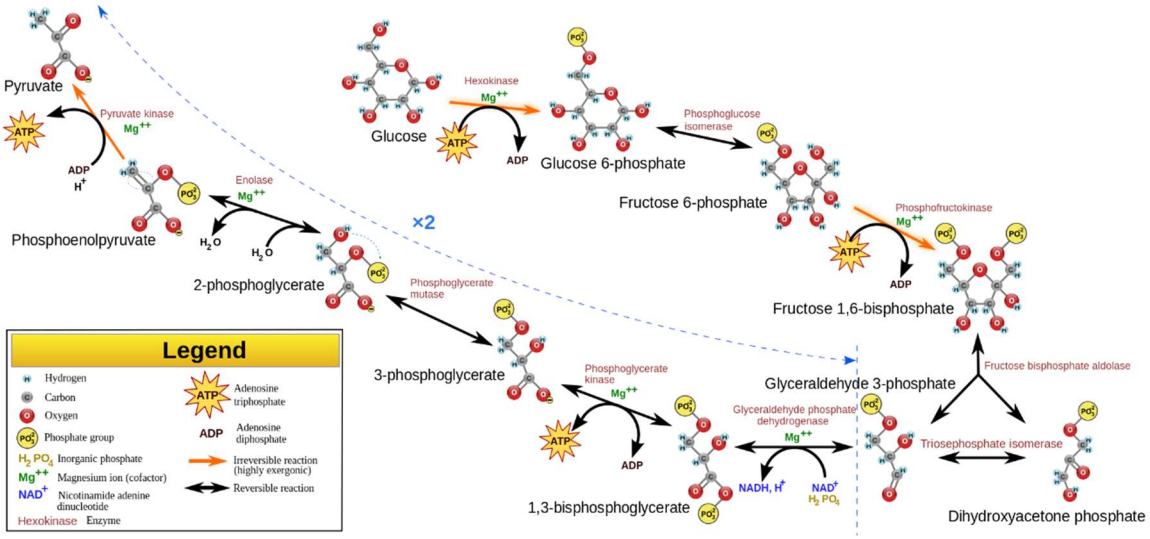

Figure 1.6 Steps of Glycolysis is a Hypergraph Structure (2023)

RDF by World Wide Web Consortium (W3C) is a model of how websites link. RDF uses a semantic query language based on subject-predicate-object, conjunctions, disjunctions, and optional patterns or SPARQL.

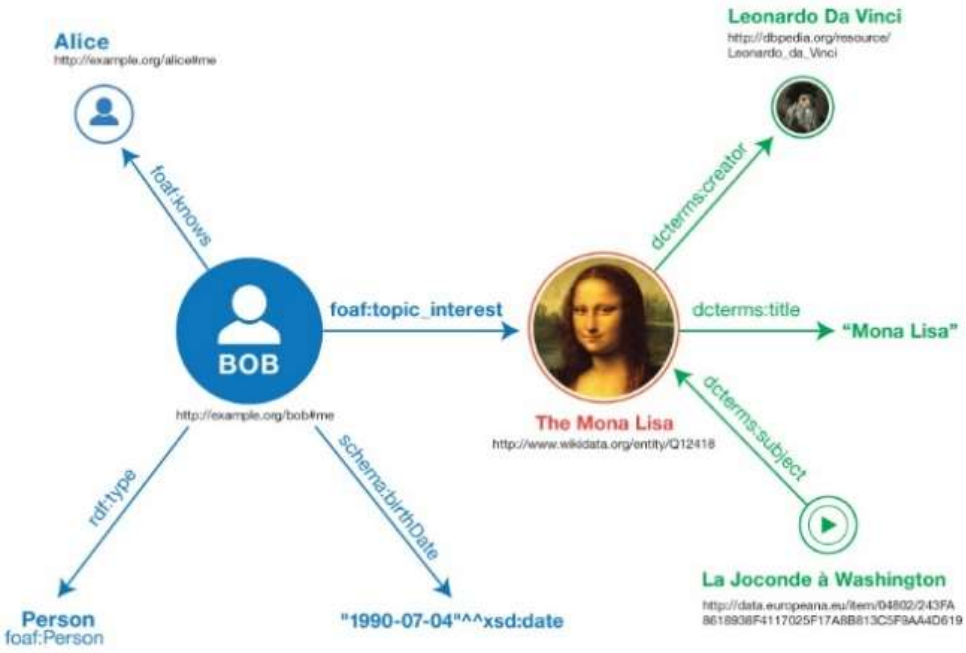

Figure 1.7 Informal Graph of the Sample Triples (W3C, 2014)

SPARQL definition is SPARQL Protocol and RDF Query Language. An RDF stores all data as a triple data group, subject-predicate-object, which can be restrictive to computational models. RDFs are good at linking objects together, but the links do not carry value only meanings based on the edge label. RDFs are usually used to link unstructured data together, web pages, text files, etc. Graph databases excel at being dynamic. For example, say you want to add new data to an existing object. All you need to do is create a relationship between the new piece of data to the existing object. Linking in new data to existing objects can help define an existing object more or morph an existing object into something new. The new relationship can be bi-directional, unlike an RDB, which only allows new data, and tables, to reference existing data. Graph databases do not have a common language, unlike RDBMSs, making the adoption of graph

databases harder. The core data structure of all graph databases is nodes and edges. A node can represent an entity, a document, or a single item. An edge is a relationship between two nodes at a minimum. Both nodes and edges need labels, correctly labeling nodes and edges in a graph database will create a semantic model. A semantic model is a conceptual data model with semantic meaning about the structure of the data. A graph database could represent a neural network of information with special nodes acting as a synapse, a node that evaluates data. Using a graph neural network, a graph database can train itself to understand the data structure of the database, opening the door to deep learning. Cognitive scientists believe humans store information in a semantic network (Sowa, 1992). Humans have an extensive set of data all linked by semantic relationships and we create subgraphs for specific problems and scenarios (Minsky, 1974) so that all our knowledge is within reach. Representing causal graphs is possible in a graph database and, using first-order logic, a database can infer information. Graph databases retrieve information similar to humans. Every time a user queries for information, they select a data node or relationship and then expand to other related data. Graph databases return a subgraph of linked information for the user to review similarly human recall of a specific memory and its related memories. Merging graphs and cognitive functions give the research a concept of a cognitive database.

1.3 Cognitive Database

A Cognitive Database falls into the large domain of Cognitive Systems. Cognitive Systems can be autonomous in behavior, perceive their environment or data, learn from the environment or data fed to the system, anticipate tasks or actions based on the

knowledge stored in the system, act on tasks given or self-created, and adapt to new environments or data. Satisfying all the previous items would give a computational system the ability to mimic human cognition. Many cognitive systems satisfy some pieces previously listed above. Some more difficult items to satisfy are learning, anticipating, and adapting to data. Learning has many definitions, but the simplest one is storing data useful for later use. The data being stored could be facts, values, or methods where a cognitive system is the storage repository. Non-cognitive databases excel at storing and retrieving data, a majority of the data is value base, within them but a Database Administrator (DBA) must structure the data, input the data, anticipate the need for the data, and adapt new data to the database. A cognitive system could structure its data, apply methods to stored data, adapt its structure to new data, and create data from stored data. Accomplishing the previous items would create a cognitive database system.

Current database systems are not cognitive by themselves. The system may seem cognitive because it has functions to analyze and present data to a user, but all that's happening is it retrieves data, algorithms analyze retrieved data and it presents it to the user, no cognitive learning processes apply to the data. Meaning the system hasn't learned why it provided the data to the user. The data is static, meaning the data doesn't change over time, and the data does not restructure with every injection of new data. Relational databases are some of the most static databases, their rigid data model does not allow the manipulation of a data model dynamically. Building a cognitive database will require a unique data structure, one that is easily reconfigurable over time, and that can store more than values and facts. The database will need to store functional models that

can apply to the data stored. The research describes a generic graph model for storing information and functions in a central location. A graph model will allow data to be reconfigurable, creating and destroying edges connecting data nodes at will by a system process. Functions can model a multipath graph where data travels the model to reach a determined outcome. Data generated by analysis will augment the model for use at a later time. Adding data will generate extra edges, and connecting new data to existing data will restructure the data stored in the system. The research's graph model will only cover bringing in data to create a new graph and adding new data to an existing graph. The research shows how a graph database can store a Bayesian network linking the data to a cognitive process, describing how to store a cognitive function. The research's system will learn the graph structure so it can predict values and understand data interdependence. In the subchapter, the research applies the concept of a cognitive database to supply chain data management.

1.4 Supply Chain Data Management and Risk

Describing a supply chain as "a network of entities such as manufacturer, suppliers, and distributors working together to transform goods from raw material to finished products while moving them to the end customer" (Wu et al., 2009). The network is heavily interdependent and losing any entity within the supply chain can disrupt the entire network. Supply chain managers quantify the disruption value as risk. Risk is the probability of an undesirable outcome happening at a point in time. In a supply chain, risk represents the probability of an entity failing to complete its task. For example, a supplier cannot deliver parts to a manufacturer. How critical an entity is to the

supply chain can either rise or lower its risk factor. Managing the risk as it propagates through a supply chain is part of supply chain management tasks. Managing the relationship between entities within a supply chain while increasing profitability and lowering risk is another. Resilience, the ability of a supply chain to handle disruption, is an important feature supply chain management focuses on improving and optimizing. Graphs and graph databases can help visualize the network of supply chain entities. Attaching risk values to each node and relationship allows a supply chain manager to see when an entity or relationship will fail. Analyzing the graph structure of a supply chain can show critical entities which can raise that entity's risk value. Alternative relationships, and paths, between entities, can help evaluate how redundancy can build a more resilient network (Chen et al., 2020). The reason new methods, like the methods detailed in this research, is because the datasets are very large. Supply chains for large companies can have hundreds or thousands of supply chain entities. Finding a single entity failure before it happens is difficult without advanced data mining techniques. This research explores these concepts for helping supply chain data management.

1.5 Research Problem

Traditional organizations store supply chain data within a relational database system. Large tables store values such as parts lists, cost, orders, parts, and location. The data is historic, meaning it is at least a day old. Algorithms exact data, run analysis, and update predictions, but all these operations require time. The operations usually run on multiple systems, which can create information gaps. Sometimes the amount of data requires migration to a separate system. This migration may miss key data and the

operations do not grow in knowledge. The research purposes a cognitive database to replace these multiple systems with the ability to learn from its data. A migration to a graph database is the first step. First, the data will need to be migrated to a graph style database, so the information represents a property graph, knowledge graph, semantic network, or ontology. These terms are usually interchangeable, which is like how humans store knowledge (Tulving, 2002). Relational Databases do not include semantic information in their implementation, some schemas may show semantic relationships, but most do not. By moving the relational database to a graph database, the missing semantic information will need to be created. The semantic information will add another level of data to the model, which helps to add new data to existing graph databases. The conversion is a hard problem because a system must infer the relationships with only the knowledge within the database. The problem with inferring the relationships comes from not understanding the circumstances around the creation of an RDBs scheme. Reducing the data context issue by using an outside source of knowledge is a good idea, but restricting the domain is a good practice. Most RDBs schemes are foreign to a human's understanding, hence a direct conversion to a semantic network would not create a meaningful graph structure. Figure 9 illustrates how an RDB structure differs from retrieving information from a semantic model. The User and the DBA both think in a semantic model structure. The user understands the concept behind the data they are looking to retrieve and explains the data to a DBA. The DBA translates the User's concepts and needs into RDB data structures and SQL queries. A simple query could lead

to a lot of work with a large return of data since the database stores many data types that have a similar meaning in human language or human data structures.

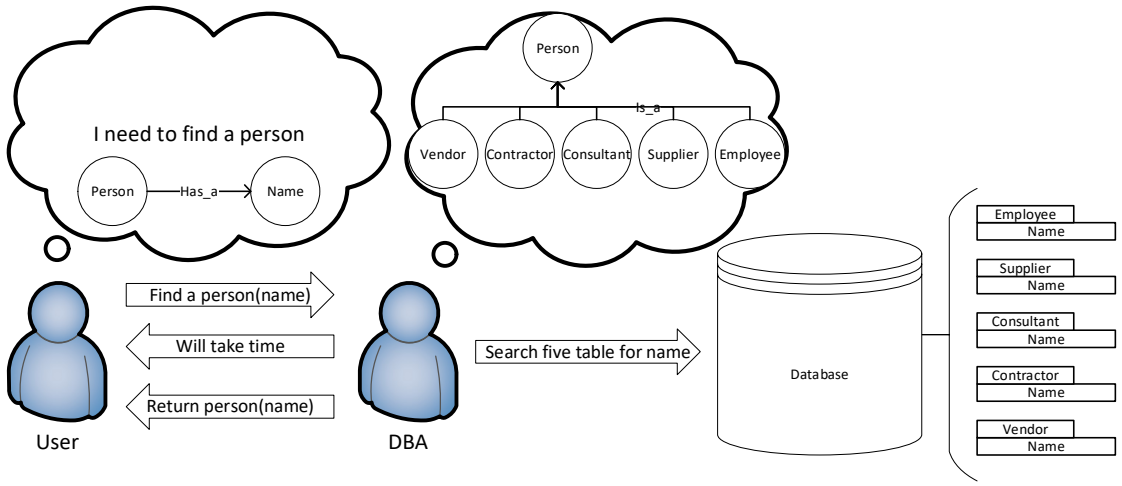

Figure 1.8 The Translation of a Human Request for a Relation Database Context

A semantic network would define a generic concept to give context to the data. The relationship between the generic concept and the target data object would create a probability of how related an object is to a concept. Using the probability plus the context will retrieve the most likely answers to a query. Representing the structure as a graph with vertices as concepts and edges as semantic relationships. Grouping vertices and edges will create a subgraph that defines a complex concept, such employee.

Figure 1.9 shows a basic semantic network with considerably basic relationships connecting base concepts. The relationships show probability without actual values, “Is_a” is 100% while “Could be” is 50%. The vertices can store data of any type to help augment a retrieval. Data stored in nodes does not have to be text or numerical values.

The data could be images or audio files representing different data items that the system understands similar to a human.

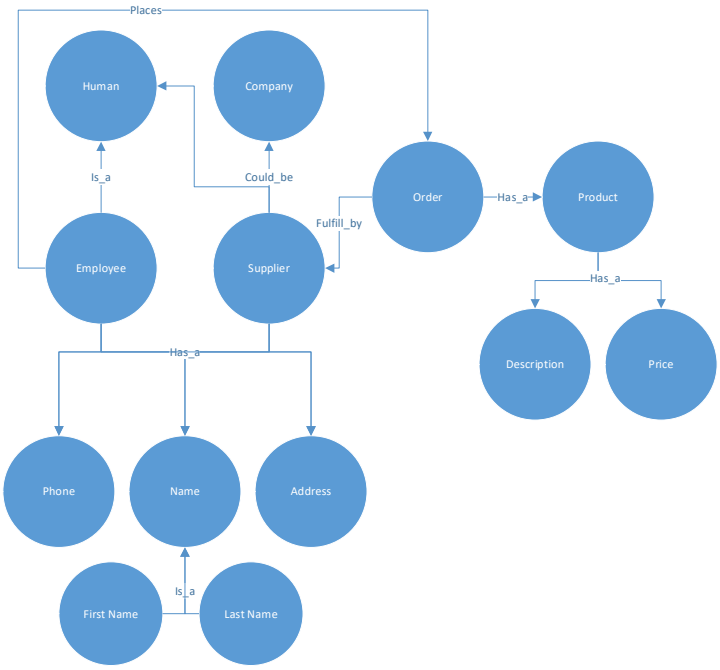

Figure 1.9 A Semantic Network of Suppliers, Employees, and Company

The migration of a traditional data structure to a graph will combine, remove, and transform data into new structures. The research will not lose any data, only represent it in a new way. The new graph structure shall show how the data is interdependent and how grouping data nodes together can build larger data objects. RDBs are very rigid structures where linking data is a predetermined path and changes require migration to a new schema. Losing data during migration is possible. A graph database generates a data path during a query because most relationships will have an applied direction. Changing the path by adding directed edges to vertices does not create a possibility of losing data

since vertices do not store edges. The edge itself stores the vertices, making edge creation and deletion a zero-impact function to the data stored. Edges are genuine relationships and data flows between vertices, unlike relationships within RDBs. Relationships in an RDB are a lookup index with no real meaning. Edges can hold their own data and augment the data as it flows from one vertex to another vertex, making a subgraph into a dynamic answer to a query. Moving the rigid RDB scheme to a more dynamic graph style structure will allow the database to flow data dynamically, change to circumstances imposed on it by the user of the data and allow restructuring of data when adding new data. The user can add data from other areas to an existing graph by adding a few key relationships to existing nodes. Another advantage is the graph structure can represent more than information. The graph database could show a method or process structure by an input data flow and become output data, for example, a Bayesian network. A graph database allows a deep learning model to learn the data structure for analysis. A graph is a common and generic data structure that allows the research to store more than values and keys.

1.6 Research Significant

The research shall provide a novel approach to converting supply chain data into a graph database and adding new data to the graph database. The problem is Non-deterministic Polynomial-time Hardness (NP-Hardness) so new processes are still possible to be found that may speed up the process or provide new insight. An NP-Hardness problem is a problem that is non-deterministic and not based in polynomial time. The reason for classifying the conversion process as NP-Hardness is that every

record requires a system to decide the meaning of the relationship between tables. Each decision will take N polynomial time to solve, which will be compounded by the number of relationships per record. A dynamic database that trains and learns from the data entered is a novel concept. This is a research area of interest (Bordawekar et al., 2017) Supply chain is one of the industry fields that relies heavily on relational databases, but the data is stored in multiple systems and the interconnection between the data is lost. Current systems link multiple data sources together, but some data is lost in the linking process, or the connections are not seen by the system. Migrating the data into a graph structure forces the interconnections to the surface showing how the data is relationship heavy. A graph can be reanalyzed for relationships, feature extraction, and discovering new relationships is possible. The discovery of new relationships is very useful to a supply chain since users may not have known the dependence beforehand. Adding outside data to the existing supply chain graph by connecting a few key points in the existing graph. For example, adding weather data to logistic data shows a correlation to a delay in shipping times. Existing nodes would not change, but relationships show a final shipping time node, which sums the estimated shipping time and delay for weather. The addition of data does not disrupt the graph, only adds to its knowledge. The research applies graph neural networks to learn the graph structure. After learning the structure, the system applies other neural models to predict future values.

The structure will need to be defined in such a way that data can flow from one vertex to another vertex with augmentation, so inferring and creating new data from the stored data. The research shows that a graph data structure is dynamic, and the data will

flow together into a cohesive structure. The concept of data flowing from one set of nodes to another and finally reaching an output node is similar to concepts found in network theory. A graph can describe the network structure with entity nodes and weighted edges. Algorithms follow a path from the input nodes to the output nodes, showing a change in data along the way. Training the graph structure to use historical data and updating the edge weights or node values is similar to a back-propagation method used in feed forward neural networks. The graph structure is useful for showing how risk mitigation uses a Bayesian network for risk propagation in a supply chain. A supply chain is a complex and relationship heavy data structure that is computationally heavy in a traditional database system. The supply chain in concept is similar to a graph with many interdependent parts. Representing supply chain processes within a graph structure is key, so they do not lose sight of the data being processed. Gaining new knowledge by linking outside data which was previously a human task. A common and generic structure for a supply chain within a graph is a relatively new field, along with the concept of using graph neural networks for analysis.

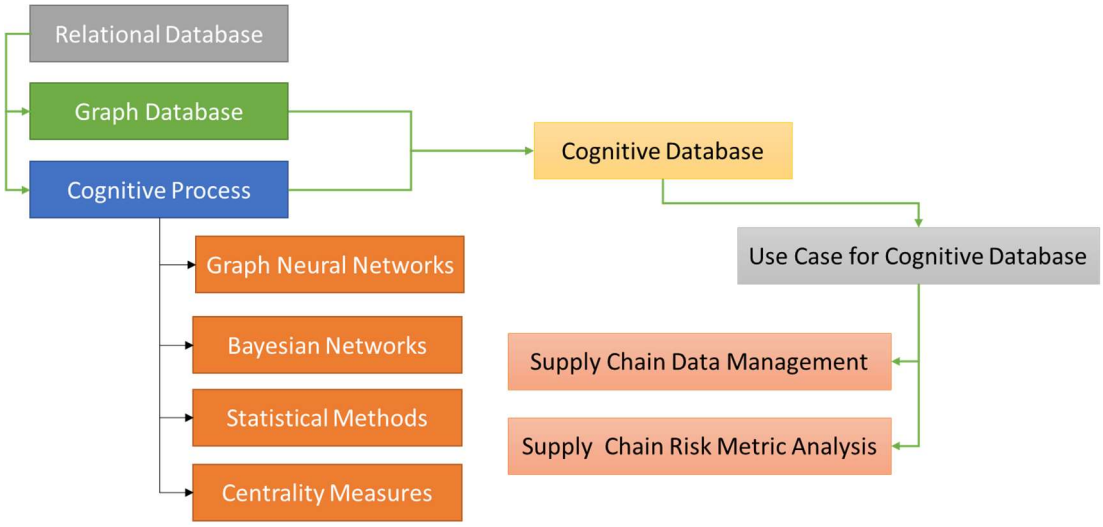

Figure 1.10 Research Parts and Their Relation to Each Other

The research starts by reviewing databases and concludes that graph databases and a cognitive process can combine to form a cognitive database. Applying the concept of a cognitive database to a supply chain dataset is not a common practice. The graph database structure applies to supply chain data management. Cognitive processes apply to supply chain risk metric analysis. Figure 9 shows the thought process this research went through to build the key concept of the research.

Contribution

- Structuring supply chain data as a graph.
- Using a graph neural network to predict future supply chain values,
- Replacing random weight value with centrality measures in a graph attention network.
- Node-based risk calculation equation.

1.7 Research Overview

Chapter 2 is a literate review of conversion techniques, Bayesian networks, learning algorithms, and graph structures. Chapter 3 details the concept of supply chain data management and how the research will quantify risk. Chapter 4 details the graph structure, centrality algorithms for finding critical nodes, graph neural networks, and a Bayesian network used by the research system. Chapter 5 details the prototype system and how it interacts with the data and the system's basic functions. The prototype system learns the structure of the graph and makes predictions of future values to a supply chain risk management algorithm to showcase the research on current fields of industry. The risk values generated are on a node per node basis. Chapter 6 will propose what future work can expand the system and make it more robust. The research summarizes the details and makes conclusions based on the information in the research.

CHAPTER TWO

Literature Review

The following subchapters will review the current work in database design, artificial intelligence, cognitive informatics, Bayesian networks, and supply chain risk management. The reason for a large grouping of different topics is that this research joins multiple disciplines together to create a unified system.

The foundation for the system is a database, an electronic place to store information. Understanding where the current systems are and where they are heading is key.

As stated in the previous chapter, datasets have become too large for the current relational database. Subchapter 2.1 describes NoSQL as the industries' solution to the data size problem. Migration of data from a relational database to a graph database is another topic.

Subchapter 2.2 will cover artificial intelligence, cognitive informatics, and Bayesian networks as it relates to creating a data structure that is better suited for data mining.

Supply chain risk management research as it relates to using new techniques from the previous fields is part of subchapter 2.3. Subchapter 2.4 is an analysis of the literature review using word clouds, word count and highlighting important articles.

The last subchapter 2.5 will summarize all the findings and explain how these fields together create a novel system.

2.1 Database Design

Traditional designs of relational databases do not manage very large datasets well because one of the most used functions is a joint operation. Joining two tables together can consume a lot of computational resources when those tables have millions of records. To ease this problem, big data companies are designing new database structures. All major companies today collect vast amounts of data. Two of the biggest data collectors are Amazon and Google. Both created their own data storage and retrieval systems, which led others to create data storage systems without RDBs and SQL. The new method is "NoSQL". The method divides NoSQL databases into four major structures: key-value, document, column family, and graph databases.

Key-value databases use hash tables to define the reference links to other tables. Giving Data a unique identifier, a hash, based on data within the data record. Document databases use documents and structure data links similar to a web of information. Column family databases store data in columns of information. Each column has a name, value, and timestamp. Graph databases use graph and network theory. Each node is a data unit, and each edge is a relationship between nodes. NoSQL's main domain is the cloud; hence, spreading stored data out over multiple servers. NoSQL stores vast amounts of unstructured data. For more details, see (Kuznetsov et al., 2014).

Graph Databases have many data models, which include property graphs, hypergraphs, and resource description framework (RDF). Neo4j (Robinson et al., 2015) is one of the popular graph databases. Neo4j is built on a property graph model with a specific query language called Cypher, which is a declarative graph language. When

using Cypher, a user declares what he needs the graph to accomplish. Neo4j graphs are built with nodes and directed edges with properties attached to both. Hyper-graphDB is exactly what the name implies a hypergraph database model. Hypergraphs are made of vertices and edges, but the edges can point to multiple vertices. Hyper-graphDB is the foundation for OpenCog (Goertzel, 2009), an artificial intelligence engine for natural intelligent systems. RDF was created by World Wide Web Consortium (W3C) as a model of how websites are linked together. RDF uses a semantic query language based on subject-predicate-object, conjunctions, disjunctions, and optional patterns or SPARQL (SPARQL Protocol and RDF Query Language). Graph databases do not have a common language, unlike RDBs. A graph database is key to this research since modeling a supply chain network as a graph allows new analysis. Migrating the data from a relational database to a graph database is a requirement for the research. Others are researching this field, but most are specific to a method or technology. The key technology used is Resource Description Framework (Miller, 1998) (RDF), a linked data structure created by the World Wide Web Consortium (W3C) to link data over the internet. W3C defines a program language based on SQL called SPARQL (Fisher et al., 2011) for data creation, manipulation, and retrieval. Mapping a relational database to the resource description framework schema is a field of research. RDB2RDF (Thuy et al., 2014) details a method to map an RDB schema and data in its entirety to an RDF system. The research shows that the system could retrieve all data using SPARQL methods and the conversion process is automatic. D2R MAP (Bizer, 2003) is a mapping language that creates a one-to-one map between relational data tables and RDF classes. Linking instances of data to a

class map for retrieval. The resulting outcome is an abstraction layer that can use SPARQL to retrieve data for an RDB. RDFs can use Web Ontology Language (W3C, 2012) (OWL) for more complex structures. D2RQ (Bizer et al., 2004) creates a virtual RDF with OWL schema to access RDB data. D2RQ is a virtual in-memory map abstraction layer for existing RDBs to expose the data to the semantic web. W3C has published two guides (Sequeda et al., 2012) for accessing RDBs through the semantic web. Research on refinement and building semantic data from RDBs with prior knowledge is another field. Mohsen Taheriyan et al., (2016) show how external Resource Description Framework databases can help to convert a relational database into an RDF database. RDF databases are a triplestore data structure meaning data must be stored and retrieved using a subject-predicate-object triple. RDFs are a specific style of graph structure. One article with similar research is Converting Relational to Graph Databases (De Virgilo et al., 2013). The article details a way to convert a relational database to a general graph structure which could be implemented in either Neo4j or OrientDB. The authors apply the basic naïve approach but refine the graph structure using query data and domain knowledge. The produced graphs are not semantic in structure. The authors did a follow up article (De Virgilo et al., 2014) detailing a tool for the automatic conversion of an RDB to a graph but the R2G tool was not released for general evaluation. A modeling architecture details are written by (De Virgilo et al., 2014) by the same authors as the previous two articles. One key feature of a graph database is its ability to show a semantic structure to the data stored within the vertices and edges. RDFs use a subject-predicate-object semantic structure which can limit the interaction with the data stored.

RDF graphs do not have a built-in structure for the vertices and edges. Labeled Property Graphs do have a basic vertex and edge structure. The semantic structure defined by OWL and RDF is simple and may not be able to handle more complex concept structures. Rajesh Bordawekar et al. (2017, 2018) explores the idea of merging relational databases and cognitive functions. In the article, the authors use the idea of turning data artifacts into feature vectors. Data artifacts can be images, documents, data blobs, or values. Feature vectors are a one-dimensional numerical set of values that describes a data artifact. An artificial intelligence algorithm can compare feature vectors to get an understanding of the data they represent. The author of this article (Hantula, 2016) gives the advantages and disadvantages of graph database vs relational database. The research has drawn many of the same conclusions. These reasons are why this research is using a property graph structure for storing the supply chain network data model.

