MODELING AND SOLVING THE OUTSOURCING RISK MANAGEMENT PROBLEM IN MULTI-ECHELON SUPPLY CHAINS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Industrial Engineering

by

Arian Nahangi

June 2021

COMMITTEE MEMBERSHIP

TITLE:	Modeling and Solving the Outsourcing Risk Management Problem in Multi-Echelon Supply Chains
AUTHOR:	Arian Nahangi
DATE SUBMITTED:	June 2021
COMMITTEE CHAIR:	Mohamed Awwad, Ph.D. Assistant Professor of Industrial Engineering
COMMITTEE MEMBER:	Reza Pouraghabagher, Ph.D. Professor of Industrial Engineering
COMMITTEE MEMBER:	Ahmed Deif, Ph.D. Associate Professor of Operation and Supply Chain Management

ABSTRACT

Modeling and Solving the Outsourcing Risk Management Problem in Multi-Echelon Supply Chains

Arian Nahangi

Worldwide globalization has made supply chains more vulnerable to risk factors, increasing the associated costs of outsourcing goods. Outsourcing is highly beneficial for any company that values building upon its core competencies, but the emergence of the COVID-19 pandemic and other crises have exposed significant vulnerabilities within supply chains. These disruptions forced a shift in the production of goods from outsourcing to domestic methods.

This paper considers a multi-echelon supply chain model with global and domestic raw material suppliers, manufacturing plants, warehouses, and markets. All levels within the supply chain network are evaluated from a holistic perspective, calculating a total cost for all levels with embedded risk. We formulate the problem as a mixed-integer linear model programmed in Excel Solver linear to solve smaller optimization problems. Then, we create a Tabu Search algorithm that solves problems of any size. Excel Solver considers three small-scale supply chain networks of varying sizes, one of which maximizes the decision variables the software can handle. In comparison, the Tabu Search program, programmed in Python, solves an additional ten larger-scaled supply chain networks. Tabu Search's capabilities illustrate its scalability and replicability.

A quadratic multi-regression analysis interprets the input parameters (iterations, neighbors, and tabu list size) association with total supply chain cost and run time. The analysis shows iterations and neighbors to minimize total supply chain cost, while the interaction between iterations x neighbors increases the run time exponentially. Therefore, increasing the number of iterations and neighbors will increase run time but provide a more optimal result for total supply chain cost. Tabu Search's input parameters should be set high in almost every practical case to achieve the most optimal result.

This work is the first to incorporate risk and outsourcing into a multi-echelon supply chain, solved using an exact (Excel Solver) and metaheuristic (Tabu Search) solution methodology. From a practical case, managers can visualize supply chain networks of any size and variation to estimate the total supply chain cost in a relatively short time. Supply chain managers can identify suppliers and pick specific suppliers based on cost or risk. Lastly, they can adjust for risk according to external or internal risk factors.

Future research includes expanding the supply chain network design, adding parts, and considering scrap or defective products. In addition, one could incorporate a multi-product dynamic planning horizon supply chain. Overall, considering a hybrid method combining Tabu Search with genetic algorithms, particle swarm optimization, simulated annealing, CPLEX, GUROBI, or LINGO, could provide better and faster results.

Keywords: Multi-echelon supply chain, globalization, supply chain risk management, outsourcing, linear program, multi-regression, Tabu Search.

ACKNOWLEDGMENTS

I am grateful for Dr. Mohamed Awwad's service as my advisor providing valuable guidance in this paper's construction.

Also, I would like to acknowledge Terry Wambolt for his aid in programming the Tabu Search algorithm presented in this paper.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1. INTRODUCTION

1.1 Outline

The following chapters organize this paper:

CHAPTER 1: provides background on supply chain management, supply chain risks, and outsourcing, as well as the motivation and simplified contributions for this paper.

CHAPTER 2: provides a detailed literature review with referenced information on supply chain risk management, supply chain network design, outsourcing, and metaheuristics.

CHAPTER 3: presents the problem description, key assumptions, and model formulation for the single-objective mixed-integer linear program solved in Excel Solver for small problem instances.

CHAPTER 4: presents a Tabu Search approach used to solve larger problem instances with varying input parameters.

CHAPTER 5: presents a statistical analysis of the Tabu Search results to identify significant predictors for association with total supply chain cost and run time.

CHAPTER 6: presents conclusions, contributions, and recommendations for future work.

APPENDICES: exhibits Python code used internally for this paper.

1.2 Background

The following section provides background information on supply chain networks, supply chain risks, and outsourcing companies.

1.2.1 Supply Chain Networks

A typical multi-echelon supply chain contains levels of suppliers, manufacturers, distributors, retailers, and customers. More specifically, products, information, or funds move between levels over a planning horizon. Modern supply chain network sizes are increasing rapidly due to globalization and the rapid growth of global economies. Companies rely on global supply chain models to meet demand, increase customer value, improve responsiveness, track financials, and establish a quality network. For example, a multi-national company contains different supply chain levels (suppliers, plants, distribution centers, retailers, and customers) worldwide that interconnect into one cohesive system (Ravindran & Warsing, 2017). Figure 1-1 shows an example of a multi-echelon supply chain network.

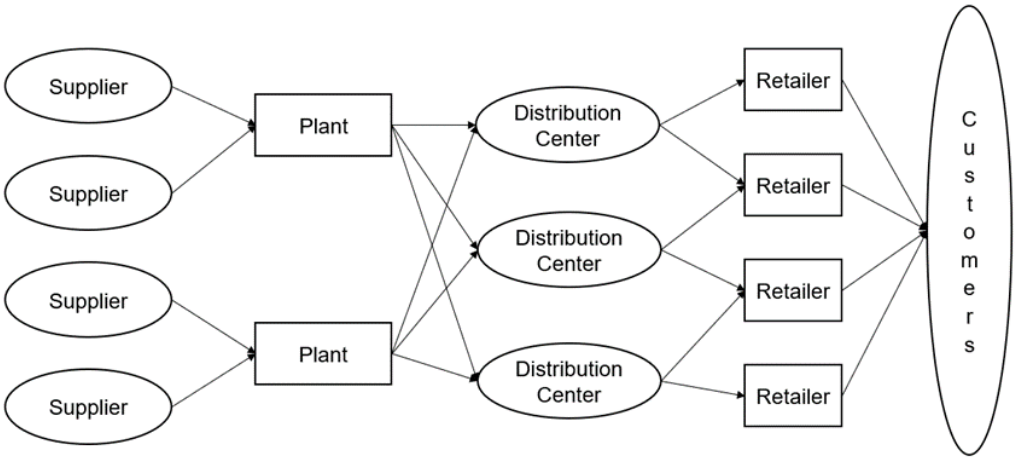

Figure 1-1: Typical Supply Chain (adopted from Ravindran & Warsing, 2017)

Multi-echelon supply chains reduce costs and minimize risks from a holistic perspective. Multi-echelon supply chains raise complexity as they take each level within the supply chain and evaluate the flow of products, information, or funds between them (Shahraki & Sharifi, 2019). Globalization has been a significant problem for product complexity and high service demands for businesses. Still, perhaps the biggest issue is their heavy reliance on multiple layers of suppliers and distribution points and outsourced manufacturing. This nature is highly realistic as supply chain managers can make critical decisions for the entire supply chain rather than each facility. When each facility tries to optimize its own decisions with little regard to the impact of those decisions on other parts of the supply chain, the overall supply chain ends up having high inventory levels and low inventory turns. Supply chain managers minimize costs with a multi-echelon approach by identifying problems early for the entire supply chain network rather than each specific node.