2.2 Artificial Intelligence

Using an artificial intelligence system on a database requires a data modeling process to prepare the data in a graph style. The research uses a semantic network model as its data model. The reason for choosing a semantic network model is that previous research shows its close relation to human cognition. Charles S. Peirce first proposed existential graphs around 1909 as a new form of logic. Existential graphs laid the groundwork for semantic networks invented by Richard Hook Richens. Richard H. Richens used semantic networks as algebraic interlingua (Richens, 1956) for machine translation of natural language (Richens, 1956). Paving the way for Allan M. Collins, M. Ross Quillian, and Elizabeth F. Loftus's groundbreaking research in cognitive science

and artificial intelligence (Collins et al., 1969, 1970, 1975, see also Quillian et al., 1967, 1969, Saumier et al., 2002). Semantic networks help to describe ontologies in information science and declarative knowledge in cognitive science. This research tried to apply residential grammar as detailed in Language and Man (Binkert, 2004) to refine the semantic structure of a general-purpose graph database. The dynamic structure of a graph database can apply a cognitive graph model, creating a decision model from existing data. Dr. David Danks details many cognitive models as graph structures in his book Unifying the Mind (Danks, 2014) which is a concept the research implements. Graphs are a fundamental data structure used in a good deal of artificial intelligence research (Minsky, 1974, see also Schank et al., 2013) and deep learning machine algorithms.

A graph database is the closes data structure to the model used in artificial intelligence and it only requires a single preprocessing step. These steps will increase the speed of data retrieval from a supply chain dataset. The type of learning algorithm that works best with graph structure data is a graph neural network.

Graph neural networks focus on two domains of structure. The spatial Domain focuses on learning the graph structure. How to encode the graph structure without losing information. Graph structures can be difficult to encode because there is an order to place the nodes within a graph. The key to understanding a graph structure is to incorporate neighbor information. Linear propagation (Gilmer et al., 2017, see also Veličković et al., 2017) builds a weight for each first order neighbor node. Directed edges tell which nodes need to aggregate neighbor weight information. The learning functions better when the data is normalized (Johnson et al., 2007) but the algorithm does not handle self-loops.

Graph convolution network (Kipf et al., 2016) adds a self-loop weight to the algorithm and renormalization of current node data plus its neighbors' data. The final encoding is a sum of the current node and neighbor average. GraphSAGE (Atwood et al., 2016) improves GCN by replacing symmetric normalization with a random walk normalization. In a complex graph structure, first order neighbors may not give enough information to encode a graph structure. High-order neighbor algorithms look to fix that problem (Defferrard et al., 2016, see also Wu et al., 2019, Tong et al., 2015, Grover et al., 2016). The focus of high-order node algorithms is to avoid over-smoothing since adding more neighbors to a node can smooth out local information. High order algorithms are useful in representing complex signals (Gasteiger et al., 2018) when a flexible order is possible with proper constraints. Many graph neural networks focus on a forward propagation of labels or features within the graph. Rational propagation [Wu et al, 2012, see also Li et al., 2019, Loukas et al., 2015, Isufi et al., 2016, Levie et al., 2018) also focuses on the reverse direction propagation to mitigate over-smoothing. The concept applies a rational function to the node's adjacency matrix. The domain of spectral GNN is to study the relationships between polynomials, eigenvalues, and eigenvectors of the graph matrix (Chen et al., 2018, see also Bianchi et al., 2021). The research does not use spectral information since the main idea of the research is to view a supply chain as a graph. Scaling graph neural networks to handle larger graph structures requires a sample method (Perozzi et al., 2014). The research's prototype supply chain graph is small, so sampling is not an issue.

Reviewing the current state-of-the-art graph neural networks did not reveal an extensive set of research papers with a focus on a supply chain. Structured sequence modeling with graph convolutional recurrent networks (Seo et al., 2018), the authors build a graph neural network model using a convolutional neural network model. The authors encode a single graph as a one-dimensional vector using a convolutional neural network. The model compares the graph vectors for similarities and groups them. The research follows a different model for encoding a graph, but the idea of viewing each graph as a time snapshot is the same. GC-LSTM: Graph convolution embedded LSTM for dynamic link prediction (Chen et al., 2022), the authors are using a convolutional neural network and an LSTM. The convolutional neural network encodes the graph like the previous article. The authors' model uses an LSTM to observe the one-dimensional vectors for patterns to help predict which nodes should have links. The research is not trying to predict links between nodes. The research is using the idea of an LSTM to learn important details of the graphs over a period. In transfer Graph Neural Networks for Pandemic Forecasting (Panagopoulos et al., 2021), the authors are using a message passing neural network for encoding the graph structure, then an LSTM to learn how the virus is spreading to different regions. The research uses the article's model but applies it to a supply chain network. Using a message passing neural network on a supply chain network works very well for encoding, since a supply chain network holds dependent data. Message passing neural networks are good at capturing the dependence and positional nature of data. The research knows the data is also time dependent. LSTMs are useful in understanding the dependence within a time series, which is the type of data

found within a supply chain. The research, like the authors, assumes at an MPNN+LSTM model will understand data dependencies and data spreading over time. The research uses a different dataset than the authors, and the read function needs to be changed to accept the dataset. The research believes the concepts in the article are like the goals of the research, which is why it is the basis for the research's neural network model. Fast graph representation learning with PyTorch Geometric (Fey et al., 2019) is a python library that allows this research to build, train, validate, and evaluate quickly without having to hand build basic neural networks. The library contains MPNN and LSTM models. PyTorch Geometric has tools for the in-memory graph structure and the batching process for faster learning. The research uses this library heavily so it can develop the experiment. Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models (Rozemberczki et al., 2021) is a graph temporal library built on top of PyTorch Geometric. This research believes the library supports the models that this research is looking to implement. PyTorch Geometric Temporal, the library name for (Rozemberczki et al., 2021), is in its early stages. The research shows that the MPNN+LSTM and other models are usable on supply chain networks with an acceptable level of error.

2.3 Supply Chain Risk Management

The concept of using network theory to evaluate supply chains is like the article "Mapping supply chain risk by network analysis of product platforms" (Nuss et al., 2016). The authors' research shows how supply chains are like networks and network theory can solve supply chain issues. Their research states that supply chains, like networks, represent the flow of information (products) from one node (location) to

another. The authors apply an in-degree calculation to show the complexity of a node. Out-degree of a node tells a node's diversity, and how it touches many other nodes in the network. Betweenness centrality to identify important nodes that could disrupt or bottleneck the supply chain. They assign risk based on environmental factors of an entity's location. Each node uses a simple averaging function on incoming risk to calculate its risk value. The authors state that the supply chain is a static snapshot in time. This research uses a graph database allowing analysis to happen at different time intervals. This research uses a Bayesian network for risk evaluation at each node, giving a better estimate.

Another group of authors explains a Bayesian network as a tool for understanding risk within a supply chain in Bayesian network modeling for supply chain risk propagation (Ojha et al., 2018). They use the network concept of a holistic measure for rating the reliability of a supply chain. The research learns the Bayesian network from the data provided using a K2 closes neighbors algorithm. The analysis adds a risk factor to each node, then calculates new risk values using Bayes' theorem. This research will use a Bayesian network as well, but it will build the calculation as a process within a graph database. The Bayesian network built within a graph database will update risk evaluation when a change has happened within the network. Learning a Bayesian network is not part of this research, since the supply chain data already has network data embedded within it. An extended Bayesian network approach for analyzing supply chain disruptions (Soberanis, 2010) details how a Bayesian network can help understand supply chain disruption. It details the novel concepts of how Bayes theory and Bayesian networks can

rate reliability, disreputability, and node failures. This research's methodologies create probability tables that help to decide on tasks and understand supply chain robustness. The author talks about how the methodologies need to apply to production level supply chains and how the calculation should run in real time. This research shows a way to apply the above methods to a graph database with real-time calculations.

Supply Chain Risk Management (SCRM) as detailed in (Wieland et al., 2012). Details of the different types of risks a supply chain can face and give methods for quantifying that risk. They detail mitigation methods in their research, such as redundant suppliers and manufacturing centers. Time to recover is a very important measure in supply chain management. This research can help to identify what supply chain entities need redundance, how risk stacks up in a supply chain, and how time to recovery of one entity can affect another.

The authors of a graph theory-based methodology for vulnerability assessment of supply chains using the life cycle inventory database Nakatani et al., 2018). Create graph structures and adjacency matrices to evaluate the vulnerabilities within a supply chain. This research uses a graph database that gives structure and does not require specific methods, such as building adjacency matrices since the graph database itself has this information built into the system.

Many articles look at how to do data mining (Chen et al., 2012) on supply chains. Kara et al. (2020) create a supply chain management framework that uses data mining to extract risk values. The authors realize that supply chains are creating a huge amount of data. Data is overwhelming organizations, and they need a method to process that data in

real time. The authors' framework identifies risk metrics, stores these values in a data warehouse, then data mine the warehouse for key values. Identifying the risk values requires multiple steps. The framework questions the organization about its supply chain size, its attitude towards risk, and the resilience value of the supply chain. Mapping out the supply chain visually is another task to better understand the topology. The authors use a heat map to visualize the supply chain. The research visualizes a supply chain as a graph and stores the data plus risk values. Combining multiple steps within a graph database allows for advanced data mining and visualization techniques.

In the paper (Tan et al., 2013, see also Fearnhead et al., 2019), the authors focus on the problem of using traditional quantitative and qualitative methods on big data. The authors review statistical methods that are traditional to supply chain analysis. The conclusion from the paper is big data can create a data noise that will skew results, big data can smooth out abnormal data points, homogeneity when using statistical methods, and multivariable/high dimensionality datasets can be difficult for a statistical method to evaluate. The paper states that researchers are making misleading conclusions because of statistical results from big data. Big data can lead statistical methods to correlate the wrong variables or infer the wrong patterns within the dataset. The issues found by the authors require statistical models to change their methods for finding results. The research uses a graph neural network to analyze big data and not a statistical method.

Current simulation models have a difficult time processing big data (Sanyal et al., 2013). Big data overwhelms cause-and-effect simulation to the point where they are not practical. Complex systems are hard to model within a simulation model. The number of

variables can confuse the models and produce inaccurate results. The size of the datasets creates a large computational load on the computer systems running the models. The computational load increases the time and cost of executing the simulation. The research uses modern graph neural networks as a simulation model. Graph neural networks train faster than traditional models and graph neural networks can pre-train and save training data, allowing for faster updates when new data is available.

The authors of this paper (Blackhurst et al., 2018) use a Petri net and Triangularization Clustering Algorithm to find vulnerabilities within a supply chain. Using the results, supply chain managers can create migration strategies and pinpoint disruptions. Petri net is a mathematical modeling language that describes distributed systems. A Patri net represents a directed bipartite graph with two elements, places, and transitions. The Patri net can activate and transfer tokens through the network. The network itself is the representation of a mathematical function. The paper explains a method for clustering the nodes into different groups. The grouping can help a supply chain manager understand the vulnerability area of the network. The activation function can describe disruption in a supply chain. The research uses a more general graph structure that can represent similar elements to a Petri net. The research automates building a graph by learning the data and translating a relational database.

The authors of (Husna et al., 2021) implement a demand forecasting algorithm using three different neural networks. The three neural networks are a standard artificial neural network, a convolutional neural network, and a long-short term neural network. The authors compare mean square error values. The problem with using these neural

networks is that algorithms lose the understanding of dependent data, which is very important to a supply chain. Graph neural network considers the network structure and data dependence when learning the graph. The research's algorithm shows a lower mean square error than the authors' algorithms. The predictions are closer to the ground truth than signal neural network algorithms.

Optimization problems are a field of supply chain management that looks to find the optimal solution. The authors (Abbasi et al., 2020) use machine learning to find an optimal solution to a demand problem. The paper compares regression tree (CART), k-nearest neighbors (k-NN), random forest (RF), and multilayer perception (MLP) artificial neural network (ANN) to see which of these algorithms can better learn the relationship between input parameters and decision values. The authors train each of the models with a few events to find an optimal demand and distribution model. Neural networks show a lot of promise in being able to select an optimal solution to the problem. The authors state that large optimization models require more data than neural networks and take longer to train. One of the problems that the author found with the neural network is that retraining requires a whole dataset. The research neural network can pre-train and save data. A graph neural network can retrain with a delta dataset, making updates quicker. The research shows that a graph neural network can produce better results for graph structure problems than single neural network algorithms similar to the one the authors use in their paper.

Creating a value of resilience for a supply chain is an important field of study. A resilience value tells an organization how well an entity within a supply chain can resist

disruption. The paper (Chen et al., 2020) builds a resilience value using standard resilience methods that consider structure and environmental factor plus a time value. The time value is a measure of how long a customer will wait for a product. The resilience value changes the cost value of the product. Understanding the amount of loss an organization will experience from a disruption is useful. Raising the resilience value mitigates the loss and decreases supply chain interruption. Interruption in a supply chain is a major issue since restarting entities can take days, which adds to delay in filling orders. Customers will only wait so long before moving to a different product or vendor. The authors' resilience value creates a relationship between capital cost ability to reduce order loss. Order loss translates into profit loss for an organization. Analyzing the resilience value using standard methods will show the relationship that needs strengthening to mitigate a profit loss. The formula views the supply chain and uses many variables to represent the different factors in a supply chain that affect resilience. The changes made to the formula create a general method to reduce loss. For example, increasing agility and dual scouring will increase the resilience value and lower the loss value. The problem here is what parts should have a dual source. What does supply chain agility really mean in an entity per entity structure. The research shows that a finer look at the entities that are part of the supply chain will give a better resilience picture than variables and formulas.

The paper (Silva et al., 2017) shows how a basic perception neural network can predict stock levels that will minimize interruption. The authors use two experiments, first they look to predict future stock levels so the supply chain can run without

interruption. Second, they look at a restock scenario, where the perception neural network predicts how to increase stock levels at the retail entity. The first experiment shows significant results with a 98% prediction accuracy. The second experiment shows 97% accuracy. The results were faster and more accurate than simulation models. The authors comment that the supply chain in the experiments is not complex. The research understands that a more complex supply chain requires a graph neural network. A real-world supply chain has multiple dependencies and is not linear.

2.4 Analysis of Literature Review

The analysis of the literature review starts with a visual viewing of the main topic. The visual representation is in the form of a series of world clouds.

Figure 2.1 – Word Cloud of Supply Chain Risk Management Topics

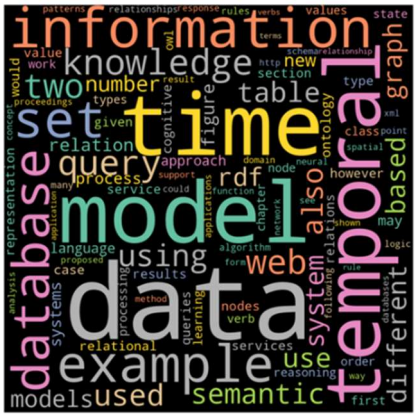

Figure 2.2 – Word Cloud of Database Design Topics

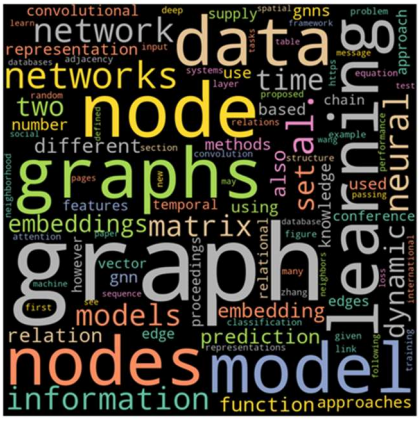

Figure 2.3 – Word Cloud of Artificial Intelligence Topics

Database Design		Supply Chain Risk Management		Artificial Intelligence	
keyword	Count	Keyword	Count	keyword	Count
data	5211	risk	2367	graph	2344
time	3355	supply	2216	node	992
model	3241	chain	1730	graphs	841
temporal	2690	network	994	model	826
information	2147	risks	824	learning	797
example	1880	management	682	data	784
set	1853	research	549	nodes	781
database	1826	model	529	networks	706

Table 2.1 – Word Count of Literature Review Topics

Reference	Title	Difference
Panagopoulos et al., 2021	Transfer graph neural networks for pandemic forecasting	Similar to the authors this research uses GNN to forecast. Unlike the authors this research uses a GAT+LSTM.
Bordawekar et al., 2017	Cognitive database: A step towards endowing relational databases with artificial intelligence capabilities	Using an GNN and a graph database for a cognitive database
Xu et al., 2018	How powerful are graph neural networks?	GAT and LSTM are not cover by this paper but gave this research the idea
Husna et al., 2021	Demand Forecasting in Supply Chain Management Using Different Deep Learning Methods	The authors use different deep learning models separately, this research uses GNN
Li et al., 2022	CAGAT: centrality-adjusted graph attention network for active scientific talent discovery	Unlike the authors this research uses Katz centrality for the GAT
Ebert-Uphoff, 2007	Measuring connection strengths and link strengths in discrete Bayesian networks	The influence measure is derived from this paper
Kosasih et al., 2022	Towards knowledge graph reasoning for supply chain risk management using graph neural networks	The authors focus on link prediction not forecasting

Table 2.2 – Important Articles Used by This Research

Topic	Gap in Research	Trending Research
Data Design	How to structure interconnected data	Scaling to large datasets
Artificial Intelligence	Centrality measures within Graph Neural Networks	Graph Neural Networks
Supply Chain Risk Management	Graph Neural Networks for Supply Chain Risk Management	Network styles analysis
	Granular Risk Calculations	

Table 2.3 – Gap and Trending Research within the Topics

The word clouds in Figures 2.1, 2.2, and 2.3 show that there are related topics between the three areas of the literature review. The word count table 2.1 shows the high-count topics that do spread across the areas. Table 2.2 highlights some of the important articles that help to inspire this research. Table 2.3 details gaps and trends in the literature review topic. Table 2.3 shows that this research's focus areas line up with the trends in the literature review.

2.5 Summary

The research reviews database structures and architecture to better understand the type of storage system cognitive databases needs. The literature shows that a relational database will not support the dynamic and relationship heavy structure of a supply chain. The research concludes that a graph database is the best solution to the storage requirements.

A cognitive database is only cognitive when it uses cognitive functions in the data stores. Preparing the data structure requires organizing the data within a cognitive frame. The research builds the cognitive framework around semantic meaning and directed

graph structure. The graph structure can represent cognitive processes and hold the feature values for calculations. The research reviews cognitive research to show that others have similar concepts of using graphs.

Learning the graph requires a computer system to use graph neural networks. Graph neural networks can encode the graph structure into vectors that are usable by deep learning models. The deep learning model shows patterns or predicts future values. The research reviews many styles of graph neural networks before choosing MPNN+LSTM. Many graphs neural networks are specific to the task.

To show the usefulness of graph neural networks, the research chose supply chain management. Supply chains are relationship heavy with many interdepend values. Graphs can show the interdependent nature of a supply chain. Graphs also consider the relationships between entities. The research uses synthetic values within the supply chain to generalize the calculations to risk analysis. Risk analysis uses many similar concepts to graph analysis to understand what parameter causes risk to increase and how to mitigate it.

This research is using the output of the graph neural network in a Bayesian network to see a cause-and-effect model. The next subchapter will discuss the problem area that the research is looking to help.

CHAPTER THREE

Supply Chain Data Management and Risk Metrics

Supply chain risk management is the identification, evaluation, qualification, and assessment of risk impact. The definition of risk is a situation involving exposure to danger. The primary purpose of a supply chain is to produce and deliver goods or services to its end customer. Risk to a supply chain can take many forms. Internal, external, supplier, manufacturer, transport, and customer risk can disrupt a supply chain's flow of products or information. Minimizing these risks is the primary purpose of supply chain management. Minimizing the risk can take many forms, such as redundancy, stock levels, or route optimization. Supply chain risk management frameworks help to identify and mitigate the risk of affecting a supply chain.

Industry has developed models to understand the risk and manage its effects. The two dominant styles are qualitative and quantitative. Qualitative models are easy to generate and are usually questionaries, surveys, or cause-and-effect style processes. Quantitative models are value measures which are further broken down into analytical, simulation, and hybrid models. There are hybrid qualitative-quantitative models too. A model can categorize, analyze, and evaluate the risk values of a supply chain or individual entities. The models help a supply chain manager make better decisions.

The result of a qualitative or quantitative model is to identify the risk metrics. Risk metrics can take many forms, depending on the data that generates the risk. Using risk strategy methods, an organization identifies data that data creates a risk metric. Using risk metrics, organizations decide on how to mitigate the risk. Algorithms exact risk metrics from a supply chain database, then the algorithm creates an actual risk value. These are separate systems, but it is possible to combine the system using a graph database.

Graph databases can store the risk values and the risk model. A graph can show a cause-and-effect diagram. Supply chain managers could run analytical and simulation models from a graph database. Risk values can be part of a node, while weights are part of an edge. Complex simulations could learn the structure and risk values of the graph from the database. Bayesian networks can quantify supply chain risk and handle uncertainty. The next subchapter discusses supply chain data management as it relates to risk metrics.

3.1 Supply Chain Risk and Risk Metrics

Supply chains need to deal with many types of supply risks. There are at least five types of supply chain risks. Plan and control risk is risk that comes from process, methods, tools, and back-office risks. The cost of those risks to the organization could be capital, opportunity, or logistic cost. Supply risk can take many forms, from quality issues, supplier failure, damage to product, single sourcing, global sourcing to part supply. The impact on the supply chain could be production stops, production delay, replacement costs, or return costs. Process risk comes from lead time, capacity issues,

quality issues, machine/technological issues, human error, planning error, and third-party logistics issues. Process risk creates difficulties to supply product and repair costs. Demand risk occurs when customers demand changes, customer preferences changes, retailers cancel orders or inflexibility. Demand risk causes supply difficulties and triggers a bullwhip effect. Environmental risk comes from natural disasters, political issues, customs issues, and an outside factor that could affect the process of a supply chain. Opportunity and replacement cost are effects that an organization must deal with when an environmental risk occurs. Supply chain management tries to control all the above risks while minimizing the impact on the supply chain. Many of the risk factors are outside of an organization's control and require mitigation, which comes with a cost benefit analysis. Internal risk is more controllable, but still brings a cost to the organization. The first step with risk analysis is identification and categorization.

Classification of the risk using a morphological analysis, created by Fritz Zwicky (Zwickly 1969), is a non-quantifiable questionnaire that helps to fill out a table. The table lists items such as planned or unplanned, time lost to the disruption, severity, probability of occurrence, cost to organization, time impact, disruption source, response process, level of disruption, and interrupted supply chain flow. Filling out the table gives an organization an idea of how bad a disruption risk is to the organization. Saving the table can be for later reference could be beneficial to an organization. Another form of risk analysis is a risk map. A risk map is a chart with a y-axis of time and x-axis of impact. Management draws an acceptance line across the chart. It does not need to be a straight line. Mapping risks on the chart is the first step, then subject expert categorizes if the risk

as acceptance, assurance, outsource, or activity. Acceptance means the risk rarely occurs, and the impact is small. Assurance is an outside source ensures they can mitigate the risk for the organization. Outsource means getting a third-party vendor to accept the risk and build the item. Activity means taking a course of action to mitigate the risk, such as training or maintenance. The inputs that generate these risks are in figure 3.1.

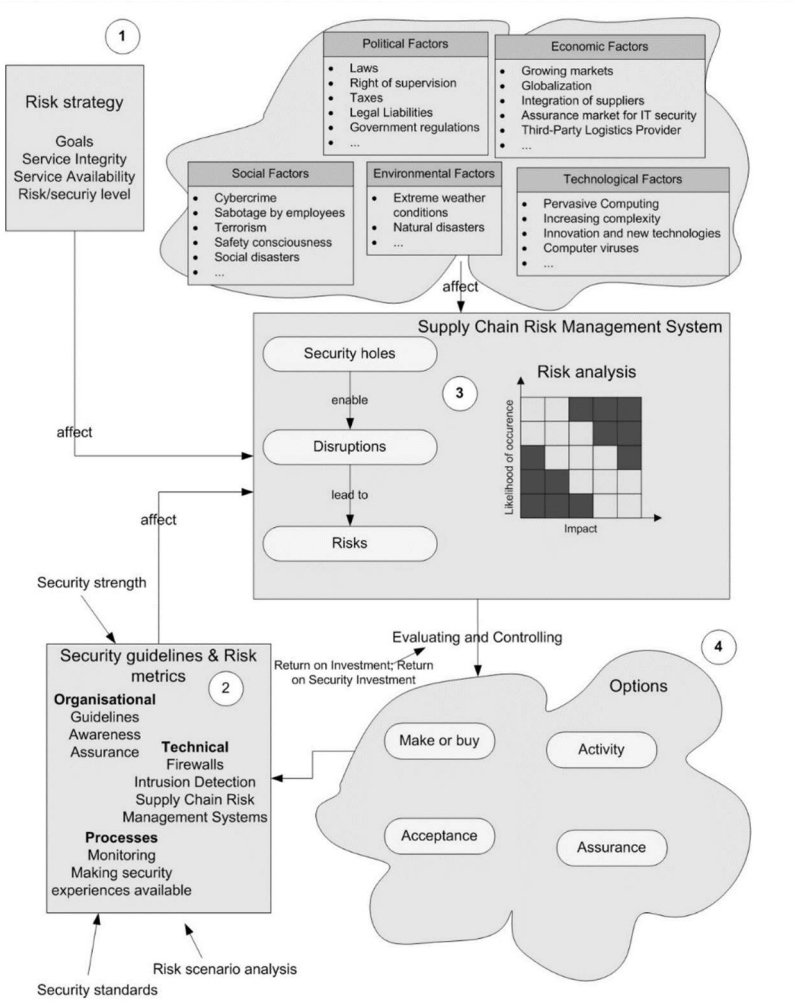

Figure 3.1 Supply Chain Risk Management Framework (Müßig 2006, p. 42)

The first is the strategy the organization uses to integrate and act on the risks. Taking outside factors into account, such as environmental, social, or political. Reviewing security and risk guidance process are part of the process. Fourth, applying the categories above to each risk. Using all the information available better describes the risks and their impact. Disruptions are the primary risk that all supply chains are trying to avoid.

After categorizing and defining a supply chain's risks, an organization can monitor and quantify the risks. Monitoring risk requires identifying what data a supply chain generates those points towards a risk being triggered. Cost, quality, delivery, responsiveness/flexibility, environment, and structure are some of the metrics that are risk triggers. Cost is the cost of goods and service. Cost risk comes from prices being inflexible and not negotiable. Quality risk came from poor design and built products. Responsiveness/flexibility is the measure of how able an organization can adapt to change. The environment is an outside factor like weather. Delivery risk comes from delivery cost issues or delivery fulfillment issues. Structure risk is the process, machines, or technology requirements of a product. Quantifying these risks can be difficult and understanding the entities within a supply chain that generate the data also proposes a similar difficulty. The risk value may need to be a combination of several factors coming together to create the risk trigger. The final risk value can become a function with several inputs and a single risk value. The risk values can be a high-level number that affects the supply chain. This only gives an organization insight into what things can disrupt the

supply chain. Many organizations face partial or temporary disruption, which requires understanding the risk values associated with each supply chain entity.

Assignment of risk values to supply entities creates a network of risk values that are interdependent. Lowering the risk in one area could lower or raise the risk in another area. The advantage of assigning risk to individual entities is that it helps to show a failure point in the network. Another advantage is being able to see which entities cause a cascade of failure through the network. Understanding failure points and why entities fail makes mitigating the failure easier. Mitigating risk can come in many forms. Cost risk may require multi-sourcing of parts or renegotiation of part contracts. Transferring risk to the supplier is an option in some supply chains. When a supplier accepts the risk of not delivering parts to its customer. The customer may receive discounts or reimbursement when parts miss delivery dates or do not satisfy the requirements. A supplier may compensate for one cost, but the customer may have another cost to mitigate. The outcome is to find out which risks are acceptable to the organization. Is the time to recover acceptable to the organization. Time to recover is how long it takes for a supply chain to return to full production. How long will a customer wait for a product to arrive before looking for alternatives? Mitigating this type of risk requires an organization to safety stock its retailers. Safety stock is a buffer of stock that allows a retailer to meet demand without running out. The problem with safety stock is that a retailer or organization may need to deal with leftover products if customer demand changes. Another risk issue that needs to be mitigated is the bull whip effect. The bull whip effect is where issues ripple up the supply chain instead of propagating down the supply chain.

Managing supply chain risk is difficult since it is able to move up and down the supply chain because of many factors. The problem is compound when planning requires a supply chain to understand risk 2 to 5 years ahead of time. Understanding future risk is mitigated by analysis and simulation models. Organizations can model their supply chains with risk factors and other key metrics. The analysis and simulation models run scenarios, giving future targets based on historical and key data. In the next subchapter, the research will review current simulation and analysis model for a supply chain.

3.2 Analysis and Simulation Models

Supply chain analysis models are a pure mathematical model using time series as input to predict future values. Time series data is a historical record of values that are important to the scenario that is being analyzed. The research details the Autoregressive Integrated Moving Average (ARIMA) statistical model that is useful for time series predictions but not as accurate as graph neural networks. Understanding ARIMA requires breaking the model down into its pieces. Autoregressive model understands previous data points affect future data points, creating a correlation. An autoregressive model has two correlations: auto-correlation and partial autocorrelation. Auto-correlation means all the values the model is considering have a correlation. Partial autocorrelation means one or more variables do not have a correlation.

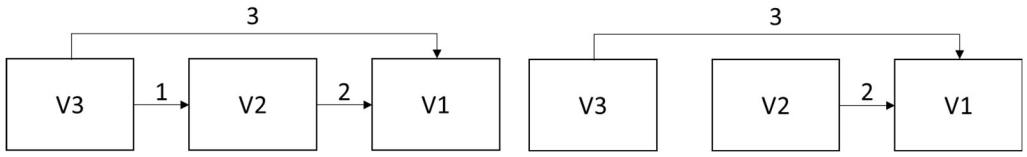

Figure 3.2. Autocorrelation and Partial Autocorrelation

The time series dataset is large, so there will be multiple correlations. A correlation is weight based on its distance from the end data. Recent data carries a higher weight than older data. Autoregressive creates the relationships between the data points. Integrated is the consideration of the difference in raw observations so the model can smooth out stationary trends. Stationary means the mean of the dataset is constant, standard deviation for the dataset is constant, and no seasonality meaning changes in the datasets follow a trend. The last part is the moving average, which uses observation and residual error to create a dependance from a moving average model over the lagged observation. Meaning the model compares the predicted values to the actual value to get an error value that is a representation of lag time. An ARIMA model also has three parameters lag observations in the model, how many times to difference the raw observation, and the size of the average window. Any of these parameters can be zero, which removes that part of the model. ARIMA makes several assumptions about the data. No anomalies are present in the data. Using constant error and parameters for the model. Historic trends have more weight than recent stress periods. The time series has a pattern to its data. Looking at ARIMA In a more formalistic way.

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3.1}$$

Where c is a constant, ε is the error term, φ is the autoregressive model's parameter, and θ is the moving average parameter for the model. ARIMA is a very simplistic model that requires data to follow a trend. Very seasonal data is not useable by

a standard ARIMA model. Very unstable data will not work with an ARIMA model. Large historical trends can dominate smaller trends leading to the model not predicting high risk areas. This was the case with the housing market in 2008. All the models were based on historical data which shows an increase year after year in the real estate market. The instability of the mortgage companies caused a crash in the system which no one predicted. ARIMA shows that prediction models need more than historical data and statistical algorithms to predict correct values of future data points.

Simulation of a supply chain is another way to optimize and minimize risk. Simulation shows an organization how a supply chain will perform at a base level. Tuning parameters, flow, processes, or suppliers can change how the supply chain will function. Supply chain simulation requires a lot of upfront information, such as time to build product, cost to buy, time to transport, and demand. Time is a key factor in supply chain simulation. Each entity has a set time cost, plus there is transport time between entities. The total path gives the best estimate of how long a product will take to reach its end customer. The key parameter can be any factor that the organization wants to study. Virtual modeling of a supply chain in a computer system is the first step. Entities have inputs that have a relationship to the output under consideration. Each entity acts as a function, passing its output to the next entity in the chain. The concepts are part of network and flow theory. The end of the supply chain has the outcome. Simulation evaluates the best outcomes when all the parameters are optimal, showing the organization of a sunny day scenario. The organization can go in and change parameters by hand, creating rainy day scenarios that they can foresee. Another path is to run a

simulation model, such as Monte Carlo, to inject parameter change on a larger scale. The reason for using a Monte Carlo simulation is that a supply chain has a high level of uncertainty in the model. Monte Carlo models are suitable for showing risks in a supply chain. A Monte Carlo assigns multiple variables as uncertain. This way, the simulation will generate multiple outcomes. Averaging the outcomes creates an estimate of the result. The simulation uses a random variable, within a random pool, repeatedly on the uncertain metrics. Each time a variable change, the model runs to produce an outcome. Averaging a set number of outcomes produces a trend that fits a probability distribution. To better explain, let's take cost to produce an item. Using historical data, a formula can generate a periodic daily cost by dividing today's cost by yesterday's cost.

$$Periodic\ Daily\ Cost = \frac{Today's\ Cost}{Yesterday's\ Cost} \tag{3.2}$$

Generating a series of cost data points. The model will average the series, get a variance of the series, and a standard deviation of the series. These values are used to calculate a drift and a random variable. Drift is the average minus the variance divided by two. The model generates a random value by multiplying the standard deviation by a true random value.

$$Drift = Average\ Daily\ Return - \frac{Variance}{2} \tag{3.3}$$

$$Random\ Value = \sigma \times True\ Random\ Number \tag{3.4}$$

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} \tag{3.5}$$

Now the model can predict the next day's cost by multiplying today's cost by e, Euler's number, raised to the drift plus random value. Repeating this equation creates a set of numbers over time, which creates a series of outcomes.

$$Next\ Day's\ Price = Today's\ Price\ \times\ e^{(Drift\ +\ Random\ Value)} \tag{3.6}$$

Mapping the outcomes to a standard model creates a probability disruption. In the example, the probability disruption follows a bell shape disruption, giving an average probability that the organization will need to pay a specific cost to produce its product. The problem with using analysis and simulation models is that the larger the dataset, the more time it takes to produce a result. Another problem is analysis requires using a whole dataset to get accurate results. Simulation models may remember previous simulation results, but simulations still use only historical data. Big data creates another problem for analysis and simulation models, being able to select the correct values. Selecting the correct values for simulation and analysis models is key otherwise, the results are not useful. Finding correlation or key metric can be difficult with large databases and large number of entities. Using graph neural networks, a cognitive database could learn the key metrics and find the correlations. After finding these values, a cognitive database can correctly predict future results.

3.3 Graph Neural Network and Graph Database for Risk Analysis

In the previous subchapters, the research reviews the strategy of risk analysis, how a model can quantify and predict risk values. The problem comes with how to store the data for all the analysis. Traditional relational databases store the time series information for use by the risk analysis models. The models run the data through the algorithms and predict results. The relational database stores historical data. Historical data creates the problem of larger trends overshadowing small spikes in data. Another problem is data exaction and model execution can take a long time. A solution to the previous problem is to store the model in a graph database. Storing all the variables and the relationships in a graph database allows the equation to be run directly on the database. Generating predictions using the database equations can go back into the database for future analysis.

Another area where a graph database can help risk analysis is understanding how entities are connecting to each other. Connecting entities together creates a network that helps to define the risk strategies an organization should implement. Graph databases can model decision trees with actual real-time data. When the data values pass a threshold value, the decision tree can trigger a new mitigation strategy in real-time.
Testing for correlations between values or entities in a graph database is simpler. Creating a correlation relationship between two values can place those values into an analysis model. During the next analysis cycle. The model will analyze those values along with all the other values. Reviewing the outcomes will show if the values have a correlation. Finding correlations by grouping or classifying graph data is possible because

related values group together. Graph neural networks group and classify nodes or edges or a graph, a graph object which holds data values, by turning that node into a one-dimensional vector. The vector holds the encoded information for the graph object. Comparing a graph object's vector to another graph object will give a similarity value. A similarity value closer to zero means the vectors are similar, while a similarity value closer to one means the objects are dissimilar. Similar objects group together, meaning there is a correlation between the values.

Graph neural networks can run standard simulation models, prediction models on non-stationary data, and analyze a network at different levels. A graph neural network starts with a graph structure that is like a simulation or network model. Linking Historical data to each node and edge creates the data point for analysis. The graph neural network aggregates the data at every node. Learning the data trends for a particular node. The graph neural network could predict values for the node, but the error rate could be high without dependency data. Using a graph neural network with a message passing structure. Neighbors of a node send data to a node from inclusion in an aggregation. Predictions for each node are now more accurate using all the influencer data. Building a future graph from the node prediction is useful for graph level analysis. Using the node information from the aggregation layer for input data into a memory neural network. A long-short-term memory neural network can learn the trends between each graph. If each graph represents a snapshot in time of a supply chain. Generating snapshots can happen at the end of a production day. The snapshots are graphs are a vector encoding of the supply chain, so they do not take up a sizeable amount of storage space. The system adds the

vectors to the learning repository for future training. The graph neural network can understand trends at a node and graph level by combining neural models.

Graph neural networks can learn a graph structure as a complete entity. Leaning the information helps to compare the separate graph against each other. For example, a supply chain could want to run a route analysis within a delivery area. Each route becomes a graph with many feature values. A graph neural network will encode each route into a data vector. A quick compare of all the data vectors shows the best routes. Finding abnormalities in a supply chain by finding short-term patterns in a node or graph or subgraph is possible. Graph neural network can be used when a graph object is not in a steady state. A steady state is the typical behavior or state that a graph object displays. Detecting abnormalities in data allows a graph neural network to understand patterns like seasonality or disruption.

Graph neural networks can learn the stable form of a supply chain. Learning the stable form over time helps the graph neural network understand small fluctuations in the network. When an abnormality happens, the graph neural network can activate a risk strategy. The abnormality can be at any level since the model holds information of a complete supply chain network. Decoding the abnormality will show exactly which entities are having issues within the supply chain. Abnormal entities can trigger decision trees or bring alternative entities online. Again, adding the data to the existing model can help the system learn which indicator led to the abnormality.

A graph neural network model is like a monitoring system for a supply chain. A network of risk metrics, correlations, and the concept of a stable state these data points

trigger activities. The activities can activate risk strategies, adjust timeline, update cost analysis, or warn an organization of abnormalities happening within a supply chain. Graph neural networks can create a supply chain that self-monitors and decides when to take action to mitigate disruptions and low the overall risk levels of the supply chain. Resilience is another key factor supply chain systems are trying to maintain. Resilience is a supply chain's ability to adapt to change without decreasing its flow of product. Product flow is the primary goal of a supply chain. Keeping the flow rate high insurances that the supply chain is delivering products on time. Any disruption in the supply chain could decrease the flow rate causing a disruption in product production. When production slows, profits could go down. Graph neural networks can scan through multiple network architectures, finding the ones with the high flow. These alternative architectures can activate when a disruption occurs.

3.4 Summary

Supply chains risk analysis can take many shapes, from paper questionaries to decision trees to quantitative analysis model to simulation models. Using historical or simulated data, these methods try to predict future supply chain issues. Organizations assign risk values to these future issues. Understanding the risk issues allows an organization to evaluate if transferring or outsourcing the risk is possible. Otherwise, the organization may try to ensure the risk or accept it.

Making the right decision requires the organization to gather data to support each strategy. The data gathering activities are part of the analysis or simulation activities. The outcomes of the analysis or simulation results help an organization understand which

strategies to follow. The problem with simulations and analysis is that a holistic approach may not be possible. Giving the organization a partial picture of future risk. Another issue is that these activities are usually during design or retrospective time periods, not during actual production time. Preliminary or post work is sometimes not the primary focus of a supply chain. Big data has its own problems, large datasets are hard to analyze or simulate. A large dataset has many variables, so feature selection, selecting data value for analysis, can be difficult to accomplish.

Graph databases and graph neural networks have the ability to be used at all stages of the supply chain process. Graph databases store the data values and metric in a manner that is more acceptable to the risk models. Graph databases used in production can run simple or complex analysis and simulation models on real-time data. A robust solution is a graph neural network that understands the stable state of a supply chain and can react to abnormalities. Graph neural networks can run alongside a supply chain monitoring the risk level and other metrics. The graph neural network can save-off data to a graph database for later training activates. A graph neural network can understand the nodes or graph level abnormalities happen within a supply chain. Graph neural networks can combine and replace analysis and simulation models in a supply chain system. Graph neural networks are robust enough to run alongside a supply chain and monitor the situation for predetermine scenarios. Graph neural networks can provide more value than traditional systems.

CHAPTER FOUR

Methodology

In this chapter, the research covers methods it uses to build a risk value per node, supply chain entity. Subchapter 4.1 introduces the basic concept of risk within a supply chain. The subchapter details the standard risk calculation and this research's extended version. A brief explanation of how supply chains are similar to a graph. Subchapter 4.2 summarizes each part of the research's expanded risk calculation formula. Highlighting each of the four data parameters. Subchapter 4.3 discusses graph databases and how they help the research build its data structure. The graph data structure helps the research's algorithm forecast and generates data. Subchapter 4.4 discusses how ARIMA is used to generate the base failure probability. The research inputs the raw time series data into an ARIMA function to predict future node base failure values. Subchapter 4.5 discusses how this research can train a graph neural network model for predicting future values of retail nodes. The retail nodes represent the impact on the business. Subchapter 4.6 Between centrality algorithms, identify important nodes within the graph structure. Based on the results of the centrality algorithms, the base probability values of each node will increase or decrease. Subchapter 4.7 shows how the graph structure will support the construction of a Bayesian network. The Bayesian network distributed probability model will show the influence nodes have on each other. Subchapter 4.8 details the research's formula for calculating risk. Subchapter 4.9 summaries all the previous subchapters.

The next subchapter introduces the concept of using a graph to represent a supply chain. How the supply chain graph allows this research to use graph neural networks, Bayesian networks, and centrality to highlight new data. The subchapter discusses risk, how to measure risk, and why it is important to a supply chain.

4.1 Supply Chain Introduction

A supply chain is a network of suppliers, manufacturers, distributors, and retail stores that work together to deliver a product or service to a customer. A network is a group of interconnected items that can communicate and pass objects to each other. The mathematical definition of a graph is a set of nodes and edges. Nodes are a point where edges meet, and edges are a connection between two or more nodes with or without a direction. The similarity to networks is apparent both have nodes and edges represent connections. A graph is related to a supply chain network as follows, supplier, manufacturers, distributors, and retails stores, supply chain entity, are the node. Connections, relationships, or routes between a supply chain entity are edges. Formally stated, G = (n, e) where n is a node and e is an edge, S = (se, r) where se is a supply chain entity and r is a relationship, se = n, r = e, hence S = G.

Figure 4.1 Supply Chain to Graph

Nodes are suppliers, manufacturers, distributors, and retailers. The edges between nodes represent the connection and flow of products or information through a supply chain. Figure 4.1 is a visual representation of the similarities between a supply chain and a graph. Storing a supply chain dataset as a graph allows the use of graph neural networks and network theory algorithms. These algorithms give an insight into the data that was not possible before the convention to a graph structure. (Nakatani et al., 2018, see also Aziz et al., 2021, Wagner et al., 2010, Tan et al., 2019) all share a similar concept to this research.

Risk is the probability of an unfavorable outcome. To illustrate the idea, take a pair of dice and roll them. Two is an unfavorable outcome, and the risk of rolling a two is .16%. Risk is not uncertainty. Uncertainty is not knowing what will happen, and the probability of something happening is not quantifiable. Knowing the probability of a risk occurring allows us a chance of avoiding the risk. Uncertainty is not avoidable. Another way to differentiate between risk and uncertainty is that risk is insurable while uncertainty is not. Measuring risk in a supply chain network is very important since decisions have large effects on a company's profit. Supply chain decisions need to be made ahead of time because changes take time to implement and every day without a product is a loss in profit. Selecting supply chain entities happens years or months ahead of time. Understanding risk allows a company to minimize loss and maximize its profit.

$Rt = Pt * Lt$ is the standard risk value formula. R is the risk value, P is the probability of failure, and L is the impact on the business value, while t is the time of the

impact event. Failure in a supply chain can be many factors, but they are quantifiable. A company knows how reliable its manufacturers and suppliers are by using historical data. Estimating a distributor's delivery times happens by calculating the route time, then factoring in all delay factors. Forecasting retail sales is based on demand and other factors. Impact on a business can be many things. Loss of profit is the main impact, but equipment breakdown has a cost to repair and a startup time. Disruptions to the flow of a supply chain can affect a retail store profit but not affect the company's profit. Distribution delays can lead to a loss in brand loyalty. Both factors create a risk that the company wants to avoid since it will affect its ability to create a product that is in demand and profitable.

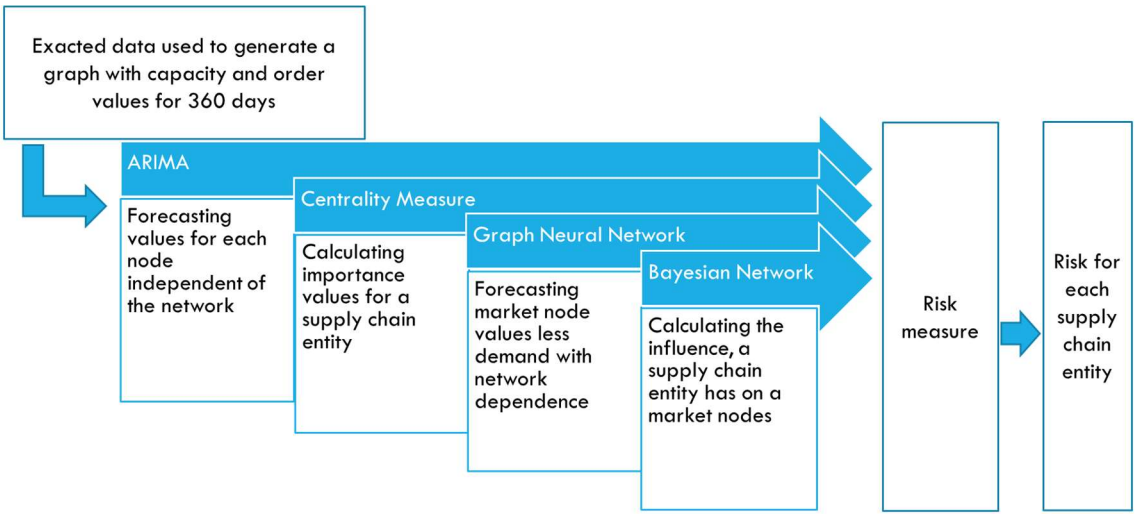

Figure 4.2 – Full Process for Risk Management Calculation

Figure 4.2 is the complete process that this research uses to calculate the final per node risk calculation. Each one of the stages will be explained in the next subchapters.

In the next subchapter, this research will discuss the proposed risk measures. This research uses a calculation to quantify the risk of a supply chain entity failing to complete its task.

4.2. Proposed Supply Chain Risk Measures

This research expands on the standard risk equation by breaking down P and L into separate node values. Creating a granularity to the standard risk formula that considers how each node affects the supply chain. The Research' s modified risk value per node formula starts with the standard risk equation 4.2 but taken for node, n, at time t.

$$R_{nt} = P_{nt} * L_{nt} \tag{4.3}$$

P_{nt} represents two values for node, n, at time t, A_{nt} and B_{nt}. A_{nt} is ARIMA prediction of node, n, failure at time t. B_{nt} is the betweenness value of a node, n, at time t. Formal shown in equation 4.4 is the probability of failure for a node.

$$P_{nt} = A_{nt} * B_{nt} \tag{4.4}$$

$$\mathrm{L_{nt}} = \mathrm{M_{nt}} * \mathrm{I_{nt}} \tag{4.5}$$

L_{nt} (4.5) also represents two values for loss impact at node, n, at time t. M_{nt} is the predicted value of market node, n, at time t. I_{nt} is a node' s influence on market node, n, at time t.

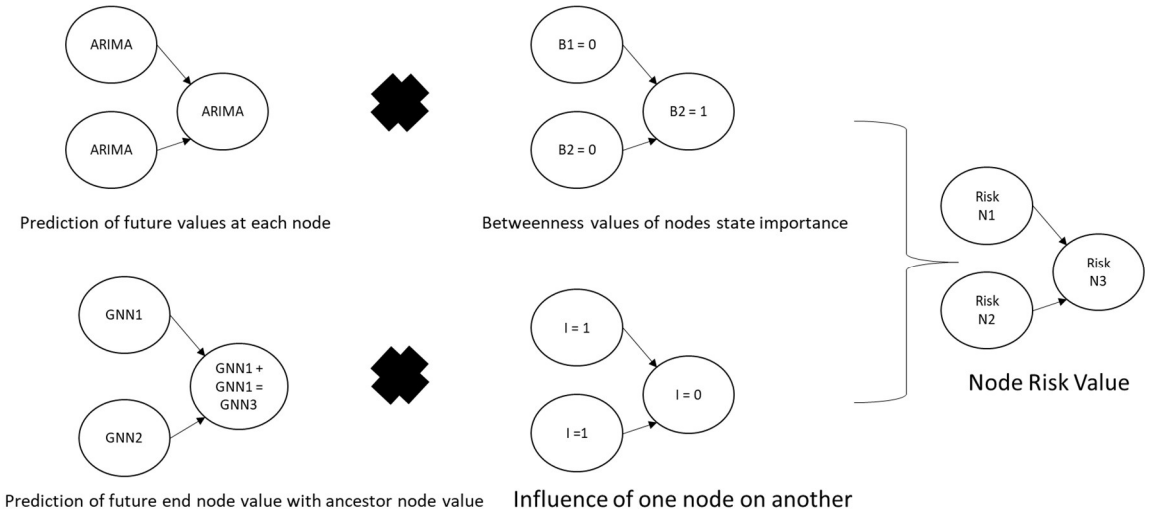

Figure 4.3 Research's Risk Equation Showing Each Method.

Figure 4.3 is a visual of the research's formulation and the sub-formulas that make the research's final risk value. This research uses the ARIMA function to predict future values for each node in the supply chain. Each node's historical data is input for separate ARIMA functions centering on a single node. Calculating the betweenness value for each node based on the supply chain's network structure. The betweenness value is the rating of how important a node is to the structure of the supply chain. The graph neural network considers the neighbor node's values when a GNN is training to learn how to predict future node value. This research predicts future node values but only uses the end nodes, since they represent the eventual outcome of the supply chain. A profit or loss outcome is what the end node values are. The influence of a node on another node is important to a supply chain. Knowing the influence value of a node on an end node helps to predict the impact of losing that node. A Bayesian Network can calculate influence

values because of its distributed probability structure. Combining these values together, the research can create a per node risk value that considers future values, importance, end node future outcomes, and a node's influence on end node values. Considering all these values within a risk calculation allows the research to build a quality risk metric for future decision-making tasks.

Representing the importance of the node to the supply chain. These values are inputs into the final risk equation 4.3, that gives a novel per node risk value. The values in each section represent the distinct features of a supply chain network. These features help define the effect a single node has on the supply chain. Understanding the importance of a single node plays within a supply chain helps to create a strategy to lessen the impact to the business. Minimizing the impact of a node failure is important to building a resilient supply chain. Identifying node of high risk can lead to creating standby or redundance nodes because interruption to the flow of product or information within a supply chain causes a loss of profit for a business. Applying risk impact to other entities within the supply chain can help a business minimize the impact to itself, meaning a manufacturer node is either internal or external site. Internal manufacturer would impact the main business while external manufacturer could be penalized for failing at a task.

The next section discusses the importance of a graph database and how this research uses it.

4.3. Graph Database

A graph database is the storage repository for all the graphs in the dataset. Graph databases do not need to have their data objects defined before creation, unlike rational database which requires a schema. The research builds a graph within the database by inputting the edges and their attached nodes. Each edge has a zero-order value, and each node has a zero capacity. This research reads each graph in the dataset into the graph database. The research uses the node names as labels for node within the supply chain graph. Adding an edge to the supply chain graph creates two nodes, if the nodes do not exist within the graph, and one edge. The direction of the edge is from source to destination. The research supply chain graph has 20 nodes and 32 edges. Populating supplier nodes with historical values from an open-source logistics dataset (Willems, 2008, see also Suiter et al., 2016) is the first step in filling the graph database with information. A simulation python script will populate the rest of the data values. The simulation was created for this research using NetworkX (Hagberg et al., 2008) and other libraries. Manufacturing nodes sum all incoming orders. Applying a daily failure rate and a reliability rate to summed orders creates a manufacturer's capacity value. Distributors have a delay buffer based on shipping trend data. If a delay exists, no product will appear at the retail nodes. If a bunch of orders reach zero delay, then all products will ship to retail nodes. Retail nodes sum of all incoming product' minus the demand. Adding left over products to the next day's shipment creates a retail node's capacity. The full graph represents the supply chain at a time snapshot. Stacking 360 graphs, one on top of another, creates the overall dataset that the risk algorithm processes. Algorithm reading

the dataset into memory for this research to generate risk values. This research can access a graph in the stack by time order, oldest to newest. Restricting the order of access allows for the research to disregard the timestamp on a graph. The graph neural network is observing 7,200 data points in two dimensions. The first dimension is across a graph and the second dimension is across time.

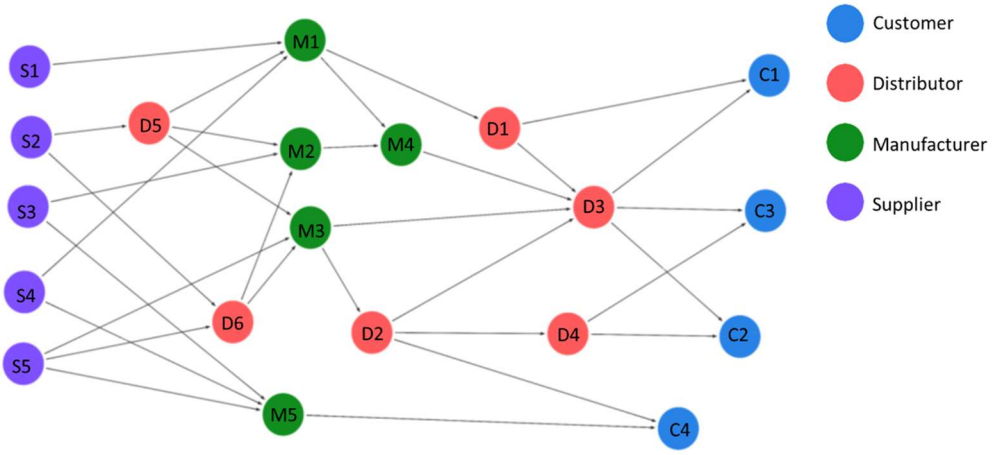

Figure 4.4 - Graph of the Supply Chain

The next section will discuss how ARIMA can forecast a failure probability for each node. The ARIMA model gathers all the data it needs from the graph database in memory. Each node contains a time series dataset, which the ARIMA function uses as input for its calculations. Front loading the input nodes in the supply chain with real supplier data from open-source datasets generates the time series for each node. The generation of internal and end node data is summing incoming data and multiplying it by trending time series to create an outside variance on the nodes. The research details the data generation algorithm in the results section.

4.4 ARIMA

Autoregressive Integrated Moving Average (ARIMA) statistical model is useful for time series predictions based on understanding historical data. Understanding ARIMA requires breaking the model down into its pieces. The autoregressive model (AR) understands how previous data points affect future data points, creating a correlation between points. An autoregressive model observes two correlations: auto-correlation and partial autocorrelation. Auto-correlation means all the values the model is considering have a correlation. Partial autocorrelation means one or more variables do not have a correlation.

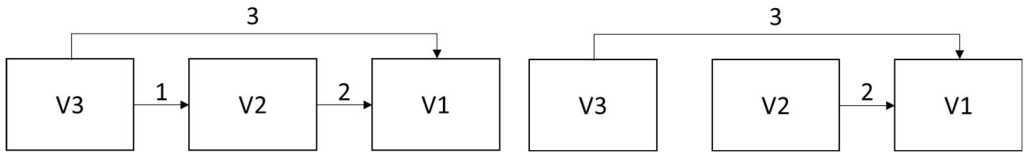

Figure 4.5 - Autocorrelation and Partial Autocorrelation

Time series datasets on average are large, so there will be multiple correlations. A correlation has a weight based on its distance from the end data. Recent data points carry a higher weight than older data. Newer data points influence predictions more than older data points.