1.2.2 Supply Chain Risks

Supply chain analysts and managers are always looking for ways to reduce costs. Increasing quantity and quality continually increases costs along with embedded risks in the supply chain. Embedded risks tend to hide well in every supply chain, surfacing with great uncertainty. Embedded risks pose numerous problems as they are difficult to pinpoint and quantify to understand their impact on the supply chain. As a result, supply chain management is crucial to increase logistical efficiency and reduce risk factor impact.

Supply chain management spends time and other resources to identify risks within a supply chain. Each risk comes from a source with its probability and impact. According to Ravindran and Warsing (2019), risks occur internally or externally in supply chain systems. Externally, risks stem from the suppliers, customers, globality of the business, or natural events; internally, risks stem from human resources, technology, management, production, finance, or transportation. Overall, these risks are evaluated based upon their probability of occurrence and their impact on the supply chain (occurrence/impact). Risk events with high probability and high impact require the most immediate attention for risk intervention. An example of a high/high risk would be losing critical suppliers or product recalls due to quality issues. On the other hand, risk events with low probability and low impact still require some attention but are not as important. An example of a low/low risk would be health and safety violations or equipment breakdowns. Most risk events contain a mix of the two parameters. These risk events require medium monitoring and attention but can still be quite dangerous due to their ability to stack easily with high frequency. An example of a high/low risk would be a blizzard or a logistics provider's failure. An example of a low/high risk would be flooding, hurricanes, tornados, union and labor problems, or a new competitor in the market.

1.2.3 Outsourcing

Supply chains worldwide are becoming more vulnerable due to the various aspects that can go wrong at any time. Globalization increased supply chain vulnerability in companies starting in the 1990s. According to the United Nations Economic and Social Council, "Globalization refers to the increasing interdependence of world economies as a

result of the growing scale of cross-border trade of commodities and services, the flow of international capital and wide and rapid spread of technologies" (Shangquan, 2000, p. 3). Companies prioritize profits and growth. Therefore, today's companies are shifting towards outsourcing, offshoring, long-term contracts, and relationships with just a few suppliers. These strategies have proven to reduce supply chain costs by pinpointing issues that cause risk vulnerability in supply chains.

Consider a business that needs raw material from a supplier. The company will consider many supplier options and evaluate the cost and risk of obtaining the product. Generally, global suppliers offer lower-cost products, but they may suffer from quality due to the transportation risk during shipping. On the other hand, domestic suppliers provide more expensive products but are more reliable when it comes to transportation uncertainties (Olson et al., 2011).

1.3 Motivation

During the last two decades, companies have witnessed the emergence of a globally competitive environment with manufacturing changes, crumbling international barriers, and increased use of information technologies. An example of a global company would be Apple. Apple has chosen to outsource its engineering work to India and outsource its manufacturing duties to China (Kasyanenko, 2019). Apple does this to reduce costs, increase core function control, and identify future solutions. Overall, outsourcing supply chain processes enables companies to focus on what makes them great by efficiently using their time, energy, and resources to maintain their core competencies. By

offloading most manufacturing work to other areas, Apple can put more time into innovating new products to continue growing.

With the rise of the COVID-19 pandemic, supply chains around the world took a significant hit. For example, multiple national lockdowns stopped the flow of raw materials and finished goods in external parts of the world, disrupting manufacturing. COVID-19 brought to light many of the risk factors associated with the supply chain. It illustrated how many companies are not fully aware of the vulnerability of their supply chain relationships to global shocks. As a result, companies are looking for new technological solutions to strengthen their supply chains, making them more robust, resilient, and agile. While the pandemic proved to be a deadly blow to global companies, it was a critical and historical event that would forever change supply chains.

This paper will focus on embedding risk management with global and domestic suppliers for multi-echelon supply chains. The motivation behind tackling this problem comes from the lack of existing research that combines outsourcing with multi-echelon supply chains. Most papers independently focus on outsourcing, supply chain risk management, or multi-echelon supply chain optimization, but none effectively incorporate them all. Therefore, this proves a need for research that combines all three industrial engineering topics to serve as a meaningful backbone for future work.

This paper focuses on embedding risk management with global and domestic suppliers for multi-echelon supply chains. The second section of this paper provides a literature

review on previous research in this field, including mathematical models and metaheuristics. The third section presents the problem description and discusses the modeling approach. The fourth section solves the mathematical model using a mixed-integer linear program. The fifth section explains the steps in creating the Tabu Search algorithm and outputs experimental results for 13 problem instances. The sixth section conducts a multi-regression statistical analysis of results. Lastly, the seventh section presents conclusions, contributions, and future work directions.

1.4 Main Contributions

This paper provides the following contributions, which differentiate it from current research:

- We are modeling a multi-echelon supply chain and incorporating risk with outsourcing.
- We are modeling and solving a single-objective liner program using Excel Solver up to decision variable limits.
- We are building a scalable Python computer program utilizing a Tabu Search algorithm.
- We conduct a multi-regression analysis of variance of the Tabu Search results using Minitab.

CHAPTER 2. LITERATURE REVIEW

The topic of risk management in a multi-echelon supply chain has brought along numerous avenues for research. This research provides current research findings and better understands the academic and commercial resources used in solving supply chain optimization problems. The following categories outline the literature research conducted:

- Supply Chain Risk Management
- Supply Chain Network Design
- Outsourcing
- Metaheuristics

2.1 Supply Chain Risk Management

To better understand supply chain risk management (SCRM), researchers focused on one key aspect of SCRM explicitly dealing with the delivered quantity along the supply chain) known as its nature. SCRM is non-binary, which means that the volumes transferred between levels do not promise supply or not. Instead, Mohib & Deif (2019) suggest that it captures and assesses different delivery levels from other suppliers along the different stages. A high-level sequence of steps is crucial for supply chain managers to minimize risk probability and impact. Supply chain managers must understand and explain the economic challenges that arise when risks spiral out of control. More specifically, the focus on the use of risk in theory and practice, particularly the integration of risk management in corporate systems and assessing the financial implications (Heckmann et al., 2015).

The agricultural industry is an excellent example of a high-risk multi-echelon supply chain due to seasonality uncertainties, long lead times, and goods' perishability. Behdazi et al., (2018) saw that the agricultural industry was new to technological solutions and that their supply chains lacked mathematical models for optimizing profits from crops. Behdazi et al., (2017) also identified significant gaps in the industry when researching solutions to this problem, which include: perishability modeling, multi-period planning, rare high-impact disruption, and the combination of them with operational uncertainty, robust and resilient strategies, demand-side disruptions, highly integrated information-driven supply chains, and approaches endorsed by high-level management. González-Zapatero et al. (2020) acknowledged that supply chain risk management strategies should fit with contextual factors like 'fit as profile deviation' and 'fit as moderation,' They considered a sample of 106 companies to confirm the proposed model.