Autoregressive creates the relationships between the data points. Integrated (I) is the consideration of the difference in raw observations so the model can smooth out stationary trends. Stationary defines the mean of the dataset as constant, standard deviation for the dataset is constant, and no seasonality meaning changes in the datasets

follow a trend. If the data is not stationary, then either the data needs a seasonal version of ARIMA or the data is too chaotic.

The last part is the moving average (MA), which uses observation and residual error to create a dependance from a moving average model over the lagged observation. Meaning the model compares the predicted values to the actual value to get an error value that is a representation of lag time.

An ARIMA model has three parameters: lag observations in the model, how many times to difference the raw observation, and the size of the average window. Any of these parameters can be zero, which removes that part of the model. This research finds the parameters by using autocorrelation and partial auto-correlation functions from Pandas library in Python.

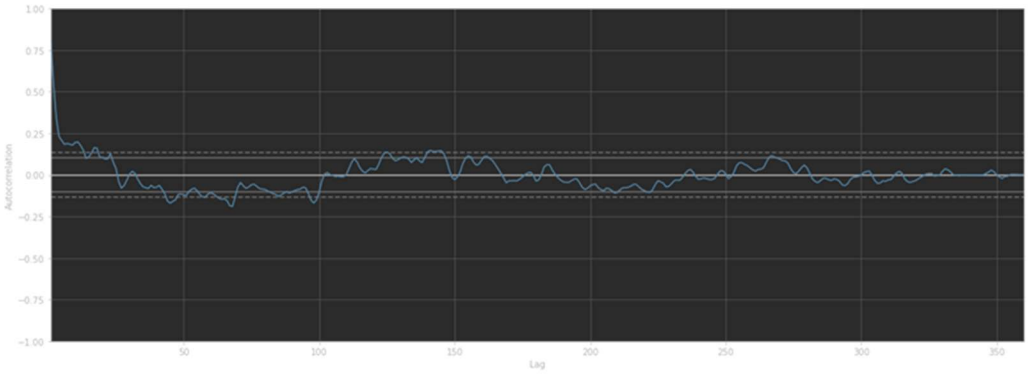

Figure 4.6 – Autocorrelation 95% Used for ARIMA Parameters

The autocorrelation with a 95% confidence graph, figure 4.6, shows the data stabilizes after.25. Meaning that the first parameter is one.

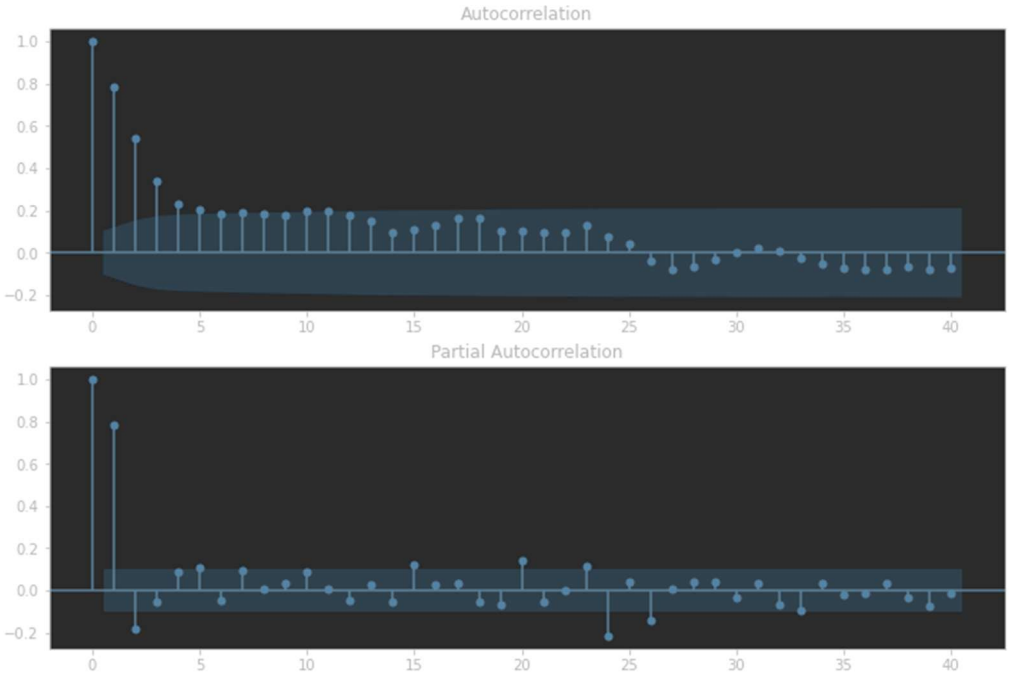

Figure 4.7 - Autocorrelation, and Partial Autocorrelation with Lag Graph

Figure 4.7 is a graph of autocorrelation and partial autocorrelation used to find the third parameter. The bars represent lag values in the data. The lags stabilize after two to three lag observations, meaning the third parameter is two or three. The middle parameter is usually one or two, higher values create too much smoothing in the data. The parameters for the ARIMA model are one, one, and two using the data in figure 4.6 and 4.7. Test the parameters by predicting values over the whole dataset and compare the difference. If values match, the parameters are good, otherwise changes the value within the limits. ARIMA makes several assumptions about the data. No anomalies are present in the data. Using constant error and parameters for the model. Historic trends have more weight than recent stress periods. The time series has a pattern to its data. Looking at ARIMA In a more formalistic way.

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \quad (4.5)$$

Where c is a constant, ε is the error term φ & θ are the autoregressive model's parameter, t is time, and i the moving average parameter for the model. ARIMA is a very simplistic model that requires data to follow a trend. Very seasonal data is not usable by a standard ARIMA model. Very unstable data will not work with an ARIMA model. Large historical trends can dominate smaller trends leading to the model not predicting high risk areas based on recent stress points. The research uses ARIMA as a failure prediction method because it is fast and accurate for most time series datasets. ARIMA gives an independent failure rate for each node. The data within the graph is interdependent but ARIMA does not take this interdependence in to a count. Graph Neural Networks can find interdependence within data and that is one of the reasons this research uses it. The next section goes into greater detail on graph neural networks.

Predicting future failure values is what the research uses the ARIMA function to do. The failure values are the base risk value for the node. The chance that the supply chain entity will not accomplish its task. Inventory levels or inventory capacity are the values of the nodes. Low capacity represents increases the chance that a node will fail its task of delivering product to the retail nodes. All other node capacity values are based on previous nodes higher in the chain. The capacities are interdependent, but ARIMA evaluates each node independently, losing specific interdependent changes can happen when using ARIMA to predict values. Nodes like the retail nodes heavily depend on

previous data. That is the reason GNN is a better fit than ARMINA. The next section discusses graph neural networks and other points.

4.5 Graph Neural Networks

Graph Neural Networks are a form of deep learning on graph structured data. Deep learning is machine learning that uses multiple layers of artificial neural networks to formulate an answer. Deep learning can take input data and map it to output data without feature extraction. Feature extraction is the act of removing important data from a large dataset. Deep learning models use artificial neural networks to extract important features from a dataset. Mapping those features to the output data. Artificial neural networks feed forward data starting at the input layer. Artificial neural networks take input data, multiply it by a weight and pass it to the next node. The next node is the hidden layer. The hidden layer sums all incoming weights add a bias value. An activation or transfer function within the hidden layer converts output value to an output range. There are many activation functions, one type is the sigmoid function. The sigmoid takes a value and squeezes it into a value between 0 and 1. A value of between 0 and 1 is good for probability. The output layer uses an activation function which can be the same as the hidden layer but does not have to be. Training an artificial neural network happens by comparing the output to a known value. The difference between the known value and output value is the update value. The artificial neural network then goes and updates all the weights in the network. The artificial neural network repeats the process until it reaches a desired output value.

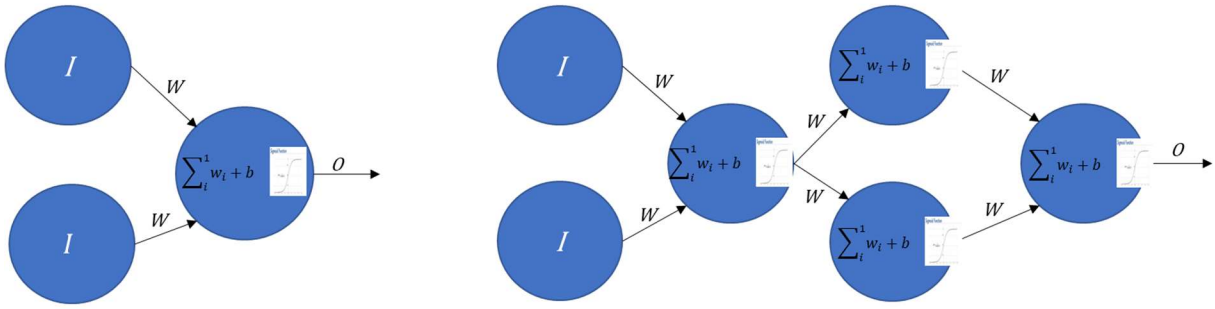

Figure 4.8 - Artificial Neural Network and Deep Learning Network

Deep learning takes an artificial neural network and uses it to encode the incoming data. The encoded data passes through n number of artificial neural network layers for processing before producing an output. Artificial neural networks are good at mapping input values to output values. If the data is spatial and temporal, then a convolutional neural network is a better choice. Spatial and temporal data are two to three dimensions in size, which is difficult for a standard artificial neural network since the input is not a single value. A convolution can take two and three-dimensional data and transform it into a single value. The convolution takes each value in the two or three dimensional and passes it through a filter. The filter multiplies the value within the array by a weight and squeezes the array into a single value. A third type of neural network is a recurrent neural network. A recurrent neural network has a memory function that allows it to remember data between training sessions. These are the three main types of neural networks that are used to build deep learning networks and graph neural networks.

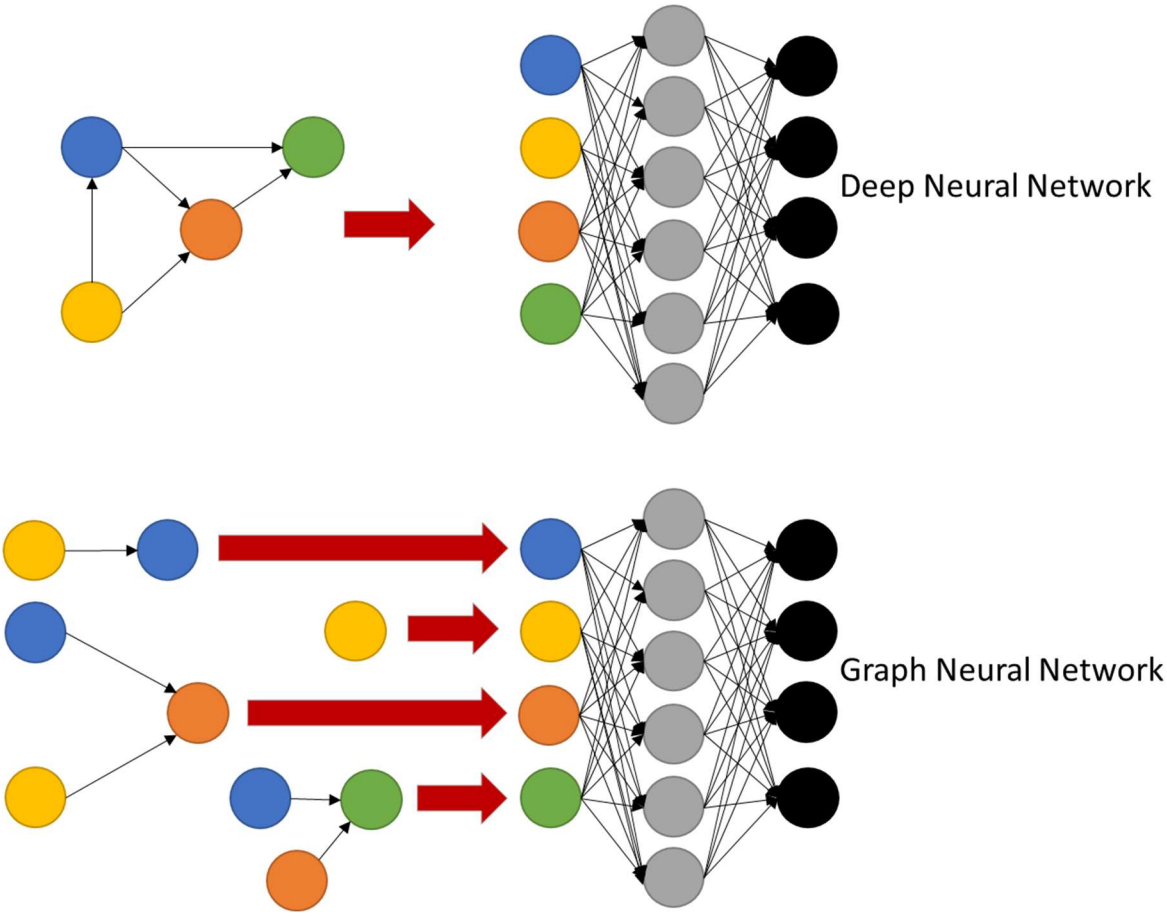

Figure 4.9 - Deep Neural Network vs Graph Neural Network

When the input has an underlying graph structure, deep learning can be inefficient. Figure 4.9. shows that a deep neural network only processes the data at each node. This type of algorithm misses the structural data of a graph. A graph neural network considers the nodes that influence the data being processed. A graph neural networks network uses a convolutional or recurrent neural network to encode node and graph data as the first stage. The convolutional and recurrent neural networks learn the structure and data held within the graph. The reason for using a graph neural network is

that standard deep learning models have difficulty understanding dependent data. Another issue is that deep learning networks cannot reliability encode the structure and order of the graph. General graphs have a unique structure that needs to be learned when reading in the data structure.

Graph data has a structure that is not well suited for general artificial neural networks. General artificial neural networks can take in data, but it can lose structure during analysis. The reason is deep learning networks view each data point as separate and independent parameters, but in a graph, the data is interdependent. Using an artificial neural network to encode a graph without taking the structure into count will create wrong output. The reason is that a graph has an order to the placement and connection of nodes. Accounting for this structure starts by taking neighbor data into account when encoding a node. The idea of neighbor sampling helps to encode the structure of the graph. The concept works very well with data such as chemical bonds. Encoding the graph structure of a chemical allows an algorithm to find similar graphs and replaceable nodes.

4.5.1 Structure of GNN

Graph neural networks are multi-layered systems. The first layer is the encoding layer. The encoding layer takes the graph structure and node features and encodes them into a single one-dimensional vector. Each one-dimensional vector is input for another neural network. The second layer is used for traditional neural network model analysis. The last layer decodes the data to rebuild the graph or creates output that the graph neural network compares to a known value. Figure 4.9 is a visual model of a GNN.

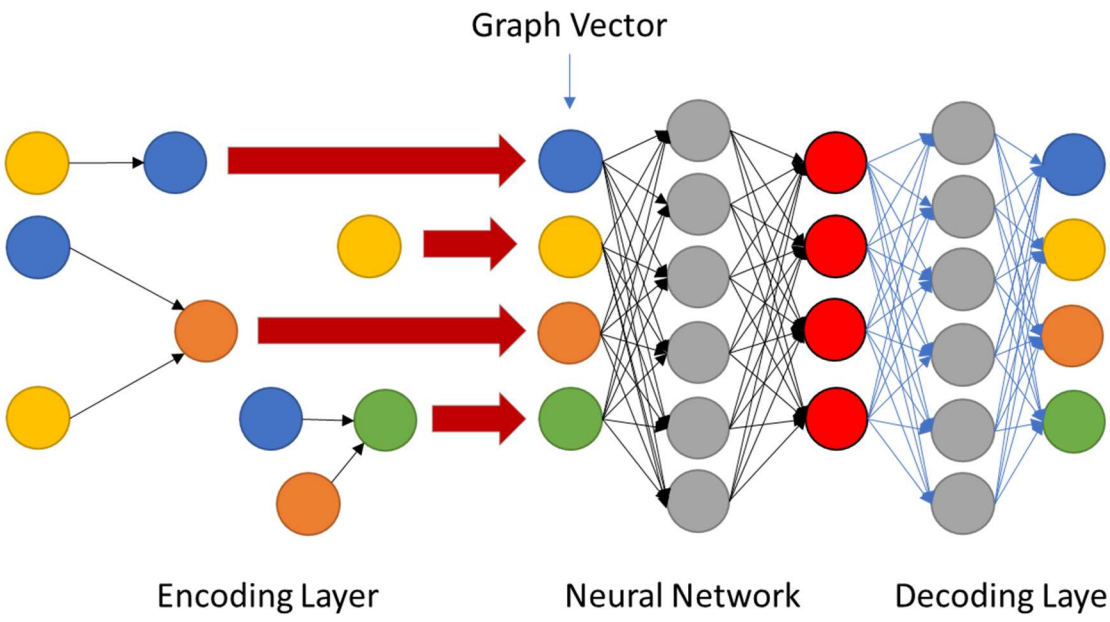

Figure 4.10 - Structure of a GNN

The encoding layer can use a neural network to encode the data. The data is not in a format that is usable by another neural network. Image data is the most common type of data the requires encoding before a neural network can act on it, but documents, networks, interdependent data have a graph structure that requires encoding as well. The research details the encoding scheme that the research implements in section 4.5.3 based on (Kipf et al., 2016).

Some graph neural networks stop at the encoding layer since the one-dimensional vector that is the output of the encoding process is useful in similarity algorithms. A similarity calculation on two vectors will create an output between 0 and 1. The closer the similarity value is to 1, the more similar the vectors are to one another. This research's second layer is trying to learn the trend of each nodes overtime. That is why the

research's second layer uses another neural network to learn from the output of the first neural network. The research uses a Long Short Term Memory (LSTM) (Qian, 1999) neural network to accomplish these tasks.

The third layer can decode the data by reversing the encoding process. The output will be an adjacency matrix and feature vectors. The third layer can also refine or decode the data into a format that is the same as the known output of training data. This research does not use a third layer.

This research only uses two layers for its graph neural network. The first layer encodes the graph structure, node features and edges weight into a single feature vector for each node. The second layer uses those feature vectors as input to learn trends in the data so it can predict future outcomes. The output of the second layer is a prediction, so a third layer is not required by this research's graph neural network.

4.5.2 Encoding and Decoding

The concept of encoding a graph using a convolutional neural network comes from image processing deep learning networks. Image processing deep learning network need to encode the image into a data format that is usable by another neural network. An image is a grid of values, and every point in that grid is a pixel color value. The concept of a grid with values is a one-for-one mapping of a matrix. A convolutional algorithm will move across the matrix using a windows size. The window is a smaller matrix with weight values. Aggregating each pixel in the image with all its neighbor pixels, reducing the image and generalizing it. The weights help to extract information from the image that other neural networks can use.

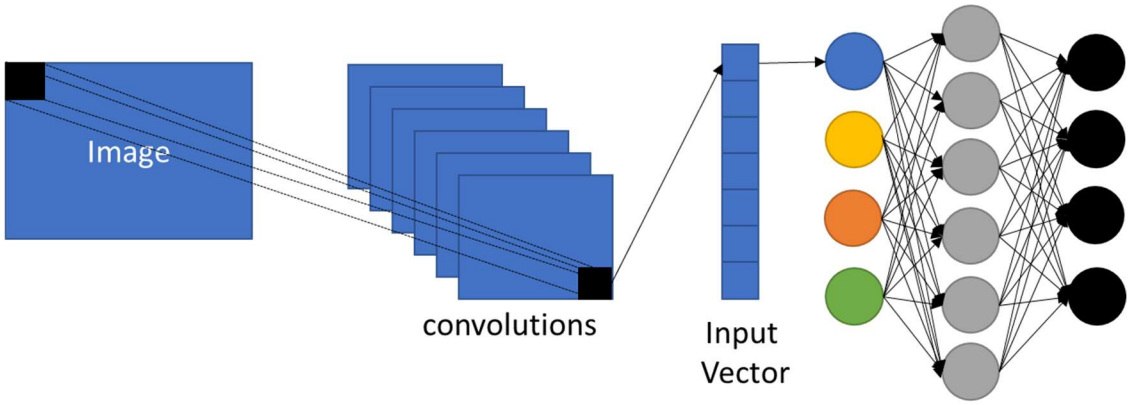

Figure 4.11 - Convolutional Neural Network

Viewing general graphs as a matrix is possible with graph theory. Graph theory states that all graphs can be represented as an n-by-n adjacency matrix where n is the number of nodes in the graph. Expressing the structure of a graph as a matrix allows algorithms to use matrix math to reduce the information within a graph into a one-dimensional vector. Encoding the graph for processing by a graph neural network starts by assembling an adjacence matrix. In an adjacence matrix, a one represents an edge is present and zero for no edge. Columns and rows are nodes, and data within the matrix is an edge. The research uses a directed graph, so an edge from node 1 to node 2 would show a 1 in the matrix but node 2 to 1 would be 0, see matrix 3 for example. The diagonal represents a self-loop on the node. A self-loop needs to be added to the adjacence matrix of the research's graph data, since the data does not contain self-loops. An identity matrix is a graph with only self-looping nodes. Adding the identity matrix to an adjacence matrix creates a graph with self-looping nodes. The reason for adding self-loop is that the update function uses the self-loop as a mechanism to update a node with all it's neighbor value plus its own value hence the current node value is seen as an

incoming value to the update function. The update function is part of the message passing algorithm explained later in the section.

The next part of the graph encoding is to embed the location data from the adjacence and identity matrix into the features or parameters of each node. Again, we can represent the features as a 1-dimensional vector held in a matrix. Stacking nodes one on top of another creates a feature matrix.

Multiple the adjacence matrix by the feature matrix will create a new n-dimensional, n is the number of features per node, feature vector for a node within the graph. Another matrix that clarifies the graph is the degree matrix. The degree matrix shows the degree of each node within the graph along the diagonal. The degree matrix can scale the data since high degree nodes will have larger values than low degrees nodes. Scaling the data helps to smooth the data out and it is easier to process.

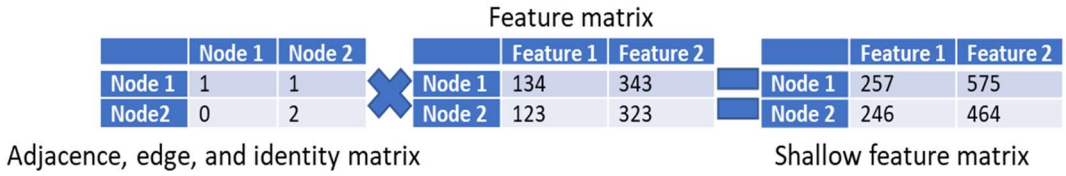

Figure 4.12 - Shallow Feature Vector Method

The problem with this method is the feature vector is shallow and does not take neighbor's features into count. The reason shallow vectors are a problem is that graphs have an arbitrary order, so using a simple embedding may not represent the actual graph, but a variant graph. Updating each node's feature vector with its neighbor's feature

vector will give the data local information. An embedding created from a node plus its neighbors feature data will better represent the data, and location recreating the dependent nature of the data. There are different methods for updating node data with neighbor data. This research first uses the message passing method based on convolutions. The concept of message passing in a graph neural network starts by viewing the graph in states. The first state is the input graph. Nodes pass feature information to their neighbor nodes, edge direction can restrict which nodes get updated, this is T+1 state where T is time. Each T+1 state is also known as the hidden state. The number of hidden states represents the number of times it takes to update all the nodes. The graph neural network model can use a message passing concept to update a node's hidden state, feature vector, with neighbors hidden states. The node being updated will receive messages from its neighbors. Aggregating all the hidden states, h, and edges, e, by a function m_v^t (4.7) that sums together all the values to create a single message. Similar to the stack of convolutions in figure 4.11.

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{4.7}$$

Using that message, the node updates its hidden state by using the following equation. U is an update function for that model.

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{4.8}$$

Every node uses the above equations to update its hidden state at time step t. Updating the hidden states will happen for a set number of time steps or until to graph has equalized. After all the time steps have processed, using function (4.8) a single graph feature vector is readout. Each time step represents a hidden layer within a neural network. Input layer is the shallow feature vectors built in the very being. Every update creates a hidden layer until an equilibrium, all nodes have shared their data, the last layer is the output. The output is for another neural network to continue the processing. Equation 4.9 shows the activation function of each hidden layer.

$$\hat{y} = R(\{h_v^t | v \in G\}) \tag{4.9}$$

This research's first attempt at a message update function is a convolution function, see equation 4.10.

$$X = D^{-1/2} A D^{-1/2} X \theta \tag{4.10}$$

D is the degree matrix, A is the adjacence matrix plus identity matrix, X is the feature vector and $\ominus$ is the edge weights of the graph. This research's first layer in the graph neural network uses the above method to build each node's final feature vector. The results of a convolution neural network as a message passing function did not give results with the desired accuracy. To improve the accuracy of the GNN a graph attention

network (GAT) will replace the convolution neural network in the massage passing neural network part of the GNN.

An MPNN using convolutional aggregation as an update message function is an efficient method when all the neighbor nodes are of equal importance. Learning the importance of a neighbor node by using a GAT as the aggregation function instead of convolutional aggregation introduces the concept of influence (Veličković et al., 2017). A GAT uses an attention parameter as an importance weight, W, for a node h. The model learns the importance weight during a training session. Equation (4.12) calculates the attention parameter, a. In equation (4.11) a is transposed and LeakyReLU is an activation function and || means to concatenate the values.

$$e(h_i, h_j) = LeakyReLU(a^T \cdot [Wh_i || Wh_j]) \tag{4.11}$$

Then the values are normalized using a Softmax function in equation 4.12. Nonlinearity σ is applied to the summation of N, neighbor nodes in (4.13).

$$a_{ij} = softmax_j\left(e(h_i, h_j)\right) = \frac{\exp\left(e(h_i, h_j)\right)}{\sum_{j' \in N_i} \exp\left(e(h_i, h_{j'})\right)} \tag{4.12}$$

$$h_i' = \sigma(\sum_{j \in N_i} a_{ij} \cdot W h_{j)} \tag{4.13}$$

Equations (4.11), (4.12), and (4.13) make up the GAT algorithm. This research uses a weighted graph to introduce the concept of importance by centrality. In this study,

four different centrality measures were tested as the weight values for each edge in the graph. Furthermore, the present work applies a centrality measure to the whole graph, meaning the edge weight represents one of the following: degree, betweenness, Eigen values, or Katz values' centrality. Degree centrality is the sum of in and out vectors for a node normalized whereas betweenness is a rating based on the shortest path algorithm. The more times a node appears on a short path, the higher its rating. The betweenness Equation (4.14) for node v where node s is the start of the path and node t is the end, σ_{st} is all the shortest paths, $\sigma_{st}(v)$ is shortest path with v, not as the end node.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4.14}$$

Eigen (4.15) values are the eigen vector for a node, which represents the node's importance to its neighbors. In Equation (4.15) A is the adjacency matrix or λ a constant and x is a one-dimensional matrix of node degrees. Katz values are a more generalized version of the Eigen values.

$$Ax = \lambda x \tag{4.15}$$

This study's GNN algorithm learns the structure, dependance between nodes, and time periods so it can make predictions on node capacity for the future supply chain graphs.

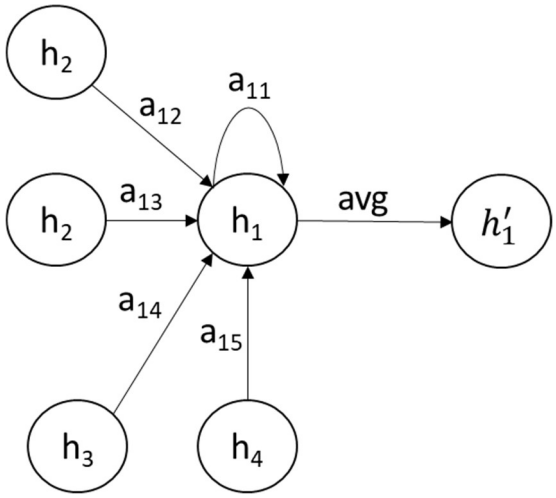

Figure 4.13 - GAT with Single Head and Centrality Values

Figure 4.13 shows a standard GAT where all the attention values a are averaged together but, in this study, the starting "a" values are replaced with centrality values. Katz centrality gives a node's importance to its neighbors, so the research uses Katz as the weight values for the experiments. The GAT sums the edge weight, Wh_i, and Wh_j together to calculate the attention value (4.13). The research replaces the random starting weights in a GAT with centrality values to help in training the model and accuracy.

The encoding or learning step detailed above can be fed into another neural network model. The research feeds the encoded output vector into a Long-Short Term Memory (LSTM) model. The LSTM captures the temporal dependence between each graph in a time step. The LSTM will understand the trend that can lead to node values changing over time. The reason LSTM can understand better than other neural network models is because the model can remember important data. LSTM built models use the idea of memory cells, similar to memory cells used by a computer. The LSTM cell takes

in the previous cell state, a hidden state, and the input. Using gates and functions, the cell learns what states to keep and merges that data with the input data to produce output data. Figure 4.14 shows the flow of data through the cell.

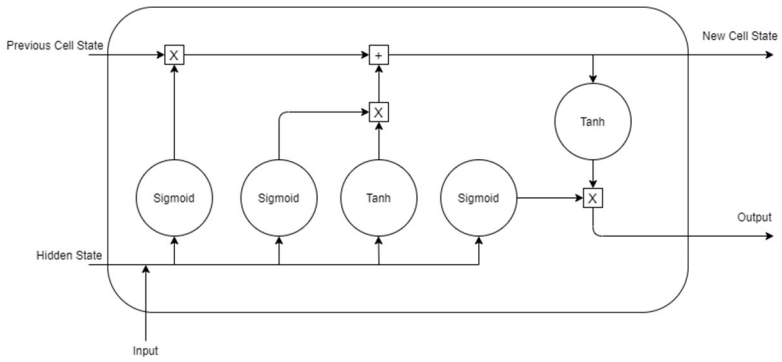

Figure 4.14 – LSTM ANN Model

The LSTM cell takes in the hidden state of the node and the input. The process within the starts on the left side and ends on the right. The top path creates the next cell state for the LSTM cell. The bottom path creates the output data for the LSTM cell. The first sigmoid function takes the input and hidden state and outputs a value between 0 and 1. A sigmoid function is an S curve function with a range of 0 to 1, non-linear and differentiable.

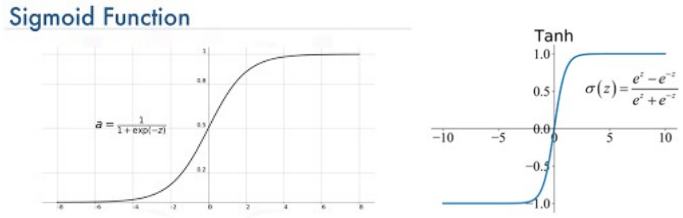

Figure 4.15 – Sigmoid and Tanh Activation Functions

The reason for using a sigmoid function is that it acts as a forget gate in the LSTM cell. Values of zero mean to forget, while values of one mean to remember. The value coming out of the first sigmoid function is multiple by the previous cell state. When the output is zero, the cell will forget data. As time goes on, the sigmoid function learns the importance of previous cell states. The second sigmoid function rates the importance of the hidden state and input. The first tanh function helps to control the data and stabilize the value. A tanh function is like the sigmoid function, but the range is between -1 to 1. The output of the second sigmoid is multiple by the first tanh function to produce a value. The cell adds the value to the current cell state, creating the cell state for the LSTM. The third sigmoid function acts as an activation function, taking the hidden state and input, creating a value v_s. The second tanh function takes the current cell state and creates a value $v_{t.}$ The LSTM multiples v_s and v_t this gives the final output for the LSTM. In summary, the first three functions build the new state of the LSTM cell, while the last two functions build the output value for the LSTM cell. This research uses an MPNN model to learn the structure and data dependance of a supply chain network at specific time. The MPNN encodes the data in the supply chain network for consumption by the LSTM model. Each one of the supply chain graphs is an input for the LSTM. The LSTM links previous supply chain graphs to current supply chain graphs, so the research can identify the highs and lows of the supply chain system and make predictions of future values.

4.5.3 Activation Functions

Activation functions, also known as transfer functions, define the output of a neuron within a neural network. The neuron sums all inputs, plus a bias is the input for the activation function. In the first part of the research's graph neural network, the activation function is a Rectified Linear Unit (ReLU). ReLU defines the positive part of the argument, meaning any value less than or equal to zero is zero.

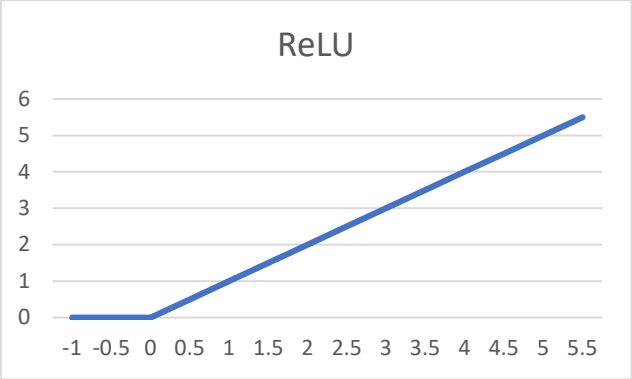

Figure 4.16 - ReLU Activation Function

The advantages of a ReLU activation function are sparse activation, better gradient propagation, efficient computation, and scale-invariant. Sparse activation means when randomly seeding the neural network only activates half the nodes. Better gradient propagation allows the neural network to avoid the vanishing gradient problem. The vanishing gradient problem is where the error value, gradient, is tiny in the early stages and decreases exponentially. Leading the neural network to even tical shut down and early training stages are very slow. Computational factor of ReLU is small since it is a comparison, addition, and multiplication. ReLU scales well because it is a max function,

max (0, ax) = a max(0, x) where a is greater than or equal to 0. ReLU is a very common activation function for graph convolutional networks.

The Long Short Term Memory neural network, in the second part of the graph neural network, uses two activation functions, Sigmoid and Tanh. Sigmoid is an s-shaped function that is first derivative is a bell shape curve. The equation is (4.15) and figure 4.15 shows the shape.

$$f(x) = \frac{1}{1+ e^{-x}} \tag{4.15}$$

The output of a sigmoid function is between 0 and 1, which makes it useful for probability calculations. A sigmoid function can fit the data better but requires more time to calculate than ReLU. Sigmoid can activate more nodes within a neural network than ReLU. The LSTM also uses a Tanh, hyperbolic tangent function for part of its activation. A tanh (4.16) activation function is like a sigmoid because it is also an s-shape curve function. The bounds of the function are -1 and 1 so negative values are strongly negative and zero maps to near zero. The function is monotonic and differentiable, but the derivative is not monotonic.

$$f(x) = \tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4.16}$$

The Tanh has the same disadvantages and advantages as sigmoid. Both functions are part of the LSTM node itself.

4.5.4 Weight update and optimization method

The method that the research uses to adjust weights within the neural network is backpropagation. Backpropagation is the idea of traveling back through the neural network layers updating the weight and bias values. The algorithm starts at the output layer and works its way to the input layer. The value that backpropagation uses to judge that the updated weights and biases are good is the output of the cost function, C in equation 4.17.

$$\theta = \theta - nG_{\theta}C(\theta; x, y) \tag{4.17}$$

The cost function is a function that measures the cost or error between the input value and the expected output value. A cost function produces a value between one and zero, $\ominus$ or theda, with one being completely wrong and zero being exactly right. Backpropagation's primary aim is minimizing the cost function, it does this by calculating the gradients, $G_{\ominus}$. The gradients are the value that is used to update weights and biases. Calculate the gradient by taking the partial derivative of the cost function as it relates to weight or bias.

$$w = w - n\frac{\partial C}{\partial w} \tag{4.18}$$

Adjusting weight so that the cost function derivative decreases. If the derivative is positive, the algorithm decreases the weight, otherwise the increase the weight.

Increasing and decreasing weight to find the correct gradients to use is called gradient decent. As the algorithm travels back through the network, previous level gradients affect the next level gradients. Using the chain rule, backpropagation can calculate the dependent gradient values. Controlling the rate of movement through the cost function is the learning rate. The learning rate, n, is the amount gradients are increased or decreased while optimizing the cost function. Stochastic gradient decent, a stepwise optimization of a neural network using backpropagation.

$$w_j = w_j - n\frac{\partial C}{\partial w_j} \tag{4.19}$$

Stochastic gradient descent is the basic optimization method for a neural network, but it has many disadvantages. First, it is slow to converge to an optimal solution. Second, it finds a local minimum, not the actual minimum value. To fix these disadvantages, other optimization methods extend the basic gradient decent. The research uses the Adam method to optimize its gradient decent. Adam stands for Adaptive Moment Estimation, which means using the moment method plus an adaptive learning method. Moment optimization (Qian, 1999) introduces a temporal value into the basic gradient decent method.

$$\theta = \theta - nG_\theta C(\theta; x, y) + \gamma_t \tag{4.20}$$

The temporal value, a time-based element, γ_t, helps to move through the cost function curve faster. The temporal value is a constant that is added to stochastic gradient

decent method. The weights are input into a cost function, which is multiplied by a learning rate, then a moment value multiplied by previous weight update is added. It is possible to reach convergence faster with the moment method, but the method doesn't know if the minimal local been reach. The adaptive learn rate is a combination of AdaGrad (4.21) and RMSprop (4.22).

$$\theta_{t+,i} = \theta_{t,i} - \frac{n}{\sqrt{\epsilon + \sum_{r=1}^{t}\left(GC(\theta_{r,i})\right)^2}} GC(\theta_{t,i}) \tag{4.21}$$

$$\theta_{t+,i} = \theta_{t,i} - \frac{n}{\sqrt{\epsilon + E|g^2|_t}} GC(\theta_{t,i}) \tag{4.22}$$

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 4.17 - Adam Algorithm

The finial Adam Algorithm, figure 4.17, uses the AdaGrad and RMSprop within the step to converge faster with a better ability to find a minimum cost value. The next section overviews the cost function the research uses.

4.5.5 Error calculation

Understanding how well the graph neural network can predict future values is based on mean squared error calculated during the validation cycle. Every time the algorithm runs through a forward pass. The algorithm compares the output value to a known output. The mean squared error, cost, function tells the neural network system how well it did at calculating the correct output.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y_i})^2 \tag{4.23}$$

The MSE value is the theda value in the previous equations from section 4.23. Using the theda as the base of the Adam function, a weight update value is the calculated. The next section discusses the centrality algorithm, part of the risk formulation.

4.6 Centrality Algorithms

Centrality algorithms can highlight nodes within a graph for farther analysis. The centrality measure helps to determine how important a node is within a graph structure (Negre et al., 2018, see also Borgatti, 2005). Centrality measures fall into two groups: flow and path. Flow is the concept of information or product flowing through a network from one point to another. Degree centrality algorithms are in the flow group. Degree focuses on the number of connected vertices to a node. The higher the degree, the more information

flows into or out of a node. Path base centrality focuses on walking through a network. The shortest path walks are the foundation for betweenness centrality measures (Freeman, 1977).

The research uses a betweenness centrality algorithm to find important nodes within a supply chain. The importance of the node is its betweenness score, which the research uses to change the base failure probability. The change to the base failure probability is a representation of that node's importance to the supply chain betweenness centrality is a graph method for finding how connected a node is within a graph. The between centrality algorithm runs a shortest path algorithm through all the node pairs in a graph. Every time a node appears within a shortest path, the algorithm increments its betweenness score by 1. Dividing the betweenness score by the total number of the shortest paths within a graph gives the betweenness value.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4.24}$$

The value g(v) is the betweenness of a node within the graph g. The σ_{st} is the total number of shortest paths through the point s and t. $\sigma_{st}(v)$ is the total number of shortest paths where v appears. The nodes with the highest betweenness value are very important to the graph, since removing them would break connectivity.

This research applies the degree, eigenvector, and Katz centrality to supplement the between values. The issue with between is that start nodes, suppliers, and end nodes, customers, in a supply chain would have between of zero since they are not on a path.

Suppliers and customers having a betweenness score of zero give a fault sense that these nodes don't matter to the supply chain, but the reality is that suppliers and customers are important. Degree centrality can bring back supplier and customer importance by giving them value. Degree centrality counts every node's in and out vector and assigns that value as a degree value. The degree value is normalized by dividing the degree value by n -1 where n is the number of nodes in a graph. Degree centrality doesn't take neighbor nodes into account when building its degree value. Eigenvector centrality finds a node's centrality based on its neighbor's centrality. Eigenvector gives the research an idea how important a node is to its neighbors.

$$Ax = \lambda x \tag{4.25}$$

Where A is an adjacency matrix x is vector and λ is the eigenvalue. The research uses the power method (Bonacich, 1987, see also Newman, 2010) to find the eigenvector. The only issue with the power method is the error tolerance needs to be adjusted so the method will converge which is not guaranteed. Katz centrality (Katz, 1953) takes the eigenvector centrality and generalizes it.

$$x_i = \alpha \sum_j A_{ij} x_j + \beta \tag{4.26}$$

$$\alpha < \frac{1}{\lambda_{max}} \tag{4.27}$$

Where A is an adjacency matrix of graph G, β is parameter that controls initial centrality and α must be less than one over the max eigenvalue λ. The value x_i is a general influence value of a node within its network of node. The network is defined by first order neighbors and other connected nodes.

Betweenness centrality in a supply chain would identify important or bottleneck entities that are critical to the supply chain. High betweenness values help to identify hub entities. Hub entities are typical high-risk areas since a hub handles a lot of products. Removing these entities could break or severely hamper a supply chain. That is the reason the research uses the between to change the base failure probability. Increasing a node's failure value by its importance modifier gives the risk probability value a system level impact. Meaning if a high between nodes fails it will affect the supply chain more and create larger business impact. The next section details how the research builds an influence value on affecting business impact.

4.7 Bayesian Network

Understanding cause and effect within a supply chain is critical. Knowing why a node fails or how risk propagates through a supply chain is important to building resilient systems (Nuss et al., 2018, see also Soberanis, 2010, Lockamy III et al., 2012, Hosseini et al., 2020). Bayesian networks can show cause and effect, decision making, and probability propagation.

Bayesian networks are directed acyclic graphs that allow a user to infer which outside influencers created the outcome at a particular node. The nodes in a Bayesian network represent a variable probability of a nodes state. The edges represent conditional

dependencies on a variable. Using joint probability, distributive law, and Bayes theorem we can understand the likelihood of an event happening given some other information about outside factors. In a supply chain, Bayesian networks can help with risk management, decision making, and time series calculation. The research focuses on the propagation of influence through a supply chain network. Each supply chain entity can influence other entities. A node failing to complete a supply chain task causes other nodes lower down the chain to fail as well. The amount of failure that transfers from one node to another is its influence value. The analysis of influence using Bayesian networks shows which entities will fail together. Knowing that a certain group of supply chain entities will fail allows a supply chain manager to add or redistribute entities in a supply chain, which should mitigate a failure.

$$P(V_1, V_2, \cdots, V_k/v) = \prod_l^k P(V/v)\ (i = 1,2, \dots k \tag{4.28}$$

$$P(V_1, V_2, \cdots, V_k) = \prod_l^k P(V_i/Parent(V_i))\ (i = 1,2, \dots k) \tag{4.29}$$

Each node in a Bayesian Network has a conditional probability table, except nodes that do not have incoming edges. A node with two incoming edges has eight probable states. The incoming edges carry a probability state of the two incoming nodes. These probability states affect the node's state. Nodes within the network depend on adjacent nodes for determining their state value. The ancestor nodes farther up the chain, influence their descendants. The amount of influence an ancestor has on a decedent is

like the mutual information equation (Ebert-Uphoff, 2007, 2009). Mutual information equation (4.30) comes from information theory. The equation shows the amount of uncertainty a node removes from the target node. Nodes directly connect to one another remove the most uncertainty. Removal of uncertainty and influence are similar in this situation. Using the mutual information equation within a Bayesian network requires a network structure, node probabilities, and conditional probability tables.

$$MI(X,Y|Z) = \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log_2 \frac{P(y|x,z)}{P(y|x)} \tag{4.30}$$

Knowing the state of node X, how much uncertainty does node X remove from node Y. If node X and Y are not adjacent to each other than nodes in between, need to be considered. Z is all the node states for the nodes in between node X and Y. These probabilities are part of equation (4.19) hence, the mutual information can mean influence of node X on node Y.

The supply chain graph is the Bayesian network structure, and each node contains the historical data from the time series. Using the data provided, the network will learn its conditional probability values for each node, parameter learning. The parameter learning algorithm uses expectation maximization to find the parameters. The MI will then be able to calculate the influence each node has on the target retail nodes.

The amount of influence a node has on another node shows the amount it can affect final supply chain risk. Influence and impact on profit are similar concepts within the context of the supply chain.

The next section discusses combining all these separate values into a final individual risk value for a node.

4.8 Integrated Risk Metric Calculator

The risk calculation is a combination of betweenness, predicted ARIMA capacity, node influence on each customer, and the predicated loss value for each customer. The betweenness value shows how important a node is to the supply chain. Supplier and customer nodes have a value of zero since they are the beginning or end of a supplier chain. Suppliers and customers are important to a supply chain, but they are easily replaceable, so their risk to a supply chain is zero. The formula filters out suppliers and customer nodes, so their risk values are always zero. Other nodes have a betweenness value rating their importance to the supply chain, which is multiplied by the next value. The next value is the predicted ARIMA capacity value. Converting the predicted capacity to a percentage keeps the values between 0 and 1. The research's formula minus the ARIMA capacity percent from one' If the node is running at a 100 percent capacity, its risk will be zero. 100 percent capacity means the node is fully operational at the time interval, so no risk of failing at a task. Multiplying these two values by a grouping of node influence on each customer times predicted loss value for each customer. Using the Bayesian network from section 4.8, the research calculates node n's influence value o' each one of the customer nodes' The research sums each one of the separate calculations into a single value, I stands for customer number and n stands for the current node. The output is the node n's influence on customer I. The next value in the two value group is the M_{nt} where M stands for the customer number. The graph neural network predicts

capacity value for each customer at time t. The predicted value is a normalized capacity value between 0 to 1. The demand value for each customer node is compared to the capacity. If the demand is bigger than the capacity that difference is the profit loss. Multiplying the profit loss by node influence gives the two group values. Summing these group values together creates the value that will be multiplied by betweenness and ARIMA value. All these values together give us the final risk value for a node at time t. Equation 4.31 shows the final formula that calculates the risk value for each node.

$$R_{nt} = B_{nt} * (1 - A_{nt}) * (\sum_{n}^{market\ node} M_{nt} * I_{nt}) \quad (4.31)$$

The risk value built by combining these distinct factors together gives the research a novel multifaceted view of the risk a single supply chain entity can pose to a supply chain network. The reason the research's formula can show the risk of a node on a chain supply comes from each of the values. The betweenness shows the importance of the node on the supply chain network. The capacity is the value the research uses to gauge success or failure of a supply chain entity. Capacity is a simple measure of how well a supply chain entity is accomplishing its task. Understanding the influence of a supply chain entity on a customer helps to understand how that node will affect the final product of the supply network. Nodes with a larger influence on customer nodes will bring down a supply chain quicker, with a larger loss. Predicted loss value helps understand how a node will affect the profitability of a supply chain network. Rating a supply chain network by its ability to produce products and the profit it creates are simple indicators of

health. The combined indicator allows the research to qualify the risk an entity proposes to a supply chain network.

4.9 Summary

The research brings together multiple methods to predict future risk and how a single entity can affect a supply chain network. This research starts by moving the data out of a relational database table format. Creating a graph structure from the data shows the interconnection between the data. Most, if not all, the data a supply chain generates is dependent data. The data from a node affects other nodes within the network. In a large supply chain, it is difficult to see all the connecting nodes. The graphs with data values are the input to a prediction model. The research predicts capacity of a supply chain entity on a per node basis.

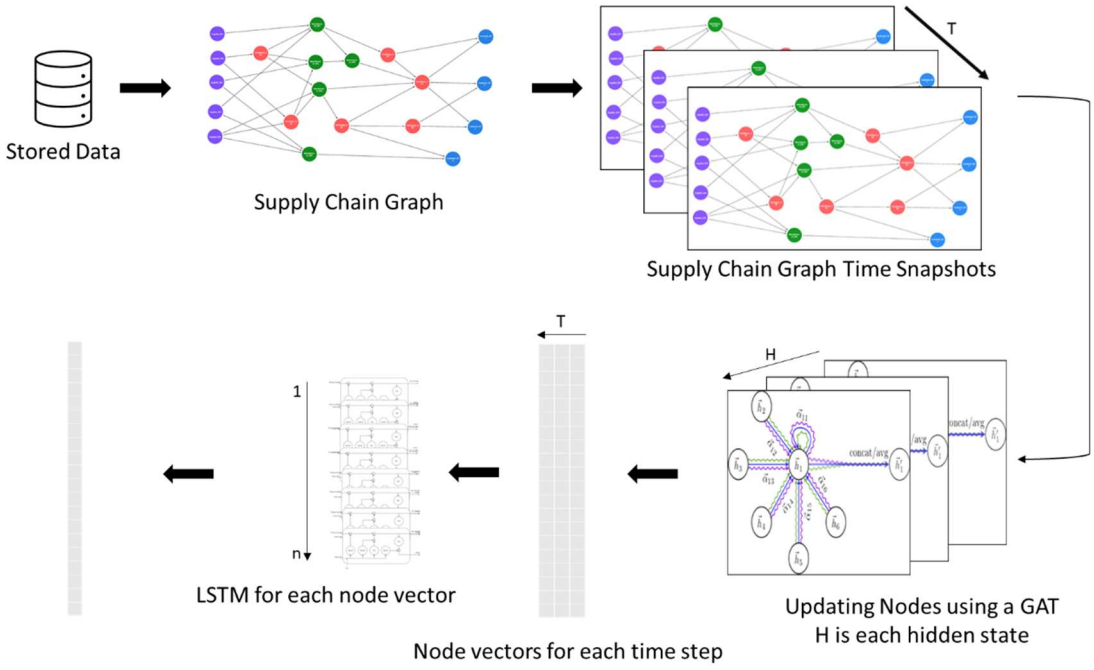

Figure 4.18 - Graph Neural Networks Pipeline

The prediction model uses a graph neural network, figure 4.18, to understand data dependance, node position, network structure, and time dependance. Help to show and understand the profit loss at a customer entity. The model makes predictions based on the training history. Using a Bayesian network will show node influence on customer entities within the supply chain network. The Bayesian network will use a uniform probability formula for every node. Using the centrality algorithms, each node shows its importance to a supply chain network, which changes the overall risk value. The overall formula will combine multiple indicators to produce a risk value at a node per node level. The next section will detail the prototype system and compare it against statistical algorithms.

CHAPTER FIVE

Results

In this chapter, the research will detail the results of the experiment. The research runs separately, each model generates the data for the final risk calculation. ARIMA generates the first set of data. The graph neural network model runs second and generates its prediction. The research uses a static supply chain graph, so betweenness and Bayesian network influence is only on the first graph. The research collected each of the separate pieces using a risk calculation python script. The script does the calculation and outputs a risk for each node. The next subchapter will detail the ARIMA data.

5.1 ARIMA data

Charts show a better detail as to how ARIMA is predicting the data. The next set of charts are line charts with an actual data line and a prediction data line. The research creates a chart for each of the supplier entities in the supply chain. The ARIMA fit model gives statistical results. Supplier_001 model uses an ARIMA (0,1,3) as the parameters. Supplier_002 ARIMA model uses the parameters ARIMA (0,1,18). Supplier_003 uses the ARIMA (0,1,2) for its parameters. Supplier_004 uses the ARIMA (1,1,6) parameters for its model. Figure 5.4 shows that ARIMA has difficultly predicting for Supplier_004. Supplier_005 uses the ARIMA (0,1,1) for its model parameters.

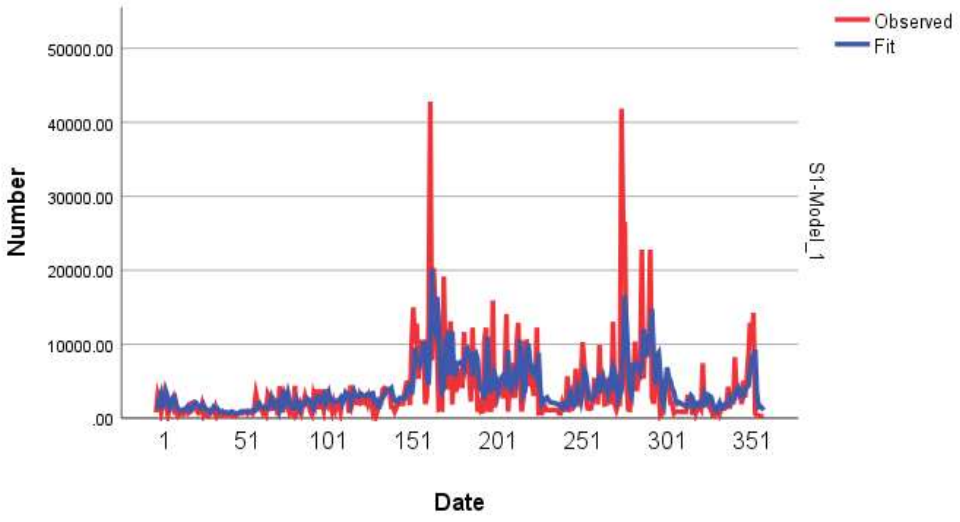

Figure 5.1 - Supplier 1 Actual and Predicted Data

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-squared	.287	.	.287	.287	.287	.287	.287	.287	.287	.287	.287
R-squared	.224	.	.224	.224	.224	.224	.224	.224	.224	.224	.224
RMSE	4446.359	.	4446.359	4446.359	4446.359	4446.359	4446.359	4446.359	4446.359	4446.359	4446.359
MAPE	120.096	.	120.096	120.096	120.096	120.096	120.096	120.096	120.096	120.096	120.096
MaxAPE	1855.663	.	1855.663	1855.663	1855.663	1855.663	1855.663	1855.663	1855.663	1855.663	1855.663
MAE	2440.151	.	2440.151	2440.151	2440.151	2440.151	2440.151	2440.151	2440.151	2440.151	2440.151
MaxAE	38156.494	.	38156.494	38156.494	38156.494	38156.494	38156.494	38156.494	38156.494	38156.494	38156.494
Normalized BIC	16.832	.	16.832	16.832	16.832	16.832	16.832	16.832	16.832	16.832	16.832

Table 5.1 - Supplier_001 ARIMA Model Fit Results

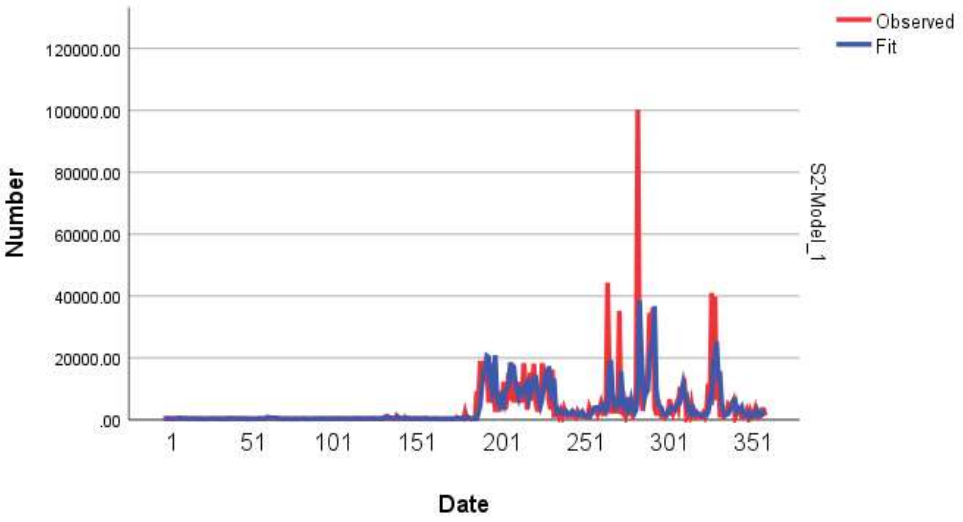

Figure 5.2 - Supplier 2 Actual and Predicted Data

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-squared	.188	.	.188	.188	.188	.188	.188	.188	.188	.188	.188
R-squared	.219	.	.219	.219	.219	.219	.219	.219	.219	.219	.219
RMSE	7262.159	.	7262.159	7262.159	7262.159	7262.159	7262.159	7262.159	7262.159	7262.159	7262.159
MAPE	73.265	.	73.265	73.265	73.265	73.265	73.265	73.265	73.265	73.265	73.265
MaxAPE	883.791	.	883.791	883.791	883.791	883.791	883.791	883.791	883.791	883.791	883.791
MAE	2554.851	.	2554.851	2554.851	2554.851	2554.851	2554.851	2554.851	2554.851	2554.851	2554.851
MaxAE	90486.567	.	90486.567	90486.567	90486.567	90486.567	90486.567	90486.567	90486.567	90486.567	90486.567
Normalized BIC	17.814	.	17.814	17.814	17.814	17.814	17.814	17.814	17.814	17.814	17.814