2.2 Supply Chain Network Design

A critical paper on the agricultural market about supply chain multi-state risk assessment discussed using the universal generating function and compared it to other models such as the power means, series, and parallel series (Mohib & Deif, 2019). The purpose was to prove that their method of UGF was far superior to the other techniques when comparing both the risk value percentage (using a quantitative mathematical method) and the method's ability to be applied to multi-level supply chains. Overall, Mohib & Deif (2019) developed a new risk assessment approach that can capture the various delivery levels

and their associated risks for different suppliers across different stages, leading to more comprehensive and accurate assessment results and mitigation decisions.

Aqlan & Lam (2015) provided an existing framework for risk mitigation within supply chains using three main components. These included a survey, Bow-Tie analysis, and fuzzy inference system (FIS). While bow-tie analysis and surveys are influential to the problem, the FIS's primary purpose is to reduce the risk data's uncertainty using fuzzy logic. This theory provides a valuable solution to understanding, quantifying, and handling uncertain and vague risk data. Aqlan & Lam (2015) also suggested other qualitative techniques for risk identification and risk analysis, such as failure mode and effect analysis (FMEA), empirical analysis, process-performance modeling, and simulation. Due to its modeling flexibility and sensitivity analysis, a simulation is an effective tool for visualizing supply chain risks. Another practical modeling approach is hybrid models because they utilize both qualitative and quantitative techniques. Due to the uncertainty and the lack of risk data, hybrid modeling techniques are effective for risk analysis, assessment, and the development of proper mitigation strategies. The most common tools used for the mixed modeling of risks in supply chains are questionnaires, analytic hierarchy process (AHP), fuzzy logic, fuzzy-AHP, decision tree analysis (DTA), and cluster analysis (Aqlan & Lam, 2015).

Yan et al. (2017) introduced risk assessment and control of supply chains under the Internet of Things. Specifically, they addressed how researchers can use programs and applications under the Python or Tabu Search internet to create models to depict multi-

echelon supply chains, a rare topic to find in research today. Other research dealt with minimizing supply chain cost with embedded risk using various computational intelligence approaches. Kumar et al. (2010) considered a multi-echelon global supply chain model, where raw material suppliers, manufacturers, warehouses, and markets are in different countries. Furthermore, this paper identified all operational risk factors, expected value and probability of occurrence, and associated additional cost amongst domestic to global supply quantities. Computational intelligence techniques such as genetic algorithm, particle swarm optimization, and artificial bee colony solved a supply chain network problem to obtain a solution (Kumar et al., 2010). Many computational approaches and metaheuristics can solve the same problem within a reasonable time. Tabu Search will be researched later in this paper.

2.3 Outsourcing

One of the most known reasons companies outsource is to reduce costs in response to changing economic pressure. However, as outsourcing shifts to being used for more vital functions, it leads to losing core competencies. In global supply chains, risks constitute a single point of failure that will disrupt the supply network.

Cha et al. (2008) presented an economic learning model for offshoring a firm's knowledge levels, production costs, and coordination costs. They learned that short-lived offshoring projects might generate substantial cost savings to the domestic firm when transfers are not sufficiently large. However, long-lived offshoring projects may disrupt the knowledge supply chain, resulting in significant losses in the project's later stages

(Cha et al., 2008). Kouvelis & Milner (2002) studied a firm's interplay of demand and supply uncertainty in capacity and outsourcing decisions in multi-echelon supply chains. They found that as the market's responsiveness to investments made by the firm increases, the reliance on outsourcing generally increases. Furthermore, more significant supply uncertainty increases the need for vertical integration, while more substantial demand uncertainty increases outsourcing reliance (Kouvelis & Milner, 2002). Kroes & Ghosh (2009) compared a firm's outsourcing drivers and its competitive priorities and assess the impact of unity on both supply chain performance and business performance. They noticed that outsourcing congruence across all five competitive priorities is positively and significantly related to supply chain performance and supply chain performance in a firm entirely and significantly associated with its business performance (Kroes & Ghosh, 2009).

Relying on external experts creates ability empowerment and false security. The pharmaceutical industry outsources many products to India and China due to the constant challenges with manufacturing processes. While outsourcing builds core competencies, it also brings risks. Mokrini et al. (2016) presented a decision model that considers the dangers of outsourcing logistics in the pharmaceutical supply chain using risk identification and a multi-criteria risk assessment model using ELECTRE TRI. In addition, König & Spinler (2016) presented a conceptual risk management framework, showing the effect of logistics outsourcing on shippers' supply chain vulnerability. They found that raw material suppliers increasingly use logistics outsourcing. Still, its relation to supply chain risk management is rarely covered as logistics outsourcing can have an

ambiguous effect on shippers influencing random internal and external factors (König & Spinler, 2016)

Lee & Hong (2018) explored both established and emerging risks that may arise from outsourcing and designed a simulation model to have a quantitative chance of outsourcing activities in the supply chain network. The proposed method involved a qualitative risk analysis known as the Supply Chain Risk - Failure Mode and Effect Analysis (SCR-FMEA). SCR-FMEA integrates risk identification, research, and mitigation actions to evaluate supply chain outsourcing risk (Lee & Hong, 2018). In addition, Lee et al. (2002) considered an advanced planning and scheduling model in which each customer order has a due date and outsourcing is available in a manufacturing supply chain. They solved the model using a genetic algorithm heuristic approach and found that the method efficiently solved the model. It produced the best process plans for operation sequence and machine selection with outsourcing and schedules for all orders (Lee et al., 2002). Another essential process involved in supply chain outsourcing is a prior evaluation of potential partners for expected costs and risks. Olson & Wu (2011) specifically used a DEA simulation model and a Monte Carlo simulation using a risk-adjusted cost concept. They found numerous potential outsourcing strategies to China and other nations under various risk forms. Hernandez & Haddud (2018) aimed to unveil the areas that required more focus, considering the point of view of Chinese manufacturers and driving the effectiveness of SCRM strategies. The study showed the main factors that impacted value creation in industry, forcing other elements such as transportation, financial, and information to require more attention (Hernandez & Haddud, 2018).

2.4 Metaheuristics

One of the most widely used metaheuristics is Tabu Search. According to Glover et al. (2007), "Tabu search (TS) is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality." Gendreau (2003) mentions Tabu Search as a highly effective metaheuristic used to solve large optimization problems. While other solution methodologies like Ant Colony Optimization, Particle Swarm Optimization, Artificial Immune Systems, Genetic Algorithms have increased in popularity due to their natural analogies, Tabu Search allows local search methods to overcome local optima. Tabu Search's basic principle is to pursue local search whenever it encounters a local optimum by not allowing non-improving moves through tabu lists (Gendreau, 2003).