Table 5.2 - Supplier_002 ARIMA Model Fit Results

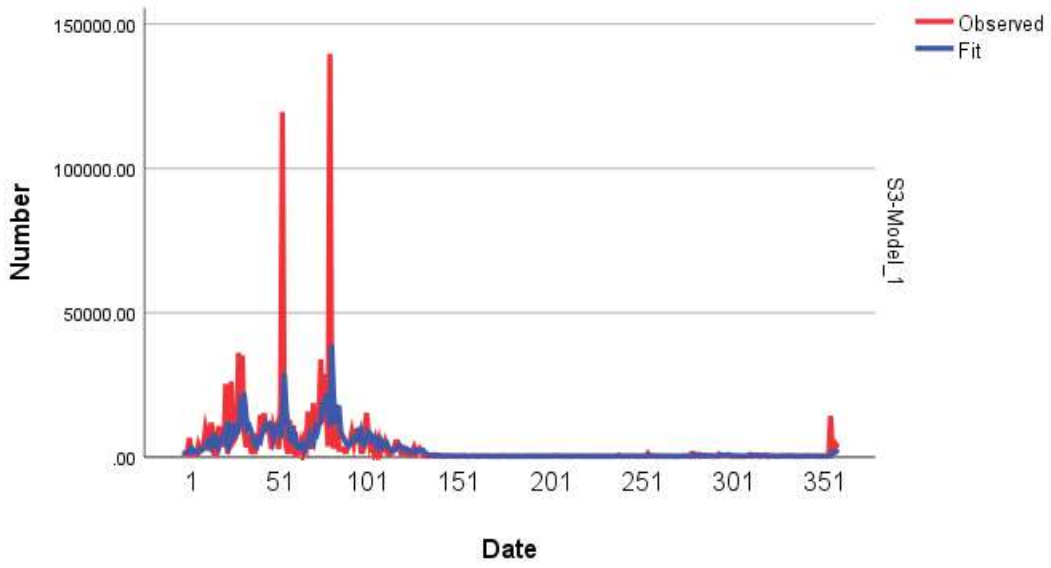

Figure 5.3 - Supplier 3 Actual and Predicted Data

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-squared	.271	.	.271	.271	.271	.271	.271	.271	.271	.271	.271
R-squared	.164	.	.164	.164	.164	.164	.164	.164	.164	.164	.164
RMSE	9948.900	.	9948.900	9948.900	9948.900	9948.900	9948.900	9948.900	9948.900	9948.900	9948.900
MAPE	69.332	.	69.332	69.332	69.332	69.332	69.332	69.332	69.332	69.332	69.332
MaxAPE	883.889	.	883.889	883.889	883.889	883.889	883.889	883.889	883.889	883.889	883.889
MAE	2567.086	.	2567.086	2567.086	2567.086	2567.086	2567.086	2567.086	2567.086	2567.086	2567.086
MaxAE	128958.348	.	128958.348	128958.348	128958.348	128958.348	128958.348	128958.348	128958.348	128958.348	128958.348
Normalized BIC	18.443	.	18.443	18.443	18.443	18.443	18.443	18.443	18.443	18.443	18.443

Table 5.3 - Supplier_003 ARIMA Model Fit Results

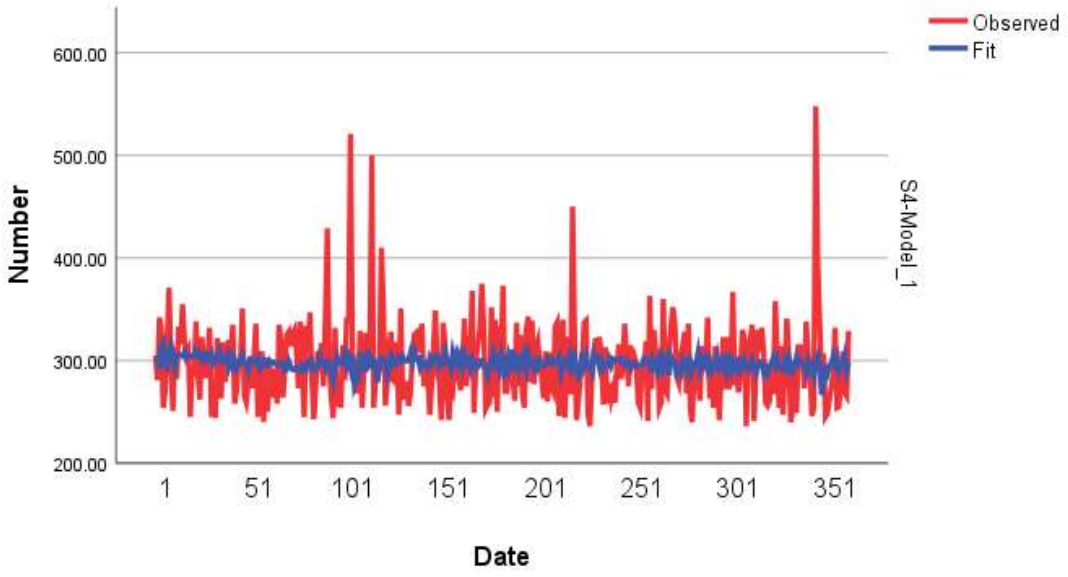

Figure 5.4 - Supplier 4 Actual and Predicted Data

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-squared	.489	.	.489	.489	.489	.489	.489	.489	.489	.489	.489
R-squared	.010	.	.010	.010	.010	.010	.010	.010	.010	.010	.010
RMSE	40.388	.	40.388	40.388	40.388	40.388	40.388	40.388	40.388	40.388	40.388
MAPE	9.884	.	9.884	9.884	9.884	9.884	9.884	9.884	9.884	9.884	9.884
MaxAPE	45.184	.	45.184	45.184	45.184	45.184	45.184	45.184	45.184	45.184	45.184
MAE	29.713	.	29.713	29.713	29.713	29.713	29.713	29.713	29.713	29.713	29.713
MaxAE	247.608	.	247.608	247.608	247.608	247.608	247.608	247.608	247.608	247.608	247.608
Normalized BIC	7.528	.	7.528	7.528	7.528	7.528	7.528	7.528	7.528	7.528	7.528

Table 5.4 - Supplier_004 ARIMA Model Fit Results

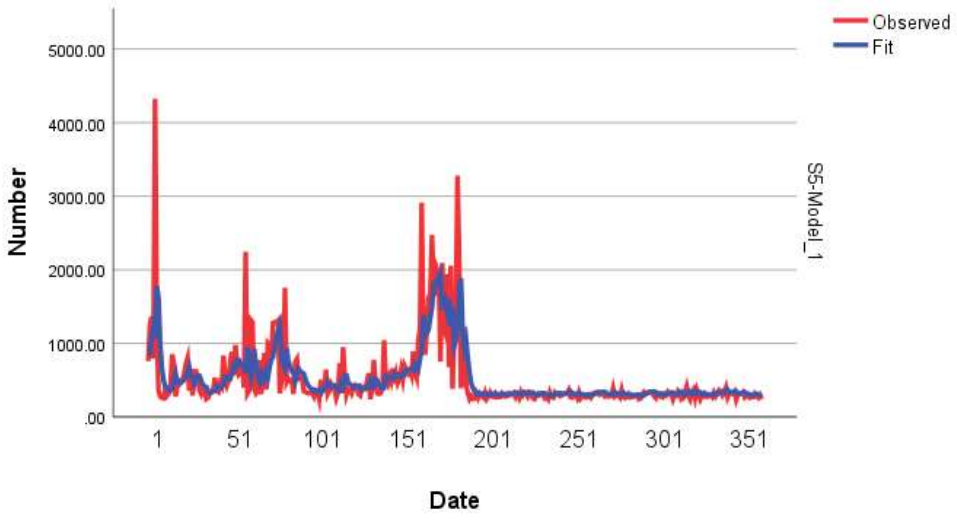

Figure 5.5 - Supplier 5 Actual and Predicted Data

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	10	25	50	75	90	95
Stationary R-squared	.270	.	.270	.270	.270	.270	.270	.270	.270	.270	.270
R-squared	.431	.	.431	.431	.431	.431	.431	.431	.431	.431	.431
RMSE	353.980	.	353.980	353.980	353.980	353.980	353.980	353.980	353.980	353.980	353.980
MAPE	29.147	.	29.147	29.147	29.147	29.147	29.147	29.147	29.147	29.147	29.147
MaxAPE	383.462	.	383.462	383.462	383.462	383.462	383.462	383.462	383.462	383.462	383.462
MAE	164.190	.	164.190	164.190	164.190	164.190	164.190	164.190	164.190	164.190	164.190
MaxAE	3263.030	.	3263.030	3263.030	3263.030	3263.030	3263.030	3263.030	3263.030	3263.030	3263.030
Normalized BIC	11.755	.	11.755	11.755	11.755	11.755	11.755	11.755	11.755	11.755	11.755

Table 5.5 - Supplier_005 ARIMA Model Fit Results

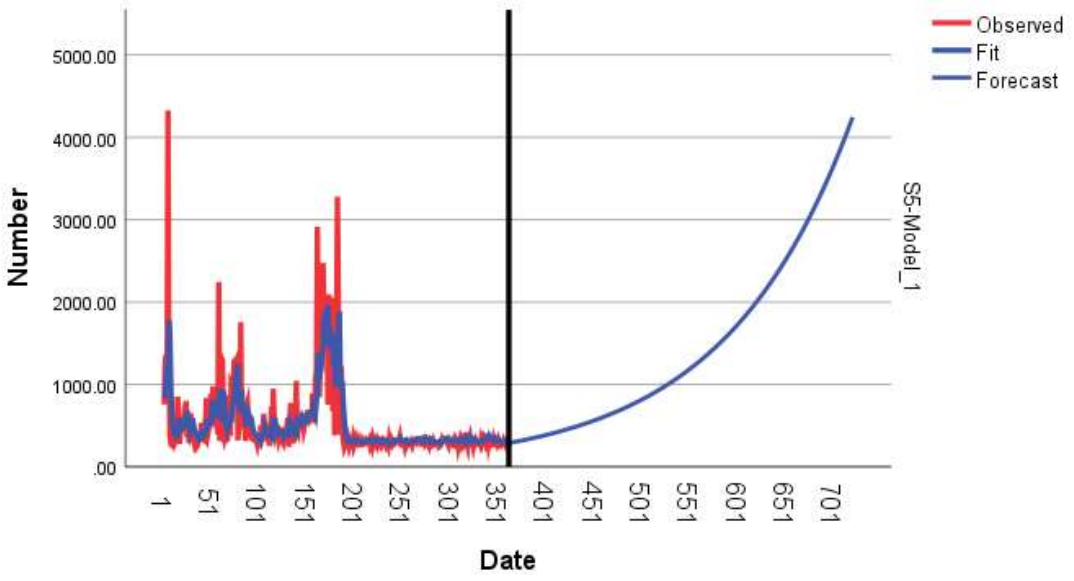

Figure 5.6 – Supplier_005 Forecast Out 360 Days

The curve of the forecast, values after 360 date in figure 5.6, is an upward arcing exponential style function. The reason for this type of result is that there is not enough data to build a better forecast model. Another reason is that the overall fit is not perfect either. ARMIA may not give the best results for the dataset we are using, that is why this research looks to a graph neural network.

5.2 Graph Neural Network

The graph neural network predicts a value for everyday in the training set. The reason for so many predictions is that the model will run multiply times until the error values during training are minimum. The training set is twenty percent of the dataset. The other eighty percent is a validation set. The validation set proves the model is predicting values correctly. The research runs sixteen experiments on the dataset. The research divides the experiments into two groups using different learning rates. Each group varies

hidden layers and dropout rate during four experiments. These two groups are run a second time to verify the results of the first group of experiments. The research uses the mean squared error to gauge how well the model is predicting the target values.

The research compares the MPNN+LSTM model to the Prophet (Mahmud, 2020) and ARIMA (Kufel, 2020) algorithms which are autoregressive models with a moving average. As a baseline, this study uses the average capacity (AVG) up to the day of testing. The second baseline used in this research is the average window (AVG_WINDOW), where the average capacity for the past days and day, d, is the window size. Table 5.6 summarizes the results of basic time series algorithms against an MPNN GNN and other neural networks such as GAT with Centrality + LSTM model. Mean squared error (MSE) is this study's comparable value and is defined as an error between the predicted value and the actual value in the validation set.

Table 1 shows that Prophet and ARIMA have large mean squared errors, greater than 1. The two baselines, AVG_WINDOW and AVG, show a mean squared error above 1. Table 1 shows that Prophet and ARIMA are not much better than AVG and AVG_WINDOW. The mean squared error drops dramatically with the MPNN model. The MPNN model's error is below 1. Table 5.6 shows the model using GAT and LSTM to evaluate the data. GAT with centrality + LSTM further increases accuracy, as Table 5.4 shows. GAT with centrality + LSTM has the best accuracy values, hence proving centrality can help GAT achieve accuracy improvements. As the results show, GAT with centrality + LSTM improves accurate predictions of capacity within the model, which is mainly

because of its ability in mapping internal connections and dependencies inside the supply chain network.

Model	Mean_Error
PROPHET	2.31457
ARIMA	1.85811
AVG_WINDOW	2.35932
AVG	2.21412
MPNN	0.38272
GConvGRU	0.0796
GCONVLSTM	0.0827
MPNN+LSTM	0.0796
GAT+LSTM	0.0721
GAT+LSTM	0.08
GAT with Centrality + LSTM	0.0712

Table 5.6 - Mean Square Error of Different Prediction Models

Design of experiments based on multi-variate analysis of variance (MANOVA) are set to find the best settings of hyperparameters for the model to reduce computation time and MSE (Figure 5.7). For this study, the hyper-parameters for MPNN+LSTM, GAT+LSTM, and GAT with centrality + LSTM were set by modifying learning rate, hidden layers, and dropout. The experiment trains each model for 1000 epochs and uses a batch setting of 8 graphs. Batching allows for PyTorch geometric to work on graphs in parallel instead of one by one. This study ran 16 experiments using different hyperparameters.

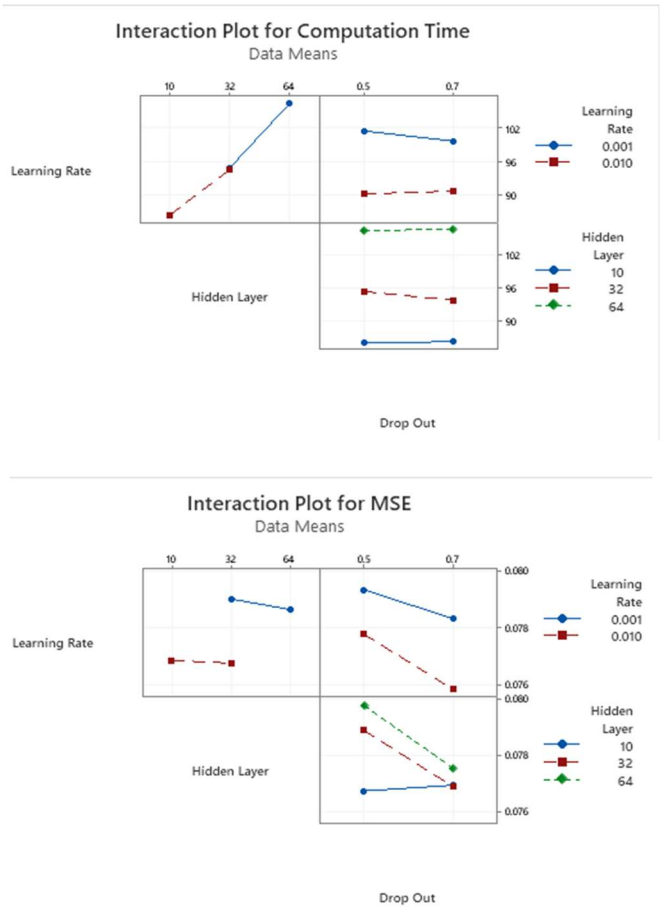

Figure 5.7 - Interaction Plot of GNN Hyperparameters

On average, the mean error of a graph neural network is better than the ARIMA model. ARIMA is a statistical model that uses math equations, data fitting and smoothing to get an approximation of future values based on a trend. The research concludes after viewing the data that a graph neural network can give better predictions with a smaller training set. The ARIMA model requires a large amount of data to generate its prediction but large trends and smoothing affect the results. A graph neural network can train on a small dataset than prediction can be evaluated for correctness. If values are not within the target range. Incremental training is possible with a graph neural network, which means

training the model on a different training set to add knowledge to the model. Incremental training is not possible in an ARIMA model. ARIMA needs all the data every time a model predicts data. Both models will perform better with larger datasets. Time to predict will increase, but a graph neural network can scale by using sampling methods. Sampling methods for ARIMA could affect the quality of the data, which is not true of graph neural network sampling methods. Graph neural networks are a robust prediction model. Graph neural networks are able to handle noise with little to no increase in error values. To prove this statement the research added Gaussian noise to the validation set. The model was not retrained to ignore the noise. The Gaussian noise was added to all nodes, only supplier nodes, only manufacturing nodes, only distributor nodes, and only customer nodes. Figure 5.8 shows the results of the sixteen experiments with noise added.

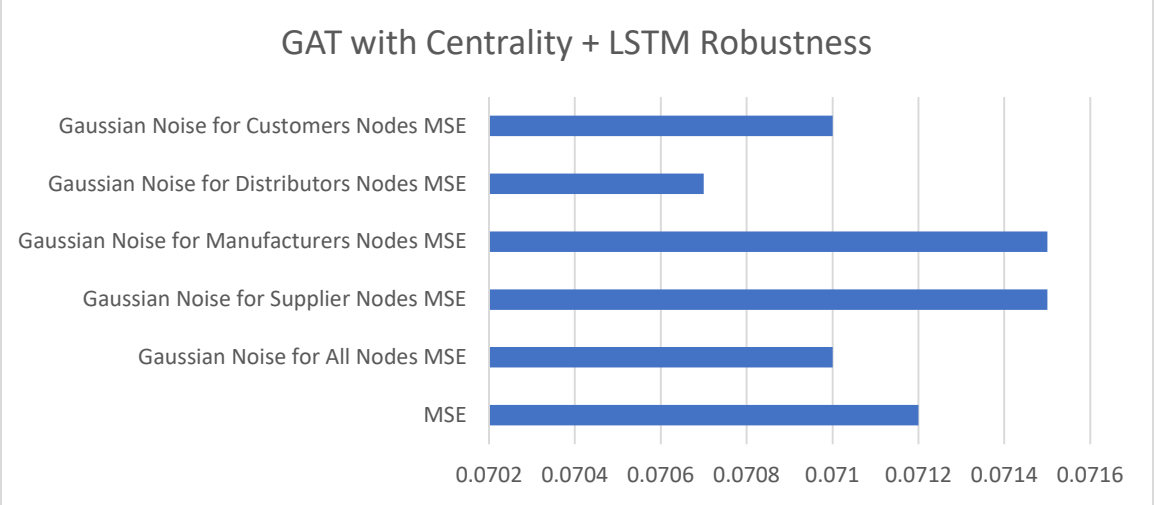

Figure 5.8 - Graph Neural Network Experiments MSE

The hyperparameters are used in these sixteen experiments. The results for little to no change in error. Proving the robustness of the graph neural network model at handling noise in a dataset. The next section will discuss the betweenness and centrality results.

5.3 Betweenness Centrality results

Betweenness centrality measures the importance of a node in a network by understanding how many paths pass through it. The higher the betweenness, the more connects a node has to other nodes. Removing a node with a high betweenness will disrupt the network, break paths, and orphan nodes. Betweenness shows how important a node is to the network overall. The results from the betweenness algorithm on the dataset's graph structure are in figure 5.9.

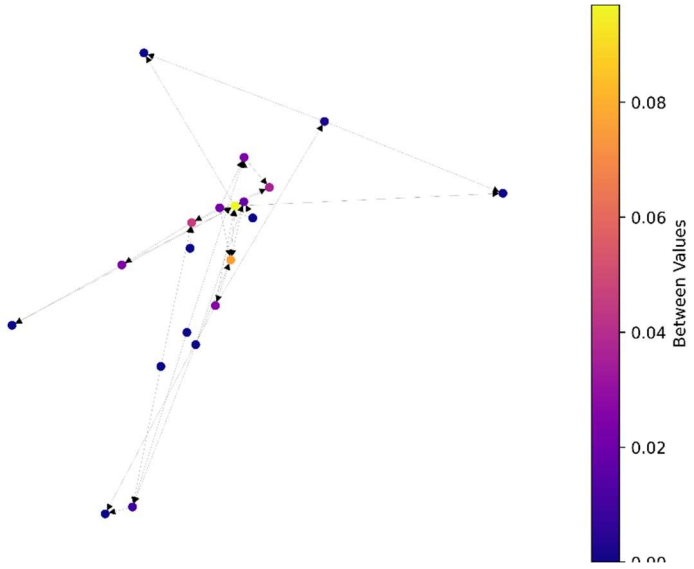

Figure 5.9 - Betweenness Graph for Supply Chain

Start and end nodes have zero betweenness values since no paths cross through them. Zero betweenness also means that these nodes are easily replaceable, since removing them will not cause a disruption in the supply chain. Manufacturers and distributors are more important to keeping the supply chain disruption free in the dataset's graph structure. The dataset's graph is static and does not change over time, so these betweenness values are valid for all time steps. The betweenness values help to understand a node's value to keeping a graph connected and its importance to a graph structure overall.

This research uses three more centrality measures to better understand node importance within the supply chain. Using degree centrality, the research gets the understanding of how many income and outgoing edges are connected to a node. The dataset's degree centrality is shown in figure 5.10.

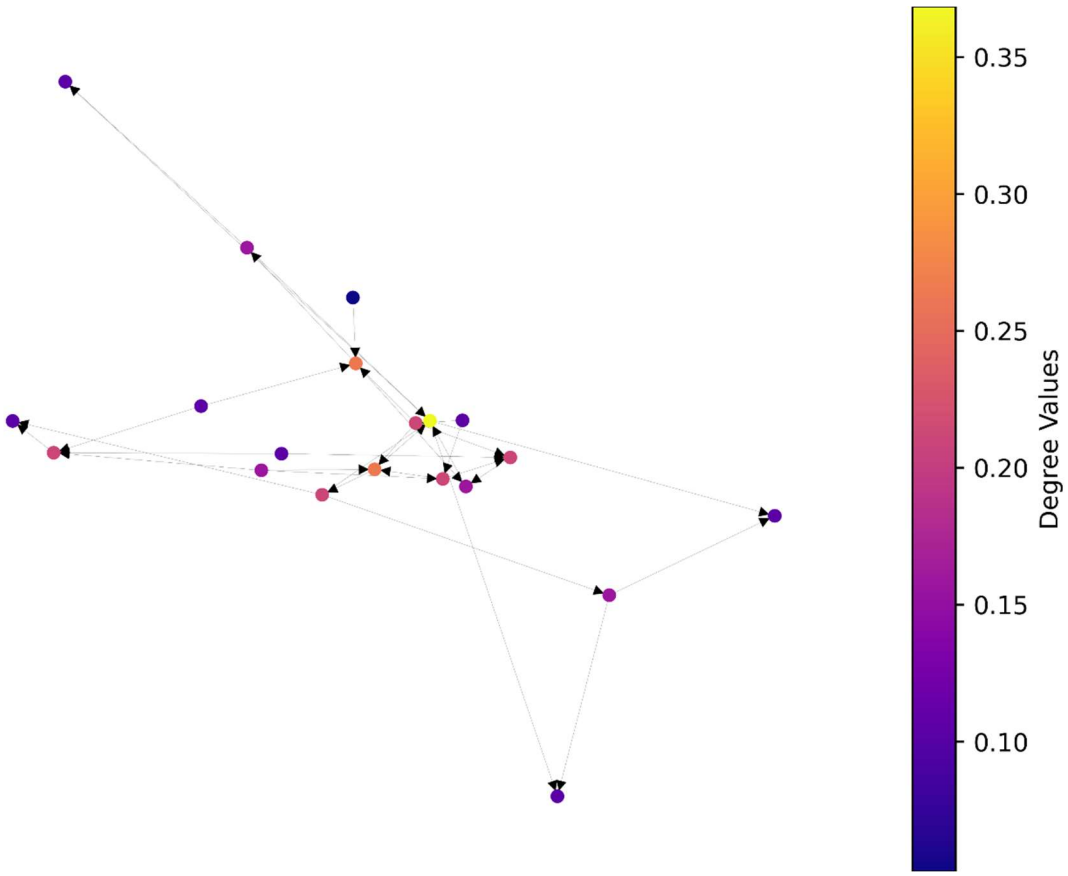

Figure 5.10 - Degree Centrality

Knowing the degree centrality of a node allows the research to understand how a node is contributing to another node. The value is useful for better understanding how much a supplier is supporting the supply chain. If a supplier is supporting two or more nodes, it may be detrimental to the supply chain if that supplier fails to produce products. The idea degree centrality can be generalized by using eigenvector. Eigenvector centralities focus on the node connection to another important node. Eigenvector centrality can be seen as an influence rating. The research applies eigenvector centrality to its dataset resulting in figure 5.11.

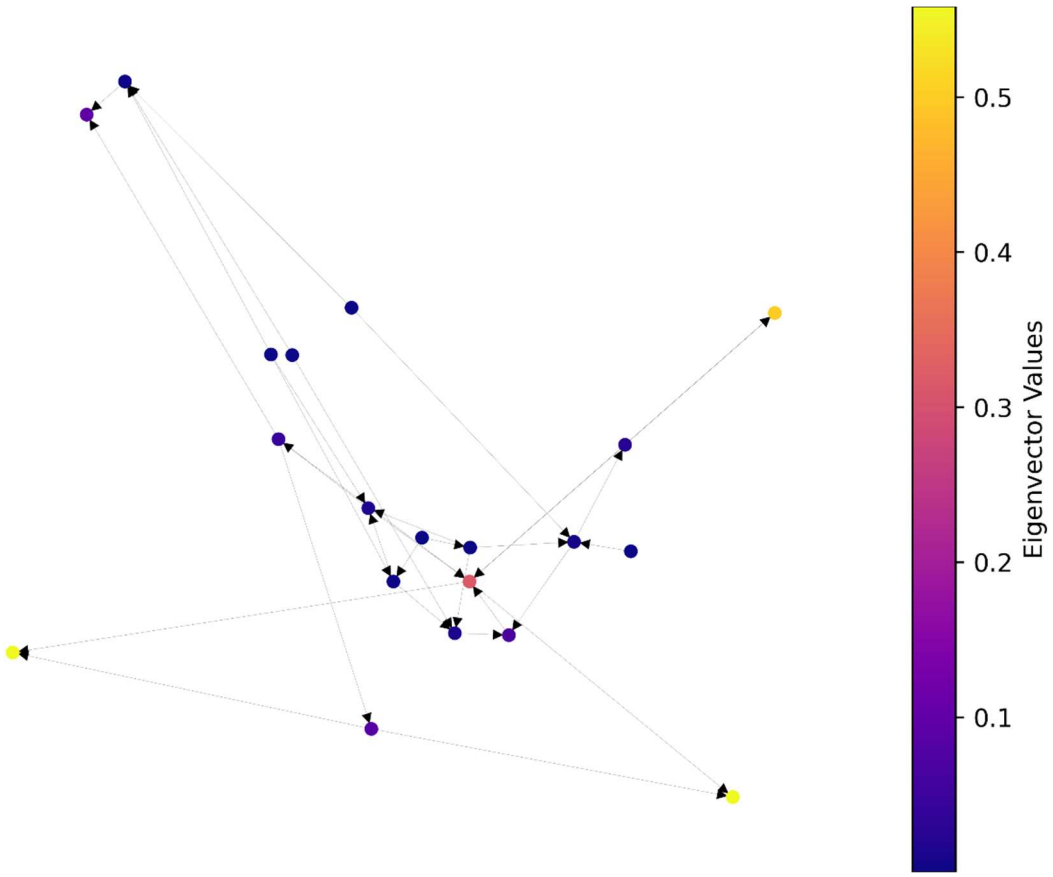

Figure 5.11 - Eigenvector Centrality

Lastly the research computes the Katz centrality similar to eigenvector centrality, but it focusses on short walks between nodes.

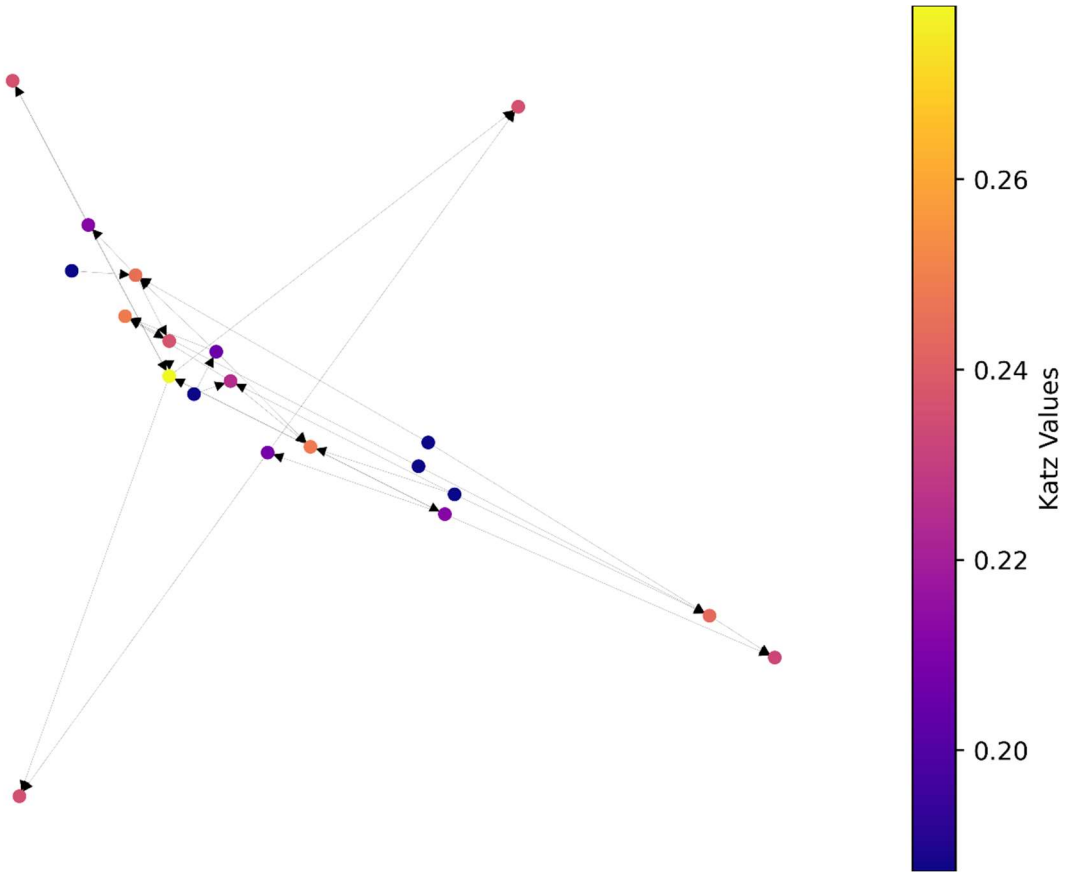

Figure 5.12 - Katz Centrality

	Supplier_001	Supplier_002	Supplier_003	Supplier_004	Supplier_005	
Betweenness	0	0	0	0	0	
Degree	0.05263	0.26316	0.10526	0.21053	0.21053	
Eigen	1.23E-04	7.90E-03	1.23E-04	1.23E-03	2.35E-03	
Katz	1.87E-01	2.45E-01	1.87E-01	2.06E-01	2.25E-01	
	Manufacturer_001	Manufacturer_002	Manufacturer_003	Manufacturer_004	Manufacturer_005	
Betweenness	0.045808967	0.023879142	0.076510721	0.036549708	0.00877193	
Degree	0.10526	0.21053	0.21053	0.10526	0.15789	
Eigen	1.23E-04	1.68E-02	3.46E-03	1.23E-04	1.23E-04	
Katz	1.87E-01	2.49E-01	2.43E-01	1.87E-01	1.87E-01	
	Distributor_001	Distributor_002	Distributor_003	Distributor_004	Distributor_005	Distributor_006
Betweenness	0.024366472	0.026315789	0.096978558	0.002923977	0.021442495	0.019493177
Degree	0.26316	0.15789	0.10526	0.36842	0.21053	0.15789
Eigen	1.68E-02	2.49E-02	5.02E-01	3.16E-01	4.57E-02	8.34E-02
Katz	2.49E-01	2.12E-01	2.36E-01	2.78E-01	2.12E-01	2.08E-01
	Customer_001	Customer_002	Customer_003	Customer_004		
Betweenness	0	0	0	0		
Degree	0.10526	0.10526	0.10526	0.15789		
Eigen	9.79E-02	5.59E-01	5.59E-01	7.05E-02		
Katz	2.33E-01	2.36E-01	2.36E-01	2.37E-01		

Table 5.7 - Centrality Measures for Each Supply Chain Entity

Table 5.7 shows the values from the centrality measures. Clear showing the comments stated above. In the section will discuss a node's influence on a customer node using Bayesian Network belief propagation.

5.4 Bayesian Network Belief Propagation

A Bayesian Network can show how much a node change in state influences another node to change its state. The research uses a Bayesian Network to understand how nodes within a supply chain influence the customer node. The research used Genie Modeler from BayesFusion as the Bayesian Network tool to analyze node influence. The research models the supply chain graph structure in Genie. The research gives each node an operating and failed state. The research assumes the chance a node is in the operating state to be eighty percent of the time, while the research sets a failed state to twenty percent. Setting observation for node in the supply chain one by one shows its influence

on the customer nodes. The customer nodes an excluded since they do not influence themselves. Table 5.8 shows the overall influence a node has on each customer node.

	Customer_001	Customer_002	Customer_003	Customer_004
Supplier_001	0.05	0.01	0.01	0
Supplier_002	0.08	0.09	0.09	0.05
Supplier_003	0	0.01	0.01	0.03
Supplier_004	0.05	0.01	0.01	0.03
Supplier_005	0.03	0.08	0.08	0.08
Manufacturer_001	0.15	0.06	0.06	0.03
Manufacturer_002	0.06	0.08	0.08	0.05
Manufacturer_003	0.08	0.16	0.16	0.12
Manufacturer_004	0.11	0.08	0.08	0.09
Manufacturer_005	0.03	0.03	0.03	0.12
Distributor_001	0.15	0.06	0.06	0.03
Distributor_002	0.08	0.16	0.16	0.12
Distributor_003	0.15	0.16	0.16	0.07
Distributor_004	0.08	0.16	0.16	0.12
Distributor_005	0.08	0.09	0.09	0.05
Distributor_006	0.06	0.11	0.11	0.08

Table 5.8 - Node Influence using a Bayesian Network

The data in table 5.8 shows some correlation to the betweenness data. Nodes with high betweenness also have higher influence on customer nodes. Supplier_001 has no influence over Customer_004 since there is no path between these nodes. Again, suppliers have little influence over customers, meaning changing them may not affect customers that much. Manufacturers and distributors have the highest influence values since they are closer to the customer and supply them directly. The influence value gives the research an idea of how important a node is to a customer. The research only

generates the values one time since the graph structure is static. Figure 5.13 shows a graphical view of Distributor_001 influence over customers.

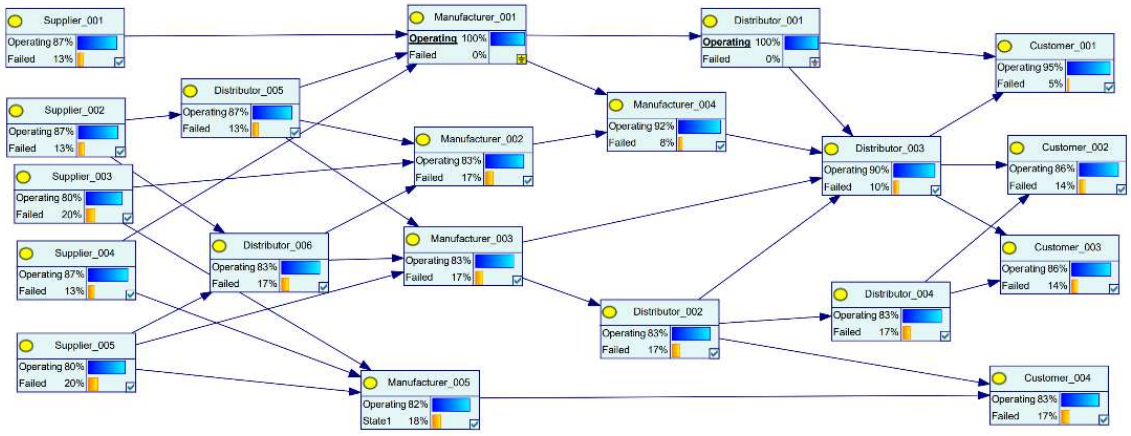

Figure 5.13 Distributor_001 influence graph

5.5 Customer Loss Values

The research builds a loss value by taking the difference between a customer's supply of a product and the normalized demand value. A positive loss means the customer is losing profit. A negative loss means the customer is meeting demand. Demand is actual demand data from the dataset for the last fourteen days. The customer column is the graph neural network prediction of supply for the last fourteen days. The supply chain is a steady flow of products to the customers and doesn't take demand into account. The research is predicting future values based on how the supplier chain is producing products. The research doesn't not adjust for demand that is why the customer capacity is mostly stable compared to the demand and lost values.

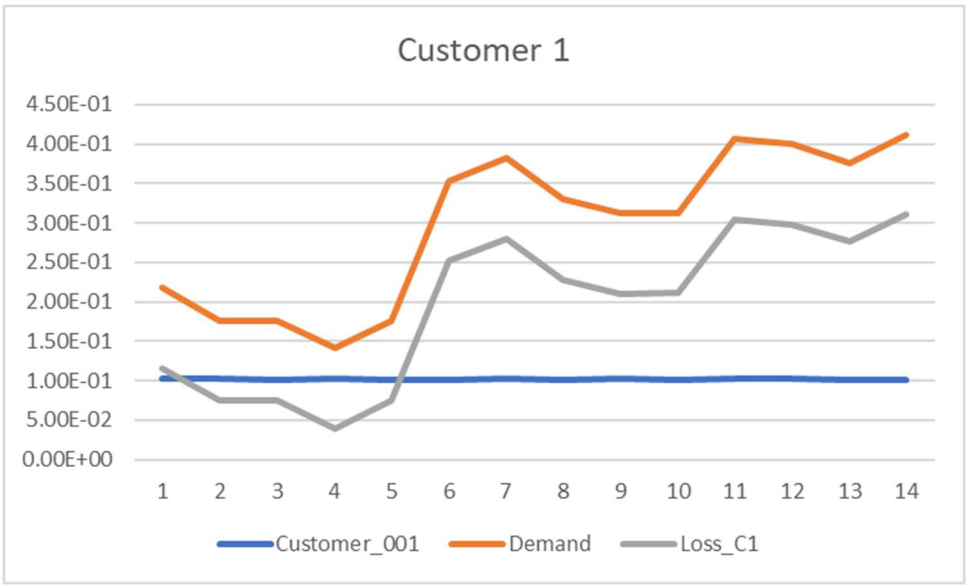

Figure 5.14 Customer 1 Loss Values

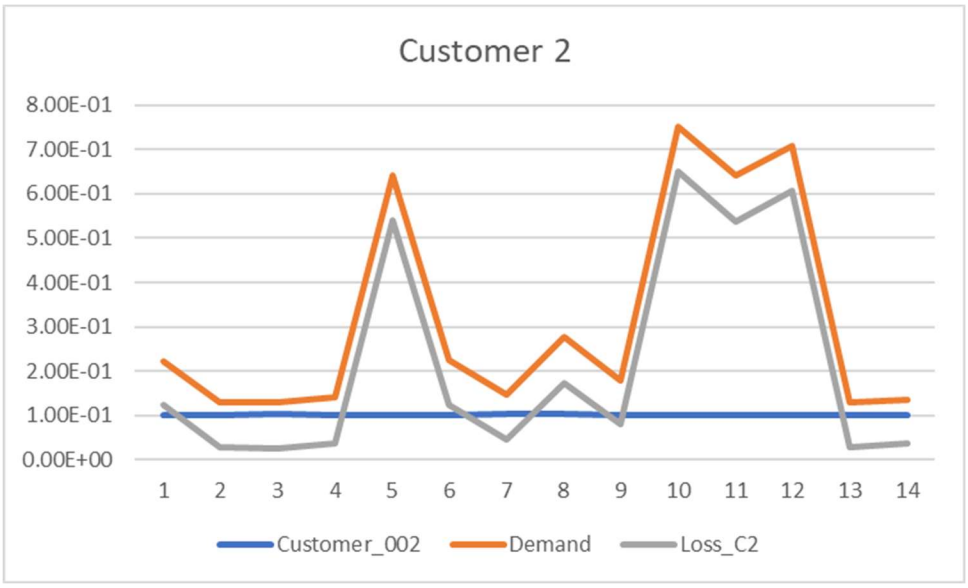

Figure 5.15 Customer 2 Loss Values

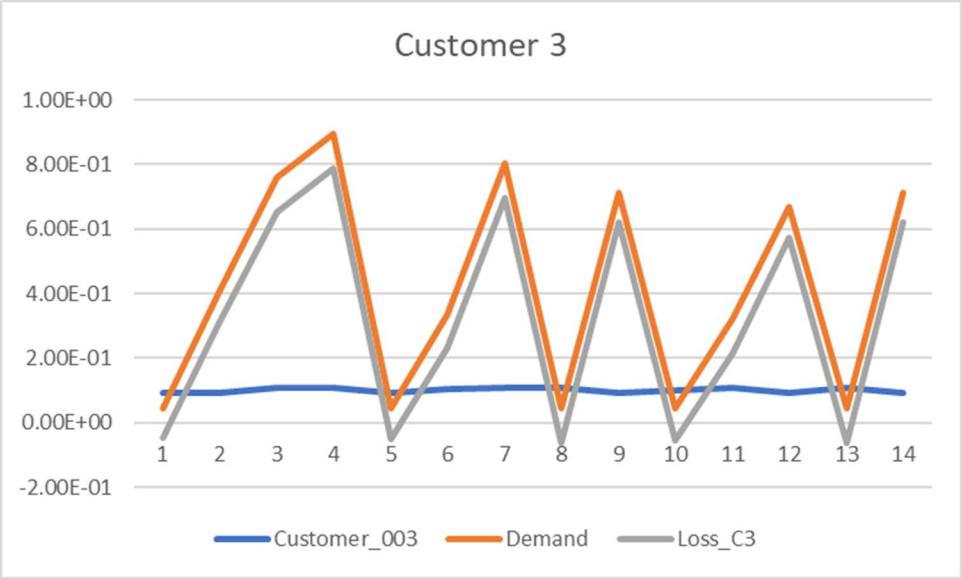

Figure 5.16 Customer 3 Loss Values

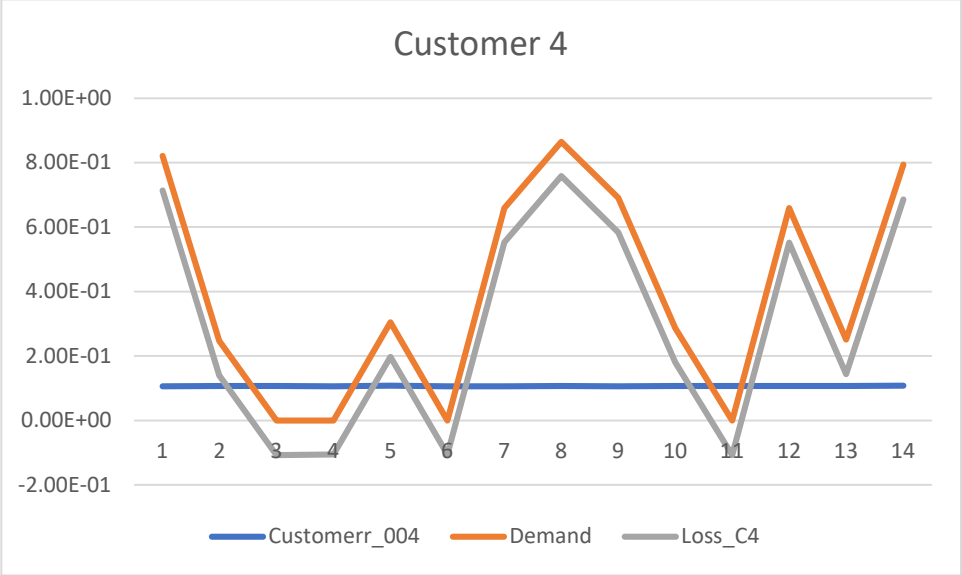

Figure 5.17 Customer 4 Loss Values

In the next section, the research will present risk data. Calculating risk by using all the above data.

5.6 Risk for Each Node

Using the betweenness, influence, loss, and ARIMA capacity value gives the research a risk value for that node. Table 5.9 shows the risk values for each node except for customers because a customer's risk of failure comes from outside of the supply chain. Suppliers have no risk values in this research calculation using between. The reason is that a supplier's failure is outside of the supply chain. Green is not a risk and red is high risk.

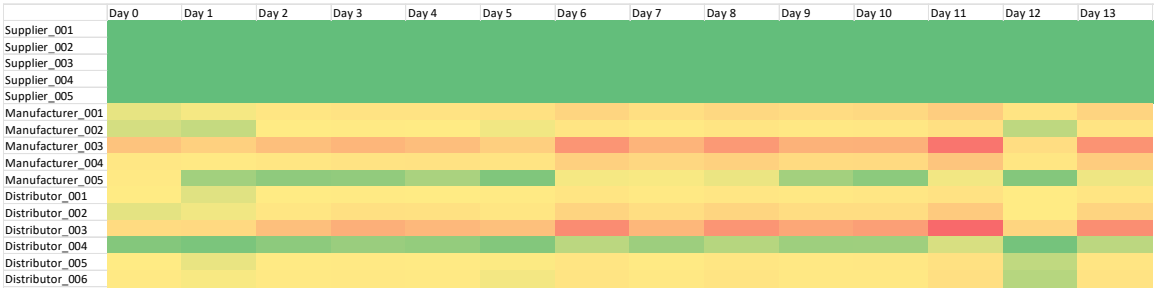

Figure 5.18 Betweenness Risk Values for Nodes

Supply Chain\Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Supplier_001	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Supplier_002	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Supplier_003	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Supplier_004	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Supplier_005	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Manufacturer_001	0.000425	0.000472	0.001032	0.001329	0.001265	0.001513	0.002609	0.001689	0.002274	0.001895	0.002261	0.003301	0.001183	0.002664
Manufacturer_002	0.000365	0.000319	0.000536	0.000667	0.00059	0.00046	0.001124	0.000722	0.001092	0.000828	0.000886	0.001589	0.000296	0.001205
Manufacturer_003	0.004289	0.003098	0.004637	0.005505	0.004752	0.003156	0.008511	0.005635	0.008193	0.005865	0.005954	0.011499	0.001801	0.008744
Manufacturer_004	0.000933	0.000691	0.001059	0.001317	0.001385	0.001173	0.003014	0.002361	0.002942	0.002061	0.002124	0.004039	0.001042	0.00341
Manufacturer_005	0.000723	0.000205	0.000144	0.000152	0.00023	9.34E-05	0.000472	0.000481	0.000438	0.000207	0.000135	0.000461	0.000108	0.00045
Distributor_001	0.000541	0.000405	0.000572	0.0006	0.000536	0.000637	0.001112	0.000728	0.000983	0.000821	0.000979	0.001427	0.000511	0.001152
Distributor_002	0.000416	0.000458	0.001058	0.001563	0.001479	0.001007	0.002709	0.001779	0.002561	0.001826	0.00185	0.003573	0.00056	0.002717
Distributor_003	0.002018	0.002215	0.004614	0.006158	0.005279	0.004505	0.009365	0.00535	0.008501	0.006815	0.007627	0.012558	0.002574	0.009208
Distributor_004	0.000109	8.04E-05	0.000138	0.000178	0.000159	0.000106	0.000285	0.000187	0.00027	0.000193	0.000196	0.000379	5.94E-05	0.000288
Distributor_005	0.000578	0.000434	0.000649	0.000738	0.000613	0.000494	0.001108	0.000683	0.001036	0.000802	0.000874	0.0015	0.000302	0.001124
Distributor_006	0.000728	0.000481	0.000696	0.000807	0.000688	0.000467	0.00123	0.000808	0.001178	0.000851	0.000872	0.001659	0.000269	0.00126

Table 5.9 - Betweenness Risk Values for Nodes

Based on the results, the research concludes the supply chain has a small risk of failure. Manufacturer 2 shows sign of high risk and should be overviewed. Distributor 3

has a high risk value at times and should be reviewed. The research calculates the risk values with degree, eigenvector, and Katz to see what the difference centrality base on a node has to the supply chain.

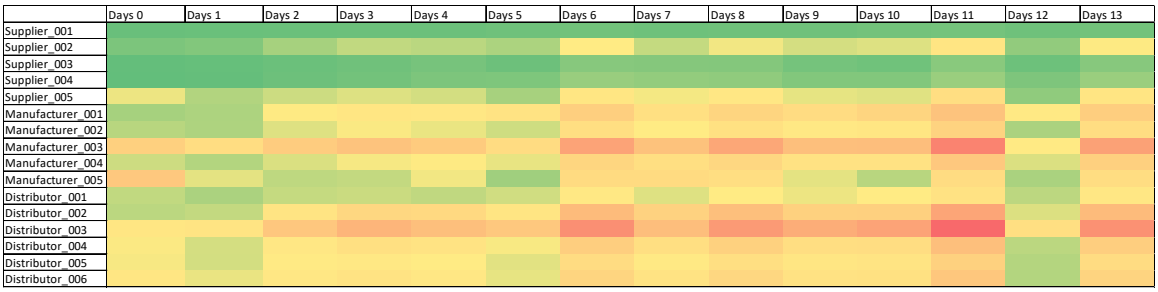

Figure 5.19 - Degree Centrality Risk Calculation

Node\Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Supplier_001	4.95E-05	8.02E-05	0.000151	0.000168	0.000166	0.000301	0.00041	0.00026	0.000348	0.000343	0.000457	0.000538	0.000288	0.000448
Supplier_002	0.000839	0.001065	0.002534	0.003657	0.003374	0.002813	0.006202	0.003796	0.005676	0.004418	0.004799	0.008158	0.00171	0.006125
Supplier_003	-0.00017	-6.56E-05	0.000122	0.000353	0.000627	0.000203	0.001261	0.001197	0.001148	0.00056	0.000355	0.001347	0.000229	0.0013
Supplier_004	-0.00021	-0.0001	0.000197	0.000453	0.000856	0.000929	0.002015	0.001773	0.001657	0.001067	0.001084	0.00207	0.000903	0.002058
Supplier_005	0.005447	0.003046	0.004094	0.004795	0.00438	0.002587	0.007923	0.005763	0.007688	0.005187	0.004931	0.010419	0.001632	0.008246
Manufacturer_001	0.002552	0.002843	0.006142	0.007892	0.007557	0.008986	0.015304	0.009955	0.013299	0.011178	0.013288	0.019244	0.007066	0.015546
Manufacturer_002	0.003304	0.002886	0.004828	0.005999	0.005335	0.004187	0.010061	0.006466	0.009719	0.007456	0.007957	0.014144	0.00275	0.010739
Manufacturer_003	0.015084	0.010904	0.016243	0.019276	0.016728	0.011193	0.029687	0.019645	0.028416	0.020572	0.02085	0.039889	0.006567	0.030369
Manufacturer_004	0.004129	0.003094	0.004698	0.005835	0.006168	0.005233	0.013227	0.010361	0.012861	0.009118	0.00937	0.017649	0.004703	0.014911
Manufacturer_005	0.017557	0.005083	0.003538	0.003724	0.005676	0.002315	0.01145	0.01165	0.010609	0.005086	0.003299	0.011186	0.002708	0.010919
Distributor_001	0.003664	0.002752	0.003838	0.004021	0.003612	0.004263	0.007356	0.004837	0.006486	0.005461	0.006493	0.009385	0.003446	0.007582
Distributor_002	0.003405	0.003753	0.008622	0.01273	0.012107	0.008303	0.021981	0.014428	0.020663	0.014898	0.015068	0.028827	0.004746	0.021947
Distributor_003	0.007932	0.008665	0.017932	0.023906	0.020599	0.017659	0.036185	0.02073	0.032662	0.026443	0.029548	0.048187	0.010291	0.035394
Distributor_004	0.006015	0.004441	0.007572	0.009786	0.008798	0.005913	0.015589	0.01025	0.014704	0.010645	0.010789	0.02064	0.003398	0.015714
Distributor_005	0.005849	0.004384	0.006513	0.007398	0.006177	0.005004	0.011057	0.006822	0.010285	0.008038	0.008749	0.014873	0.003117	0.011166
Distributor_006	0.008042	0.005315	0.007657	0.008874	0.00761	0.005198	0.013475	0.008846	0.012834	0.009376	0.009589	0.018078	0.003072	0.013741

Table 5.10 - Degree Centrality Risk Values

Switching out betweenness with degree centrality gives the research the ability to see that some suppliers may have trouble supplying products to nodes in the supply chain. Supplier 2 risk increases as time goes on. Green is low risk while red is high risk.

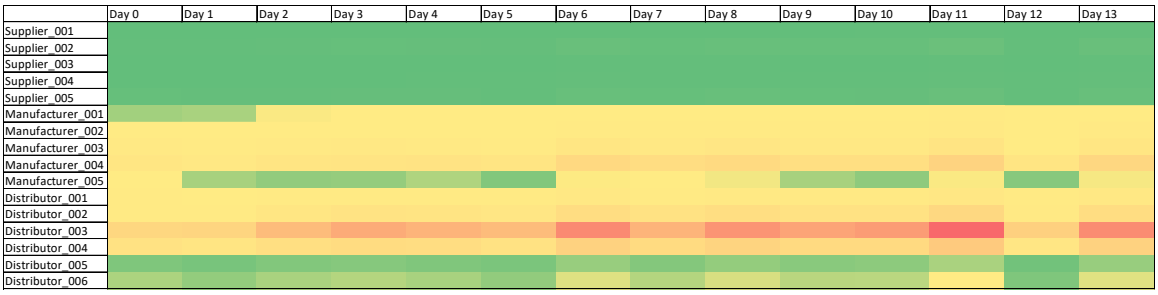

Figure 5.20 - Eigenvector Centrality Risk Calculation

Node\Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Supplier_001	1.16E-07	1.88E-07	3.53E-07	3.95E-07	3.89E-07	7.05E-07	9.60E-07	6.10E-07	8.17E-07	8.05E-07	1.07E-06	1.26E-06	6.76E-07	1.05E-06
Supplier_002	9.84E-07	1.25E-06	2.97E-06	4.29E-06	3.96E-06	3.30E-06	7.27E-06	4.45E-06	6.66E-06	5.18E-06	5.63E-06	9.57E-06	2.01E-06	7.18E-06
Supplier_003	-1.98E-07	-7.69E-08	1.43E-07	4.13E-07	7.36E-07	2.38E-07	1.48E-06	1.40E-06	1.35E-06	6.57E-07	4.17E-07	1.58E-06	2.69E-07	1.52E-06
Supplier_004	-2.52E-07	-1.19E-07	2.32E-07	5.31E-07	1.00E-06	1.09E-06	2.36E-06	2.08E-06	1.94E-06	1.25E-06	1.27E-06	2.43E-06	1.06E-06	2.41E-06
Supplier_005	4.26E-06	2.38E-06	3.20E-06	3.75E-06	3.42E-06	2.02E-06	6.19E-06	4.51E-06	6.01E-06	4.06E-06	3.86E-06	8.15E-06	1.28E-06	6.45E-06
Manufacturer_001	7.66E-05	8.54E-05	0.000184	0.000237	0.000227	0.00027	0.000459	0.000299	0.000399	0.000336	0.000399	0.000578	0.000212	0.000467
Manufacturer_002	0.000263	0.00023	0.000385	0.000478	0.000425	0.000334	0.000802	0.000516	0.000775	0.000595	0.000634	0.001128	0.000219	0.000856
Manufacturer_003	0.000962	0.000696	0.001036	0.00123	0.001067	0.000714	0.001894	0.001253	0.001813	0.001312	0.00133	0.002545	0.000419	0.001937
Manufacturer_004	0.001843	0.001381	0.002097	0.002605	0.002753	0.002336	0.005905	0.004625	0.005741	0.00407	0.004183	0.007879	0.002099	0.006656
Manufacturer_005	0.000288	8.34E-05	5.81E-05	6.11E-05	9.32E-05	3.80E-05	0.000188	0.000191	0.000174	8.35E-05	5.42E-05	0.000184	4.45E-05	0.000179
Distributor_001	0.000579	0.000435	0.000606	0.000635	0.00057	0.000673	0.001162	0.000764	0.001024	0.000862	0.001025	0.001482	0.000544	0.001197
Distributor_002	0.000739	0.000814	0.00187	0.002762	0.002626	0.001801	0.004769	0.00313	0.004483	0.003232	0.003269	0.006254	0.00103	0.004761
Distributor_003	0.006796	0.007423	0.015362	0.02048	0.017648	0.015129	0.031	0.01776	0.027982	0.022654	0.025314	0.041283	0.008817	0.030323
Distributor_004	0.003179	0.002347	0.004002	0.005172	0.00465	0.003125	0.008239	0.005417	0.007771	0.005626	0.005702	0.010908	0.001796	0.008305
Distributor_005	3.43E-05	2.57E-05	3.82E-05	4.34E-05	3.62E-05	2.93E-05	6.48E-05	4.00E-05	6.03E-05	4.71E-05	5.13E-05	8.72E-05	1.83E-05	6.55E-05
Distributor_006	8.96E-05	5.92E-05	8.53E-05	9.89E-05	8.48E-05	5.79E-05	0.00015	9.85E-05	0.000143	0.000104	0.000107	0.000201	3.42E-05	0.000153

Table 5.11 - Eigenvector Centrality Risk Values

Eigenvector centrality and betweenness are similar in risk value when looking at heatmap. The green, low risk, and red, high risk. Seems to match in specific areas. The biggest difference is distributor 5 and 6 have lowering risk in the eigenvector calculation which may be to the have that they don't have a high connection to other nodes hence the eigenvector is lower.