Mohammed & Duffuaa (2020) utilize a Tabu Search algorithm combined with supply chain network optimization to demonstrate the scalability of larger and more complex problem instances. Then, they used CPLEX to solve the multi-objective linear program to obtain an optimal solution. Mohammed & Duffuaa (2020) found that the developed Tabu Search algorithm can obtain high-quality solutions, short computational times, and solution diversity. The same authors used simulated annealing as a different approach (Mohammed & Duffuaa, 2019). Fatehi-Kivi et al. (2021) developed a three-echelon supply chain structure and solved their mixed-integer linear program using three different metaheuristics: Harmony Search, Tabu Search, and Genetic Algorithm. They used an ANOVA statistical analysis to compare results and found that harmony search provided

the best quality solution (Fatehi-Kivi et al., 2021). Lee & Ozsen (2020) used three metaheuristics to solve an integrated location-inventory problem. However, this paper focused heavily on incorporating operational and tactical aspects such as lead times and safety stock. They developed a Lagrangian algorithm, a Genetic Algorithm, and a Tabu Search algorithm by introducing a novel concept known as the indirect cost ratio used to evaluate candidate facilities. They concluded that the proposed Tabu Search heuristic yielded near-optimal solutions and outperformed the other two in computational efficiency, solution quality, and robustness. Lee & Kwon (2010) implemented the Tabu Search metaheuristic to solve a supply chain optimization problem and compared results to other methods. This single-objective optimization problem was solved using CPLEX and a Tabu Search algorithm adopted from the literature that draws arcs from DC's to other nodes based on a priority index known as Unit Cost Ratio (Lee & Kwon, 2010). After obtaining the results, Braido et al. (2016) expanded and found an 81.03% reduction of the average processing time but an increase of 4.98% in the average cost of the solutions compared to the optimal results. They concluded their work to be successful due to their ability to solve large-scale supply chain optimization problems with less computational time than previous literature (Braido et al., 2016). Melo et al. (2012) claimed to be the first to investigate the suitability of Tabu Search for tackling large-scale multi-period, multi-objective supply chain networks. They used CPLEX to solve the smaller problem instances but use the Tabu Search algorithm to solve large-scale problems with a shorter computational time. Once again, they compare their linear program CPLEX results with Tabu Search results with a gap ratio/percentage and find they can reach solutions within 1% of the linear relaxation bound in reasonable

computational times (Melo et al., 2012). Lastly, Shahraki & Sharifi (2019) used a multi-level, multi-period supply chain network problem in agile organizations only. Each level of a company's production, storage, and transportation requires efficient decision-making. Their goal was to minimize overall operating costs across the entire supply chain and improve customer satisfaction. In addition to Tabu Search, they used a Lagrange algorithm to solve the problem. After analyzing the results, they found that answers were within 3% of the optimal solution achieved (Shahraki & Sharifi, 2019).

2.5 Literature Review Summary

In summary, research on supply chain risk management, supply chain network design, outsourcing, and metaheuristics (specifically Tabu Search) provided insight into potential literature gaps. Although current literature contains a few topics in each paper, no one has combined all four topics into one cohesive report with a mathematical model and Tabu Search approach.

This paper aims to solve a comprehensive supply chain network optimization problem with outsourcing and embedded risk. Furthermore, we propose a mixed-integer linear programming model and use a Tabu Search algorithm to solve small, medium, and large problem instances. Lastly, we conduct a multi-regression statistical analysis.

2.6 Literature Contributions

Table 2-1 explicitly differentiates the topics and solution approaches found in literature research from work completed in this paper. The contributions that differentiate this

paper from others are solving a single-objective linear program with Excel Solver and Tabu Search for small, medium, and large multi-echelon supply chain networks with embedded risk and outsourcing. While other optimization software may be more suitable to solve large-scale problems, Excel Solver demonstrates the difficulty in scaling and replicating a typical supply chain network. To establish a scalable model, a computer program utilizing the Tabu Search algorithm is a more promising computational approach for solving large-scale optimization problems in minimal time. Lastly, a multi-regression statistical analysis provides critical findings within the data.

Table 2-1: Summary of Contributions to Literature

Source	Topic			Solution Approach							
Reference	Supply Chain Optimization	Outsourcing	Risk Management	Math Model	Simulation	Tabu Search	Genetic Algorithm	Simulated Annealing	Particle Swarm Optimization	Artificial Bee Colony	Statistical Analysis
Aqlan & Lam, 2015 [2]	x		x	x							
Behzadi et al., 2018 [3]	x		x	x							
Behzadi et al., 2017 [4]	x		x	x							
Braido et al., 2016 [6]	x			x		x					
Lee et al., 2018 [7]	x	x	x	x							

Cha et al., 2008 [8]	x	x		x							
Fatehi Kivi et al., 2021 [10]	x			x		x	x				x
González-Zapatero et al., 2020 [14]	x		x								x
Heckmann et al., 2015 [15]	x		x	x							
Hernandez & Haddud, 2018 [16]	x	x	x								
König & Spinler, 2016 [18]	x	x	x	x							
Kouvelis & Milner, 2002 [19]	x	x	x		x						
Kroes & Ghosh, 2009 [20]	x	x									x
Kumar et al., 2010 [21]	x	x	x	x			x		x	x	
Lee & Ozsen, 2020 [22]	x					x					
Lee & Kwon, 2010 [23]	x		x	x		x					
Lee et al., 2002 [24]	x	x	x				x				
Melo et al. 2012 [26]	x			x		x					
Mohammed & Duffuaa, 2020 [28]	x		x	x		x					
Mohammed & Duffuaa, 2019 [29]	x		x	x				x			
Mohib & Deif, 2019 [30]	x		x	x							
Mokrini et al., 2016 [31]		x	x								x
Olson & Wu, 2011 [32]	x	x	x		x						
Shahraki & Sharifi, 2019 [35]	x		x			x					
Williamson, 2008 [38]	x	x	x								
Yan et al., [39]	x		x			x					
Present Study (this paper)	x	x	x	x		x					x

CHAPTER 3. MATHEMATICAL MODEL FORMULATION

3.1 Problem Description

Supply chain risk management has been an increasingly researched topic in the past decade, especially in multi-echelon supply chains. Multi-echelon supply chains drive lower costs, reduce capital assets, and get products to market more efficiently than the competition. They do this by evaluating supply levels and risk probabilities for each possible path between suppliers, plants, warehouses, distributors, and retailers at each level. Rather than a binary matter of receiving the entire supply or not from one source all at once, multi-echelon supply chains assess all supplier avenues and solutions for the final product farthest downstream, which is a better representation of real-life supply chains. Also, many supply chains are incredibly vulnerable to different risk factors that constantly influence operations with the rise of globalization. Costs are often associated with each risk factor when allocating specific goods at the required quality, quantity, place, and time. These issues tie into issues involving outsourcing semi-finished or finished products when in-house production risks are very high. Incorporating this concept into the already established and well-researched topic of multi-echelon supply chain poses a complex challenge that requires solving.

As a result, this paper addresses the need for an updated model and method that presents an accurate depiction of outsourcing in global, multi-echelon supply chains where mitigating risks and minimizing costs are critical. Overall, this paper considers a multi-echelon supply chain network with global and domestic raw material suppliers, manufacturing plants, warehouses, and markets. The objective function aims to minimize

the total cost of the supply chain network with embedded risk. The problem is modeled and solved as a mixed-integer linear program with a commercial solver and solved using a Tabu Search algorithm for small, medium, and large problems.

The mixed-integer linear program presented in this section extends the work of Kumar et al. (2010) in formulating the optimization problem for multi-echelon supply chains with embedded risk. Table 3-1, Table 3-2, and Equations 1-28 summarize nomenclature, parameters, and equations used in this paper's model respectfully adopting similar principles used by Kumar et al. (2010). This paper's work expands previous literature by including outsourced suppliers in the model and solving the model with a Tabu Search approach.

3.2 Key Assumptions

The mathematical model carries the following assumptions:

- All raw material suppliers operate domestically or globally.
- The supply chain has multiple suppliers, plants, warehouses, and markets in a global supply chain network.
    - A supply chain level cannot transfer material to the same level (i.e., plant→plant)
- Each supply chain level must fulfill the order quantity for the next level.
- The material never gets lost in the supply chain network.
    - Random market demand determines the quantities transferred (a normal distribution with a minimum and maximum value plus/minus six*sigma).

- All products are homogeneous produced at a 1:1 ratio whose quality and quantity depend on the raw material suppliers and location country.
    - Finished products are kept in the warehouses and incur an inventory cost.
    - All transactions converge into a common currency through exchange rates.
    - All decisions are for one exclusive planning horizon.

3.3 Limitations

This section outlines the limitations of the mathematical model presented in this paper:

- The mathematical model considers a static planning horizon, but in reality, supply chains are constantly sending material/products between echelons in a dynamic approach.
- The mathematical model only considers risk probability and impact along the supply chain system; however, literature has shown risk mitigation strategies as a key parallel topic when discussing risk management.
- The mathematical model disregards quality management principles. In reality, material/products are lost in transferring between echelons.
- The mathematical model considers only one homogeneous product. In reality, supply chains contain hundreds to thousands of products made up of millions of parts in a mathematically complex Material Requirements Planning (MRP) system.
- The mathematical model picks random values according to the data parameter range given; however, it would be beneficial to conduct a sensitivity analysis to understand the impact of risk probability and reliability on total supply chain cost.

- The mathematical model considers a single-objective cost function with embedded risk. Supply chain managers may want to separate risk and total cost into two different objective functions in practice.

3.4 Nomenclature

Table 3-1 provides a comprehensive list of notations frequently used throughout the model:

Table 3-1: Model Notations and Descriptions

Notation	Description
SD	Domestic suppliers
SO	Global suppliers
P	Plants
W	Warehouses
M	Markets
rsD_{ij}^{SDP}	Raw material supplied from domestic supplier i to plant j
rsO_{ij}^{SOP}	Raw material supplied from global supplier i to plant j
rs_{ij}^{SP}	Raw materials supplied from supplier i to plant j
q_{ij}^{PW}	quantity supplied from plant i to warehouse j
q_{ij}^{WM}	quantity supplied from warehouse i to market j
Q_i^P	Total quantity supplied to plant i
Q_i^W	Total quantity supplied to warehouse i
Q_i^M	Total quantity supplied to market i
$Q_{max}^{P_j}$	Maximum production capacity of plant j, where $j \in \{1,2,3,\dots P\}$
$Q_{max}^{W_j}$	Maximum production capacity of warehouse j, where $j \in \{1,2,3,\dots W\}$
Md_j	Mean demand of market j
Md_{min}^{j}	Minimum demand to be satisfied for market j, where $j \in \{1,2,3,\dots M\}$
Md_{max}^{j}	Maximum demand to be satisfied for market j, where $j \in \{1,2,3,\dots M\}$
uc_i^P	Cost of unit production for plant i
fc_i^P	Fixed cost of operation for plant i
ic_i^W	Inventory cost for warehouse i

t	Time
DSL_{tj}^{i}	Lead time for domestic supplier i to deliver raw material to plant j
OSL_{tj}^{i}	Lead time for global supplier i to deliver raw material to plant j
PL_{tj}^{i}	Lead time for plant i to deliver to warehouse j
WL_{tj}^{i}	Lead time for warehouse i to deliver to market j
CSD_{ij}	The cost function per unit of raw materials supplied domestically i to plant j
CSO_{ij}	The cost function per unit of raw materials supplied outsourcing i to plant j
CP_{ij}	The cost function of supply for plant i to warehouse j
CW_{ij}	The cost function of supply for warehouse i to market j
ηD_{ij}^{SDP}	Set of all scenarios of lead times for delivering raw materials from domestic supplier i to plant j
ηO_{ij}^{SDP}	Set of all scenarios of lead times for delivering raw materials from global supplier i to plant j
ηP_{ij}^{PW}	Set of all scenarios of lead times for delivering supply from plant i to warehouse j
ηW_{ij}^{WM}	Set of all scenarios of lead times for delivering supply from warehouse i to market j
Ω_{i}^{SD}	Set of all scenarios for domestic suppliers i
Ω_{i}^{SO}	Set of all scenarios for global suppliers i
Ω_{i}^{P}	Set of all scenarios for plants i
α_{i}^{SD}	Reliability for domestic supplier i
α_{i}^{SO}	Reliability for global supplier i
α_{i}^{P}	Reliability for plant i
p	Probability
L	Loss function in terms of ordered quantity of supply due to failure
L_t	Lead Time
er_{i}^{SD}	Exchange rate for domestic supplier i
er_{i}^{SO}	Exchange rate for global supplier i
er_{i}^{P}	Exchange rate for plant i
er_{i}^{W}	Exchange rate for warehouse i
er_{i}^{M}	Exchange rate for market i
R_{i}^{C}	Inventory cost for an excess of supply
G_{i}^{C}	Goodwill loss cost for a shortage of supply
$f_j(d)$	Probability density function market j demand
DSC	Total global supplier cost
OSC	Total global supplier cost
PC	Total plant production cost
PWC	Total plant-warehouse cost
WMC	Total warehouse-market cost
MC	Total market cost

3.5 Decision Variables

Table 3-2 shows the decisions variables of all quantities supplied and transferred across the supply chain network:

Table 3-2: Model Decision Variables

Notation	Description
rsD_{ij}^{SDP}	Raw material supplied from domestic supplier i to plant j
rsO_{ij}^{SOP}	Raw material supplied from global supplier i to plant j
q_{ij}^{PW}	Quantity supplied from plant i to warehouse j
q_{ij}^{WM}	Quantity supplied from warehouse i to market j

3.6 Objective Function

This model aims to optimize the quantities transferred among all combinations between suppliers-plants, plants-warehouses, and warehouses-markets while minimizing the expected cost of operations with embedded risk. The cost and risk functions considered in this problem are assumed to be known and given based upon historical data found on all suppliers, plants, warehouses, and markets.