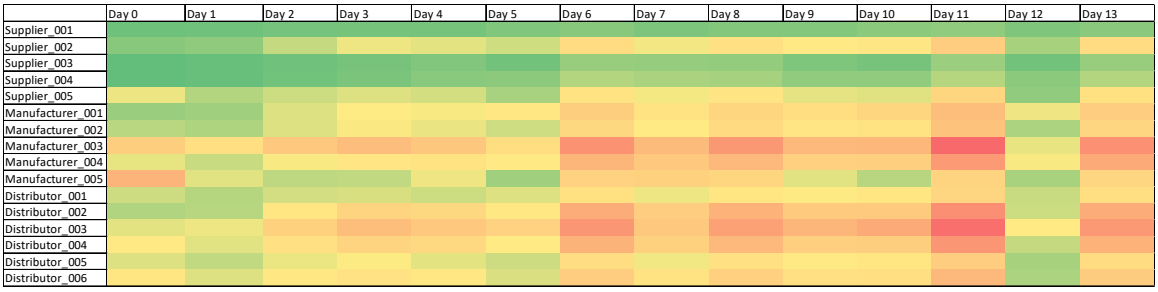

Figure 5.21 - Katz Centrality Risk Calculation

Node\Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Supplier_001	0.000163	0.000266	0.000509	0.000569	0.000558	0.001033	0.001417	0.000889	0.001205	0.001183	0.001585	0.001874	0.00099	0.00156
Supplier_002	0.001446	0.001839	0.004403	0.006361	0.005838	0.004843	0.010841	0.006624	0.009976	0.007684	0.008361	0.014351	0.002886	0.010758
Supplier_003	-0.0003	-0.00011	0.000214	0.000618	0.001089	0.000351	0.002221	0.002113	0.002028	0.000975	0.000622	0.002373	0.000387	0.00229
Supplier_004	-0.00037	-0.00017	0.000334	0.000766	0.001454	0.001595	0.003517	0.003096	0.002895	0.001839	0.001878	0.003619	0.001548	0.003602
Supplier_005	0.006336	0.003529	0.004767	0.005584	0.005072	0.002972	0.009265	0.00675	0.009038	0.006025	0.005739	0.012239	0.001823	0.009678
Manufacturer_001	0.002275	0.002522	0.005519	0.007106	0.006765	0.008094	0.013954	0.009034	0.012161	0.010136	0.012091	0.017657	0.006325	0.014246
Manufacturer_002	0.003804	0.003323	0.005592	0.006954	0.006152	0.004801	0.011717	0.007524	0.011382	0.008639	0.009234	0.016573	0.003083	0.012566
Manufacturer_003	0.013959	0.010082	0.015091	0.017916	0.015466	0.010271	0.0277	0.01834	0.026664	0.019087	0.019376	0.037424	0.005862	0.028458
Manufacturer_004	0.006052	0.004482	0.006866	0.008542	0.008978	0.007608	0.019544	0.01531	0.019077	0.013367	0.013776	0.026191	0.006753	0.022112
Manufacturer_005	0.020036	0.005678	0.00399	0.004205	0.006365	0.002589	0.013077	0.013334	0.012143	0.005728	0.003734	0.012774	0.002994	0.012477
Distributor_001	0.004711	0.003521	0.004974	0.005222	0.004664	0.005538	0.009672	0.006331	0.008554	0.007142	0.00852	0.012419	0.004449	0.01002
Distributor_002	0.003354	0.003693	0.008525	0.012592	0.011912	0.008109	0.021827	0.014335	0.020634	0.01471	0.014903	0.028783	0.004509	0.021887
Distributor_003	0.005784	0.00635	0.013228	0.017654	0.015133	0.012913	0.026846	0.015336	0.024369	0.019535	0.021864	0.035999	0.007379	0.026397
Distributor_004	0.00775	0.005716	0.009794	0.012663	0.011325	0.007556	0.020252	0.013323	0.01921	0.013751	0.013959	0.026962	0.004223	0.020502
Distributor_005	0.005554	0.004168	0.006233	0.007087	0.005887	0.004745	0.010645	0.006557	0.009955	0.0077	0.008396	0.014411	0.002898	0.010802
Distributor_006	0.008399	0.005547	0.008031	0.009312	0.007944	0.005387	0.014198	0.009324	0.013599	0.009824	0.010063	0.019155	0.0031	0.014541

Table 1.12 - Katz Centrality Risk Values

Katz centrality risk calculation gives a riskier outlook for the supply chain since more red and yellow can be seen in the heatmap. Overall figure 5.14 can be seen as a cautious view of the supply chain risk since the research knows the supply chain entities in the dataset are highly connected to each other.

Overall, the supply chain has an acceptable amount of risk at each node except when Katz centrality is used. This research can conclude that the supply chain could use some improvement specifically in the distributor and manufacturer nodes.

5.7 Summary

Overall data generation is very good at predicting future values using GNN. Betweenness of start and end nodes is zero, which makes sense since these nodes do not disrupt connectivity of a supply chain. Things change when different centrality calculations are used instead of betweenness. Influence of start nodes is small on the ends. These two factors exclude suppliers and customers from the risk calculations until betweenness is replaced. The risk values come from the manufacturers and distributors because the dataset's graph structure controls the overall stability of the supply chain.

Meaning nodes in the middle have more influence on the supply chain. The risk on these middle nodes also increases, proving their overall importance to the supply chain in general. In the next chapter, the research will present the conclusion and future work.

CHAPTER SIX

Conclusion

The chapter summarizes this research finding on the information contained in this paper. Drawing a conclusion to problem of data mining large dataset for supply chains. Using the data in the previous chapter, this research will decide if the methods used improve the current set of algorithms and data structures for supply chain risk analysis. Many different areas were viewed by this research meaning that the investigation into the topics presented was not exhaustive. Leaving many different areas of future research open for exploration.

The chapter is broken down into three sections. A summary of the research, future work the research could do on this topic, and a conclusion.

5.1 Summary

The research starts off by giving an understanding of how data has become more interconnected now days. The amount of data is difficult to process using tradition mean like SQL database. The connectiveness and size of the data leads the research to a new style of data structure. The data structure that this research choses is a graph database. The graph database allows the research to model the connections between suppliers, manufacturers, distributors, and customers in a natural way. Supply chain are a network of entities working together to accomplish a common goal. Viewing a supply chain as a network allows the research to apply a graph database to the supply chain dataset. The graph structure allows the research to explore graph neural networks as a way to predict

node values of supply chain entities. The research was not able to find a dataset which gave a network style structure with data values related to risk. This research decided to generate a dataset using different generally available datasets as seed data and trend patterns. Capacity data propagates from suppliers through the supply chain with quality trends affecting manufacturers and delay trends affecting distributors. The dataset is used to supply an ARMIA and GNN function, The GNN is a combination of MPNN for encoding and updating of the graph structure and LSTM to understand node value. MPNN is a generic mechanism for updating node information. The update method explored by this research were convolutional neural networks, graph attention networks, basic update function, and GAT with centrality provided by this research. Both functions can predict future node values. ARMIA does not see the connection between nodes but generates database on node trend. The GNN is able to see the connections between node and generates a holistic value for each node. The research uses centrality to show node importance within the network. Betweenness gives the research shortest path importance which is useful for highly connected nodes. Degree, eigenvector, and Katz centrality all show an influence value of a node. Degree centrality is a basic influence base on connections. Eigenvector and Katz centrality take into account neighboring nodes and their values. The research needs to know how important a node is to an end node, customer node. Bayesian networks give the research this value by calculating its influence on an end node. Last the research looks at the loss of profit at a customer node when a product supplied doesn't meet demand.

All of these pieces are brough together in a single risk calculation that gives the research a risk value for each node in the supply chain. The risk value can show the overall health of the supply chain and where it needs to be improved.

The results chapter shows data gain from the different methods. The data is brought together by the risk calculation function as heatmaps. The heatmaps give a quick view of hot spots with the supply chain. Getting the outcome requires a lot of data.

ARIMA is able to predict future values but the accuracy of the results for specific points is not very good. The reason for this conclusion is the mean squared error is very high. MSE may improve with a large dataset since ARIMA operates better on large datasets. Another issue may be the research dataset does not have many strong trends. ARIMA works best with data that has strong trends. Finally, the interconnectedness of the data could be an issue. ARIMA is viewing each node as an individual and predicting values without outside data. The narrow viewing could be causing the accuracy issue.

Centrality results help the research to understand node importance to a supply chain network. Choosing the right centrality measure requires knowing the context of the node. Betweenness centrality as shown by the results chapter is able to identify hub nodes. Hub nodes in a supply chain are the manufacturers, distributors, or warehouses. Hub nodes are important to a supply chain because many products move through them but betweenness forgets suppliers. Suppliers are important to the nodes within their subgraph. Degree, Eigen, and Katz centrality measures are able to identify these nodes to different degrees. Degree centrality is very basic and only helps to understand how many connections are on a node. Connections may not reveal a node’s true importance to its

neighbor. A better measure is Katz centrality since it builds an Eigenvector for a node in relation to its neighbor. Eigen centrality creates an Eigenvector for each node, but the calculation is based on the whole graph. The results chapter showed that for the research's dataset that Eigen and betweenness centrality are very similar. Leading to Katz centrality as the best measure for the research's supply chain dataset.

Improving the prediction values of the system was investigated by graph neural networks. The inherit graph structure of a supply chain allowed the research to apply different graph neural network pipelines to see which one would work the best. Preparing the dataset for processing required building a graph data structure for holding the information. The data structure was usable by many different graph neural network pipelines which allowed for quick comparison. The comparison metric was mean squared error. After comparing many different GNNs the research decided on a graph attention network. A GAT ability to learn importance of a node during its training is useful in supply chain. The prediction results were good, but the processing time was higher than a graph convolutional neural network. The research was able to make a connection between centrality measures and GAT attention weights values leading to a new style of GAT with centrality. GAT with centrality improved processing time without increasing mean squared error. The Katz centrality was chosen based on the previous centrality results. The overall results were positive showing that GNNs are good for supply chain prediction.

Influence is an important measure within a supply chain. How much an entity influences another entity is difficult to understand. Bayesian networks by design show

how much a decision at one node can affect the preceding nodes. The researcher was not able to apply a basic Bayesian network to the supply chain without decision factors. To solve this problem the supply chain was viewed as an information passing network. The information being passed was uncertain. The uncertainty of a node succeeding at a task can be propagated through its ancestors. The propagation value is equal to a node's influence. Knowing how a node higher up in the supply chain can influence end nodes is helpful for knowing loss.

Each of the different values are brough together in the risk calculation function. The values generated at a node level help identify pain points within the supply chain. A correlation between the risk values, shown as a heatmap in the results chapter, and the structure of the supply chain shows that manufacturers and distributors are important to a supply chain which is common knowledge. Switching out the centrality values in the risk function gave insight into the importance of suppliers. Katz centrality shows that suppliers can affect risk because of their relationship with other nodes. Knowing the pain points in a supply chain allows a supply chain management to build resistance into the overall structure of the supply chain.

In summary this research was able to define a granular risk value for each node, supply chain entity, within a supply chain. The risk value considers the importance and influence of a supply chain entity. Along with the predicted future values from ARIMA and GNN the research created a heatmap highlighting high risk nodes. Predicting future values quickly and accurately required the researcher to build a new style of GAT with centrality. The GAT with centrality improved predictions and contributed a new GNN

pipeline. The research succeeded in its goal of showing that a graph database system has the ability to store, mine, and revel information hind in large dataset that can be difficult from transitional system to handle.

5.2 Future Work

This study presents a wholistic understanding of the data and can make better predictions of future graph data and provides excellent results on the dataset; however, there is room for improvement. More specifically, the model needs a larger dataset to show its ability to scale. Future research could show how the model could handle a dynamic graph, since a supply chain changes shape. The graph structure would change over time which would change node influence. This study graph has a static influence map that does not change over time. Centrality of a node with a graph would also change since the graph would have a dynamic structure at specific time steps. A dynamic graph structure would better represent real world supply chains. Nodes and edges disappearing and reappearing would represent a supply chain entity going offline. The reason for a node going offline could be from outside influences or a node critical failure. The augmentation of the GAT is very simplistic in nature. A future study could use an equation to better incorporate centrality into the attention function. LSTM may not be the best neural network model for predicting future values. Future studies could investigate replacing the LSTM with other models to see if they produce better prediction values.

5.3 Conclusion

Viewing the research as a whole the GNN part shows a very good MSE meaning the model is able to learn the supply chain's behavior from a small training sample. The

risk calculates heatmaps show that the risk is mostly in the middle of the graph which is proved by the structure. The graph structure shows that the middle nodes are more connected to other nodes hence losing them would create a higher chance of failure. The correlation between middle nodes and higher risk values is a logical conclusion to make based on the data provided by the experiments. The research is able to bring multiple pieces together to present a holistic risk view at a node level. The node level risk allows a supply chain manager to adjust levels to lower risk and get the heatmap closer to all green. A green heatmap symbolizes a supply chain with little to no risk.

REFERENCES

Abbasi, B., Babaei, T., Hosseinifard, Z., Smith-Miles, K., & Dehghani, M. (2020). Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management. Computers & Operations Research, 119, 104941.

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. Advances in neural information processing systems, 29.

Aziz, A., Kosasih, E. E., Griffiths, R. R., & Brintrup, A. (2021). Data considerations in graph representation learning for supply chain networks. arXiv preprint arXiv:2107.10609.

Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. (2021). Graph neural networks with convolutional arma filters. IEEE transactions on pattern analysis and machine intelligence, 44(7), 3496-3507.

Binkert, Peter J. (2004). Language and Man. The Langtech Corporation

Bizer, C. (2003). D2r map-a database to rdf mapping language.

Bizer, C., & Seaborne, A. (2004, November). D2RQ-treating non-RDF databases as virtual RDF graphs. In Proceedings of the 3rd international semantic web conference (ISWC2004) (Vol. 2004). Hiroshima: Springer.

Blackhurst, J., Rungtusanatham, M. J., Scheibe, K., & Ambulkar, S. (2018). Supply chain vulnerability assessment: A network based visualization and clustering analysis approach. Journal of Purchasing and Supply Management, 24(1), 21-30.

Bonacich, P. (1987). Power and centrality: A family of measures. American journal of sociology, 92(5), 1170-1182.

Bordawekar, R., Bandyopadhyay, B., & Shmueli, O. (2017). Cognitive database: A step towards endowing relational databases with artificial intelligence capabilities. arXiv preprint arXiv:1712.07199.

Borgatti, S. P. (2005). Centrality and network flow. Social networks, 27(1), 55-71.

Borgelt, C., & Kruse, R. (2002). Graphical models: methods for data analysis and mining. John Wiley & Sons, Inc..

Castillo, E., Gutiérrez, J. M., Hadi, A. S., Castillo, E., Gutiérrez, J. M., & Hadi, A. S. (1997). Learning bayesian networks. Expert systems and probabilistic network models, 481-527.

Chen, D., Sain, S. L., & Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. Journal of Database Marketing & Customer Strategy Management, 19, 197-208.

Chen, J., Wang, X., & Xu, X. (2022). GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. Applied Intelligence, 1-16.

Chen, L., Dui, H., & Zhang, C. (2020). A resilience measure for supply chain systems considering the interruption with the cyber-physical systems. Reliability engineering & system safety, 199, 106869.

Chen, Z., Chen, F., Lai, R., Zhang, X., & Lu, C. T. (2018). Rational neural networks for approximating jump discontinuities of graph convolution operator. arXiv preprint arXiv:1808.10073.

Codd, E. F. (1972). Relational completeness of data base sublanguages (pp. 65-98). IBM Corporation.

Codd, E. F. (1983). A relational model of data for large shared data banks. Communications of the ACM, 26(1), 64-69.

Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. Psychological review, 82(6), 407.

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. Journal of verbal learning and verbal behavior, 8(2), 240-247.

Collins, A. M., & Quillian, M. R. (1970). Does category size affect categorization time?. Journal of verbal learning and verbal behavior, 9(4), 432-438.

Danks, D. (2014). Unifying the mind: Cognitive representations as graphical models. Mit Press.

De Virgilio, R., Maccioni, A., & Torlone, R. (2013, June). Converting relational to graph databases. In First International Workshop on Graph Data Management Experiences and Systems (pp. 1-6).

De Virgilio, R., Maccioni, A., & Torlone, R. (2014). Model-driven design of graph databases. In Conceptual Modeling: 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings 33 (pp. 172-185). Springer International Publishing.

De Virgilio, R., Maccioni, A., & Torlone, R. (2014, March). R2G: a Tool for Migrating Relations to Graphs. In EDBT (Vol. 2014, pp. 640-643).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.

Ebert-Uphoff, I. (2007). Measuring connection strengths and link strengths in discrete Bayesian networks. Georgia Institute of Technology.

Ebert-Uphoff, I. (2009). Tutorial on how to measure link strengths in discrete Bayesian networks. Georgia Institute of Technology.

Fearnhead, P., & Rigaill, G. (2019). Changepoint detection in the presence of outliers. Journal of the American Statistical Association, 114(525), 169-183.

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428.

Fisher, M., Blace, R., Hebeler, J., & Perez-Lopez, A. (2011). Semantic web programming. John Wiley & Sons.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. Sociometry, 35-41.

Gasteiger, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In International conference on machine learning (pp. 1263-1272). PMLR.

Glycolysis. (2023, February 20). In Wikipedia. https://en.wikipedia.org/wiki/Glycolysis

Goertzel, B. (2009, June). OpenCogPrime: A cognitive synergy based architecture for artificial general intelligence. In 2009 8th IEEE International Conference on Cognitive Informatics (pp. 60-68). IEEE.

Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864).

Gudivada, V. N., Irfan, M. T., Fathi, E., & Rao, D. L. (2016). Cognitive analytics: Going beyond big data analytics and machine learning. In Handbook of statistics (Vol. 35, pp. 169-205). Elsevier.

Hagberg, A., Swart, P., & S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hantula, D. (2016). Models for storing relationships: Relational vs. Graph Databases.

Holst, A. (2021). Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. *Statista, June*.

Hosseini, S., & Ivanov, D. (2020). Bayesian networks for supply chain risk, resilience and ripple effect analysis: A literature review. Expert systems with applications, 161, 113649.

Husna, A., Amin, S. H., & Shah, B. (2021). Demand forecasting in supply chain management using different deep learning methods. In Demand forecasting and order planning in supply chains and humanitarian logistics (pp. 140-170). IGI Global.

Isufi, E., Loukas, A., Simonetto, A., & Leus, G. (2016). Autoregressive moving average graph filtering. IEEE Transactions on Signal Processing, 65(2), 274-288.

Johnson, R., & Zhang, T. (2007). On the effectiveness of Laplacian normalization for graph semi-supervised learning. Journal of Machine Learning Research, 8(7).

Kara, M. E., Fırat, S. Ü. O., & Ghadge, A. (2020). A data mining-based framework for supply chain risk management. Computers & Industrial Engineering, 139, 105570.

Katz, L. (1953). A new status index derived from sociometric analysis. Psychometrika, 18(1), 39-43.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kosasih, E. E., Margaroli, F., Gelli, S., Aziz, A., Wildgoose, N., & Brintrup, A. (2022). Towards knowledge graph reasoning for supply chain risk management using graph neural networks. International Journal of Production Research, 1-17.

Kufel, T. (2020). ARIMA-based forecasting of the dynamics of confirmed Covid-19 cases for selected European countries. Equilibrium. Quarterly Journal of Economics and Economic Policy, 15(2), 181-204.

Kuznetsov, S. D., & Poskonin, A. V. (2014). NoSQL data management systems. Programming and Computer Software, 40, 323-332.

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 67(1), 97-109.

Li, C., Zhang, J., Wang, Y., & Liu, X. (2022). CAGAT: centrality-adjusted graph attention network for active scientific talent discovery. Personal and Ubiquitous Computing, 1-8.

Li, Q., Wu, X. M., Liu, H., Zhang, X., & Guan, Z. (2019). Label efficient semi-supervised learning via graph filtering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9582-9591).

Lockamy III, A., & McCormack, K. (2012). Modeling supplier risks using Bayesian networks. Industrial Management & Data Systems, 112(2), 313-333.

Loukas, A., Simonetto, A., & Leus, G. (2015). Distributed autoregressive moving average graph filters. IEEE Signal Processing Letters, 22(11), 1931-1935.

Mahmud, S. (2020). Bangladesh COVID-19 daily cases time series analysis using facebook prophet model. Available at SSRN 3660368.

Miller, E. (1998). An introduction to the resource description framework. D-lib Magazine.

Minsky, M. (1974). A framework for representing knowledge.

Minsky, M. (1974). A framework for representing knowledge.

Nakatani, J., Tahara, K., Nakajima, K., Daigo, I., Kurishima, H., Kudoh, Y., ... &

Moriguchi, Y. (2018). A graph theory-based methodology for vulnerability assessment of supply chains using the life cycle inventory database. Omega, 75, 165-181.

Negre, C. F., Morzan, U. N., Hendrickson, H. P., Pal, R., Lisi, G. P., Loria, J. P., ... &

Batista, V. S. (2018). Eigenvector centrality for characterization of protein allosteric pathways. Proceedings of the National Academy of Sciences, 115(52), E12201-E12208.

Neves, J. L., & Bordawekar, R. (2018). Demonstrating ai-enabled sql queries over relational data using a cognitive database. Knowl. Discov. Data Min.

Newman, M. E. J. (2010). Networks: an introduction.

Nuss, P., Graedel, T. E., Alonso, E., & Carroll, A. (2016). Mapping supply chain risk by network analysis of product platforms. Sustainable Materials and Technologies, 10, 14-22.

Ojha, R., Ghadge, A., Tiwari, M. K., & Bititci, U. S. (2018). Bayesian network modelling for supply chain risk propagation. International Journal of Production Research, 56(17), 5795-5819.

Panagopoulos, G., Nikolentzos, G., & Vazirgiannis, M. (2021, May). Transfer graph neural networks for pandemic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 6, pp. 4838-4845).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural networks, 12(1), 145-151.

Quillian, M. R. (1967). Word concepts: A theory and simulation of some basic semantic capabilities. Behavioral science, 12(5), 410-430.

Quillian, M. R. (1969). The teachable language comprehender: A simulation program and theory of language. Communications of the ACM, 12(8), 459-476.

Richens, R. H. (1956). General program for mechanical translation between any two languages via an algebraic interlingua. Report on research: Cambridge Language Research Unit, Mechanical Translation, 3.

Richens, R. H. (1956). Preprogramming for mechanical translation. Mech. Transl. Comput. Linguistics, 3(1), 20-25.

Robinson, I., Webber, J., & Eifrem, E. (2015). Graph databases: new opportunities for connected data. " O'Reilly Media, Inc.".

Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., ... & Sarkar, R. (2021, October). Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 4564-4573).

Sanyal, J., & New, J. (2013, May). Simulation and big data challenges in tuning building energy models. In 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES) (pp. 1-6). IEEE.

Saumier, D., & Chertkow, H. (2002). Semantic memory. Current neurology and neuroscience reports, 2(6), 516-522.

Schank, R. C., & Abelson, R. P. (2013). Scripts, plans, goals, and understanding: An inquiry into human knowledge structures. Psychology press.

Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25 (pp. 362-373). Springer International Publishing.

Sequeda, J. F., Priyatna, F., & Villazón-Terrazas, B. (2012, November). Relational Database to RDF Mapping Patterns. In WOP.

Silva, N., Ferreira, L. M. D., Silva, C., Magalhães, V., & Neto, P. (2017). Improving supply chain visibility with artificial neural networks. Procedia Manufacturing, 11, 2083-2090.

Soberanis, I. E. D. (2010). An extended Bayesian network approach for analyzing supply chain disruptions (Doctoral dissertation, University of Iowa).

Sowa, J. F. (1992). Semantic networks. Encyclopedia of artificial intelligence, 2, 1493-1511.

Suiter, M. C., & Taylor, K. G. (2016). Resources for economic educators from the Federal Reserve Bank of St. Louis. The Journal of Economic Education, 47(1), 71-75.

Taheriyan, M., Knoblock, C. A., Szekely, P., & Ambite, J. L. (2016). Learning the semantics of structured data sources. Journal of Web Semantics, 37, 152-169.

Tan, S. C., & Lau, J. P. S. (2013, December). Time series clustering: A superior alternative for market basket analysis. In Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013) (pp. 241-248). Singapore: Springer Singapore.

Tan, W. J., Zhang, A. N., & Cai, W. (2019). A graph-based model to measure structural redundancy for supply chain resilience. International Journal of Production Research, 57(20), 6385-6404.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015, May). Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web (pp. 1067-1077).

Thuy, P. T. T., Thuan, N. D., Han, Y., Park, K., & Lee, Y. K. (2014, January). RDB2RDF: completed transformation from relational database into RDF ontology. In Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (pp. 1-7).

Tulving, E. (2002). Episodic memory: From mind to brain. Annual review of psychology, 53(1), 1-25.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Wagner, S. M., & Neshat, N. (2010). Assessing the vulnerability of supply chains using graph theory. International journal of production economics, 126(1), 121-129.

Wieland, A., & Wallenburg, C. M. (2012). Dealing with supply chain risks: Linking risk management practices and strategies to performance. International journal of physical distribution & logistics management.

Willems, S. P. (2008). Data set—real-world multiechelon supply chains used for inventory optimization. Manufacturing & service operations management, 10(1), 19-23.

World Wide Web Consortium. (2012). OWL 2 web ontology language document overview.

World Wide Web Consortium. (2014). RDF 1.1 Primer.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019, May). Simplifying graph convolutional networks. In International conference on machine learning (pp. 6861-6871). PMLR.

Wu, T., & Blackhurst, J. (Eds.). (2009). Managing supply chain risk and vulnerability: tools and methods for supply chain decision makers (pp. 37-38). London: Springer.

Wu, X. M., Li, Z., So, A., Wright, J., & Chang, S. F. (2012). Learning with partially absorbing random walks. Advances in neural information processing systems, 25.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. arXiv preprint arXiv:1810.00826.