3.6.1 Objective Function Costs

The following sub-costs total up to the supply chain's total cost:

- Domestic Supplier Cost
- Global Supplier Cost
- Plant Production Cost
- Plant-Warehouse Cost
- Warehouse-Market Cost
- Market Cost

3.6.1.1 Domestic Supplier Cost

At the start of the supply chain network, suppliers supply raw materials to plants. Each domestic supplier incurs a cost-dependent lead time. Equation 1 shows the cost of raw materials for domestic suppliers below:

$$\sum_{i=1}^{SD}\sum_{j=1}^{P}\sum_{t\geq \min DSL_{tj}^{i}}^{\max DSL_{tj}^{i}} rsD_{ij}^{SDP} \times CSD_{j}^{i}(t) \times p\left(\eta D_{ij}^{SDP}\right) \times er_{i}^{SD} \quad (1)$$

The raw materials' quality is a risk factor, which is always associated with the domestic suppliers. Equation 2 shows the risk factor for the quality of raw materials for domestic suppliers:

$$\sum_{\Omega_i^{SD}} p\left(\eta D_{ij}^{SDP}\right) \times \left(\alpha_i^{\mathrm{SD.}}\right) \times er_i^{SD} \quad (2)$$

Risk at any level in the supply chain is directly dependent on the level previously established. For example, the production of goods at the plant level depends on the quality and quantity supplied from the supplier level. There is also a risk cost due to the supplier's failure to deliver the raw materials within the maximum allowed lead time. Equation 3 shows a loss of production and associated profit loss below:

$$\sum_{i=1}^{SD}\sum_{j=1}^{P}\left(1-\sum_{t\geq \max DSL_{tj}^{i}} p\left(\eta D_{ij}^{SDP}\right)\right) \times L\left(rsD_{ij}^{SDP}\right) \times er_i^{SD} \quad (3)$$

Equation 4 summarizes the domestic supplier cost below:

$$DSC = \left(\sum_{i=1}^{SD} \sum_{j=1}^{P} \sum_{t \geq \min DSL_{tj}^{i}}^{\max DSL_{tj}^{i}} rsD_{ij}^{SDP} \times CSD_{j}^{i}(t) \times p\left(\eta D_{ij}^{SDP}\right) \times er_{i}^{SD} \right)$$

$$\div \left(\sum_{\Omega_i^{SD}} p\left(\eta D_{ij}^{SDP}\right) \times \left(\alpha_i^{\mathrm{SD.}}\right) \times er_i^{SD} \right) \tag{4}$$

$$+ \left(\sum_{i=1}^{SD} \sum_{j=1}^{P} (1 - \sum_{t \geq \max DSL_{tj}^{i}} p\left(\eta D_{ij}^{SDP}\right)) \times L\left(rsD_{ij}^{SDP}\right) \right.$$

$$\left. \times er_i^{SD} \right)$$

3.6.1.2 Global Supplier Cost

Global suppliers incur different raw material costs and risk factors that influence their mathematical approach. Equation 5 shows the cost of the raw materials for global suppliers below:

$$\sum_{i=1}^{SD} \sum_{j=1}^{P} (1 - \sum_{t \geq \max DSL_{tj}^{i}} p\left(\eta D_{ij}^{SDP}\right)) \times L\left(rsD_{ij}^{SDP}\right) \times er_i^{SD} \tag{5}$$

The raw materials' quality is a risk factor, which is always associated with the global suppliers. Equation 6 shows the risk factor for the quality of raw materials for global suppliers below:

$$\sum_{\Omega_i^{SO}} p\left(\eta O_{ij}^{SOP}\right) \times \left(\alpha_i^{\mathrm{SO}}\right) \times er_i^{SO} \tag{6}$$

Like the domestic supplier cost, Equation 7 shows the loss of production and associated profit loss for global supplier cost below:

$$\sum_{i=1}^{SO}\sum_{j=1}^{P}(1-\sum_{t\geq \max OSL_{t_j}^{i}} p\left(\eta O_{ij}^{SOP}\right))\times L\left(rsO_{ij}^{SOP}\right)\times er_i^{SO} \tag{7}$$

Equation 8 summarizes the global supplier cost below:

$$OSC=\left(\sum_{i=1}^{SO}\sum_{j=1}^{P}\sum_{t\geq \min OSL_{t_j}^{i}}^{\max OSL_{t_j}^{i}} rsO_{ij}^{SOP}\times CSO_j^i(t)\times p\left(\eta O_{ij}^{SOP}\right)\times er_i^{SO}\right) \div\left(\sum_{\Omega_i^{SO}} p\left(\eta O_{ij}^{SOP}\right)\times\left(\alpha_i^{\mathrm{SO}}\right)\times er_i^{SO}\right) +\left(\sum_{i=1}^{SO}\sum_{j=1}^{P}(1-\sum_{t\geq \max OSL_{t_j}^{i}} p\left(\eta O_{ij}^{SOP}\right))\times L\left(rsO_{ij}^{SOP}\right)\times er_i^{SO}\right) \tag{8}$$

3.6.1.3 Plant Production Cost

Each plant participating in the supply chain has a defined cost per unit of production and fixed cost of operation. Fixed costs may not always be present. Equation 9 shows the plant production cost below:

$$\sum_{i=1}^{P}\left(\left(\sum_{j=1}^{W}\mathrm{q}_{\mathrm{ij}}^{\mathrm{PW}}\times uc_i^P\right)+fc_i^P\right) \tag{9}$$

The geographical location and other factors among the participating plants significantly impact the quality and quantity of products produced. Equation 10 shows the plant quality risk factor below:

$$\sum_{\Omega_i^P} p\left(\eta \mathrm{P}_{\mathrm{ij}}^{\mathrm{SP}}\right) \times \left(\alpha_i^{\mathrm{P}}\right) \tag{10}$$

Equation 11 summarizes the plant production cost below:

$$PC = \left(\sum_{i=1}^{P}\left(\left(\sum_{j=1}^{W} \mathrm{q}_{\mathrm{ij}}^{\mathrm{PW}} \times uc_i^P\right) + fc_i^P\right)\right) \div \left(\sum_{\Omega_i^P} p\left(\eta \mathrm{P}_{\mathrm{ij}}^{\mathrm{SP}}\right) \times \left(\alpha_i^{\mathrm{P}}\right)\right) \tag{11}$$

3.6.1.4 Plant-Warehouse Cost

The cost between the plants and warehouses participating in the supply chain includes the logistics costs between them and their associated risk cost due to supply failure. Equation 12 shows the plant-warehouse transportation cost below:

$$\sum_{i=1}^{P}\sum_{j=1}^{W}\sum_{t\geq \min PL_{tj}^{i}}^{\max PL_{tj}^{i}} q_{ij}^{PW} \times CP_j^i(t) \times p\left(\eta P_{ij}^{PW}\right) \times er_i^P \tag{12}$$

Equation 13 shows the risk cost associated with disruption of supply between plant and warehouse below:

$$\left(\sum_{i=1}^{P}\sum_{j=1}^{W}\left(1 - \sum_{\mathrm{t}\geq \min PL_{tj}^{i}}^{\max PL_{tj}^{i}} p(\eta P_{ij}^{PW})\right) \times L\left(q_{ij}^{PW}\right) \times er_i^P\right) \tag{13}$$

Equation 14 summarizes the plant-warehouse cost below:

$$PWC = \left(\sum_{i=1}^{P}\sum_{j=1}^{W}\sum_{t\geq \min PL_{tj}^{i}}^{\max PL_{tj}^{i}} q_{ij}^{PW} \times CP_j^i(t) \times p\left(\eta P_{ij}^{PW}\right) \times er_i^P\right) + \left(\sum_{i=1}^{P}\sum_{j=1}^{W}\left(1 - \sum_{\mathrm{t}\geq \min PL_{tj}^{i}}^{\max PL_{tj}^{i}} p(\eta P_{ij}^{SP})\right) \times L\left(q_{ij}^{PW}\right) \times er_i^P\right) \tag{14}$$

3.6.1.5 Warehouse-Market Cost

The plant-warehouse supply cost is like the warehouse-market supply cost. Equation 15 shows the warehouse-market transportation cost below:

$$\sum_{i=1}^{W}\sum_{j=1}^{M}\sum_{t\geq \min WL_{tj}^{i}}^{\max WL_{tj}^{i}} q_{ij}^{WM}\times CW_{j}^{i}(t)\times p\left(\eta W_{ij}^{WM}\right)\times er_{i}^{W} \tag{15}$$

Equation 16 shows the risk cost associated with disruption of supply between warehouse and market below:

$$\left(\sum_{i=1}^{W}\sum_{j=1}^{M}(1-\sum_{\mathrm{t}\geq \min WL_{tj}^{i}}^{\max WL_{tj}^{i}} p(\eta W_{ij}^{WM}))\times L\left(q_{ij}^{WM}\right)\times er_{i}^{W}\right) \tag{16}$$

Equation 17 summarizes the warehouse-market cost:

$$WMC=\left(\sum_{i=1}^{W}\sum_{j=1}^{M}\sum_{t\geq \min WL_{tj}^{i}}^{\max WL_{tj}^{i}} q_{ij}^{WM}\times CW_{j}^{i}(t)\times p\left(\eta W_{ij}^{WM}\right)\times er_{i}^{W}\right)+ \left(\sum_{i=1}^{W}\sum_{j=1}^{M}(1-\sum_{\mathrm{t}\geq \min WL_{tj}^{i}}^{\max WL_{tj}^{i}} p(\eta W_{ij}^{WM}))\times L\left(q_{ij}^{WM}\right)\times er_{i}^{W}\right) \tag{17}$$

3.6.1.6 Market Cost

The market cost appears if there is an excess of supply or a shortage of supply. The market cost for an excess of supply will be:

$$\left(\sum_{i=1}^{Q}\sum_{j=1}^{M}(Q_i^M - Md_j) \times f_j(d) \times R_i^c\right) \tag{18}$$

The market cost for a shortage of supply will be:

$$\sum_{i=1}^{Q}\sum_{j=1}^{M}(Md_j - Q_i^M) \times f_j(d) \times G_i^c \tag{19}$$

Equation 20 summarizes the market cost:

$$MC = \left(\sum_{i=1}^{Q}\sum_{j=1}^{M}(Q_i^M - Md_j) \times f_j(d) \times R_i^c\right) + \left(\sum_{i=1}^{Q}\sum_{j=1}^{M}(Md_j - Q_i^M) \times f_j(d) \times G_i^c\right) \tag{20}$$

There can only be an excess of supply cost or a shortage of supply cost in any model depending on if the total quantity supplied to market j is under or over the expected market demand. If market demand is over satisfied, the excess market cost will be present, and supply shortage will not and vice versa.

3.6.1.7 Total Supply Chain Cost

As noted previously, the objective function is to minimize the domestic supplier cost, global supplier cost, plant-warehouse cost, warehouse-market cost, and market cost of the entire supply chain. Equation 21 illustrates the objective function:

$$Min\ TC = DSC + OSC + PC + PWC + WMC + MC \tag{21}$$

3.7 Constraints

$$Q_i^M = \sum_{i=1}^{W} q_{ij}^{WM} \tag{22}$$

$$Q_i^W = \sum_{i=1}^{P} q_{ij}^{PW} \tag{23}$$

$$Q_i^P = \sum_{i=1}^{S} rs_{ij}^{SP} \tag{24}$$

$$\sum_{i=1}^{M} q_{ij}^{WM} \leq Q_{max}^{W_j} \tag{25}$$

$$\sum_{i=1}^{W} q_{ij}^{PW} \leq Q_{max}^{P_j} \tag{26}$$

$$\sum_{j=1}^{M} Md_{min}^{j} \leq \sum_{i=1}^{M} q_{ij}^{WM} \tag{27}$$

$$\sum_{j=1}^{M} Md_{max}^{j} \geq \sum_{i=1}^{M} q_{ij}^{WM} \tag{28}$$

Equations 22 through 28 depict constraints of the mathematical model. Equation 22 ensures that the total quantity supplied to markets equals the quantities released from warehouses. Equation 23 ensures that the total quantity supplied to warehouses equals the quantities released from plants. Equation 23 ensures that the raw material supplied to plants equals the quantities released from suppliers. Plant capacities are limited in

Equation 25. Warehouse capacities are limited in Equation 26. Equation 27 ensures that all markets' minimum demand is less than the total quantities delivered from warehouses-markets. Lastly, Equation 28 ensures that all markets' maximum demand is greater than the total quantities delivered from warehouses-markets.

3.8 Experimental Results

To examine the capability of obtaining an optimal solution of the mixed-integer linear model, we use Excel Solver, a downloadable commercial solver for optimization problems (Excel Solver, 2019). The single-objective linear program solves to optimality three problem instances that test the capability of Excel Solver. These instances were solved using a Windows laptop with 16 GB RAM supported by an AMD Ryzen 4700U processor with eight cores and eight threads, a max boost clock up to 4.1 GHz, and a 4 MB cache size (AMD, 2020).

The following section describes the three problem instances. The proposed model begins with the base case schematic of the supply chain network shown in Figure 3-1, adapted from Kumar et al. (2010). The base model, also identified as problem instance #2, entails five suppliers (S), two plants (P), and three warehouses (W), and six markets (M). The proposed model splits the five suppliers into domestic and global suppliers. Therefore, suppliers 1, 2, 3 are domestic suppliers (SD), and suppliers 4 and 5 (SO) are global suppliers that provide the plants' raw materials in the next stage. Suppliers 1-5 may operate in different countries under different environments. The model considers all risk types, including supplier side risks, logistics risks, manufacturer risks, distribution risks,

market risks, and demand risks. Each level of the resilient supply chain carries risk, but holistically they impact the transfer of goods from one level to another and impact the entire supply chain. In addition, the supply chain network incurs costs along the way in terms of the following: raw materials costs, quality costs, supplier costs, production costs, fixed costs, transportation costs, inventory costs, goodwill loss costs, excess supply costs, and shortage supply costs. The purpose of this mathematical model is to minimize the total risk and total costs of the entire supply chain while satisfying demand over a single given planning period.

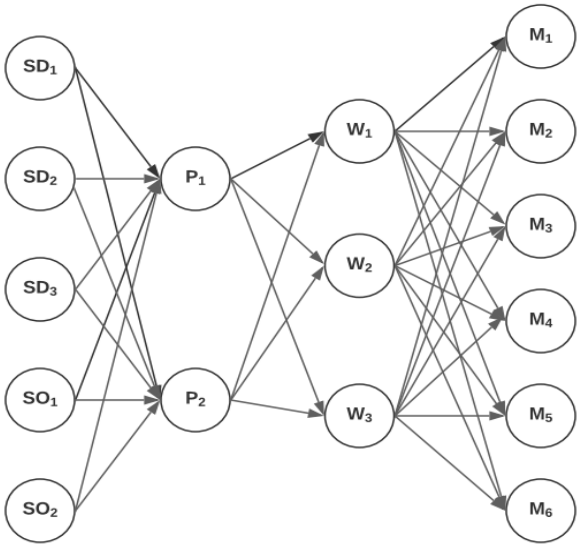

Figure 3-1: Base Case Schematic

Given the mathematical model, Excel Solver solves three problem instances. In problem instance #1, the supply chain schematic contains one domestic supplier, one global supplier, one plant, one warehouse, and one market. Problem instance #1 acts as a proof of concept to verify and validate the equations and mathematical calculations presented in Chapter 3 of this paper. Figure 3-2 depicts problem instance #1:

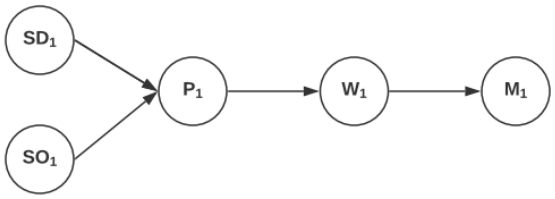

Figure 3-2: Problem Instance #1 Schematic

Problem instance #2 involves three domestic suppliers, two global suppliers, two plants, three warehouses, and six markets. Figure 3-3 depicts the base case of problem instance #2:

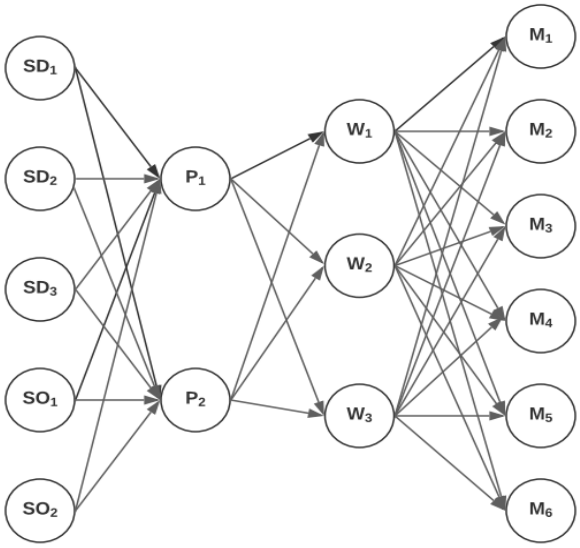

Figure 3-3: Problem Instance #2/Base Case Schematic (Kumar et al., 2010)

Problem instance #3 maximizes the number of decision variables that Excel Solver can handle (200 decision variables) with six domestic suppliers, six global suppliers, five plants for production, seven warehouses, and fifteen markets. Figure 3-4 depicts problem instance #3:

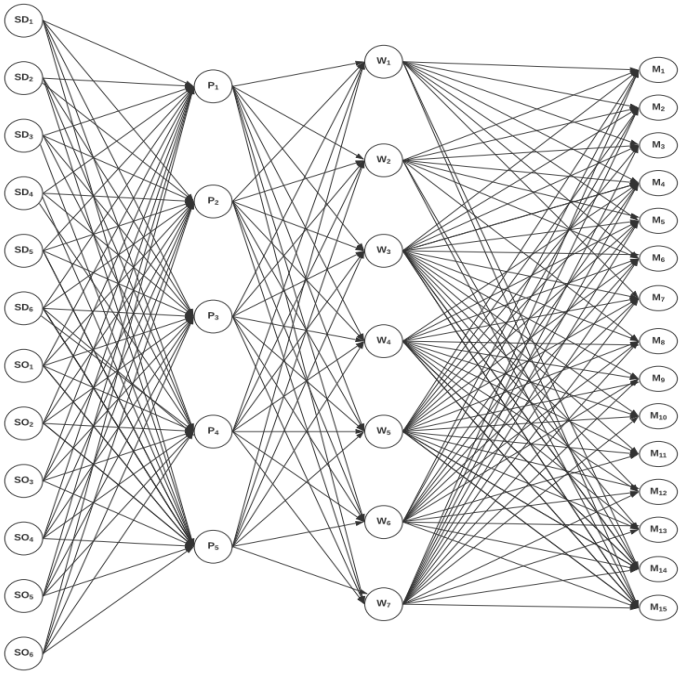

Figure 3-4: Problem Instance #3 Schematic

The mixed-integer linear model was manually scaled and replicated for all three problem instances. The model utilized the previously noted decision variables, constraints, and objective function to obtain an optimal total cost solution using the Simplex LP option in Excel Solver. Table 3-3 displays a comprehensive list of the commercial solver's results.

Table 3-3: Excel Solver Results

Problem Instance	SD	SO	P	W	M	Total Supply Chain Cost (USD)
1	1	1	1	1	1	$167,707.90
2	3	2	2	3	6	$723,519.52
3	6	6	5	7	15	$1,939,812.60

In all three problem instances, Excel Solver was able to find a global minimum solution for total supply chain cost (United States Dollar). Although we reached an optimal solution with one replication, it is essential to acknowledge Excel Solver's limitations in scalability and replicability. Problem sizes exceeding 200 decision variables prove problematic for the Excel Solver software. Therefore, we present a Tabu Search heuristic in the following chapter.

CHAPTER 4. TABU SEARCH HEURISTIC

4.1 Tabu Search

Once again, "Tabu search (TS) is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality" (Glover et al., 2007). The Tabu Search algorithm prevents moves that take the solution into previously visited search spaces known as tabu. While tabu search does accept non-improving solutions, specific parameters prevent the program from getting stuck in local minimums. Tabu Search utilizes short-term memory based on recency of occurrence. Short-term memory returns suitable components to localize and intensify a search known as intensification (Liang, 2020). It accomplishes this by creating a tabu list. The tabu list is an input parameter to access short-term memory. Tabu list solution moves are kept within the list on a countdown timer as they are not visited more than once.

The Tabu Search program contains three key input parameters: iterations number, neighbor size, and tabu list size. Iterations number serves as the stopping criterion and indicates a maximum amount of trial runs. Each provides a total cost preventing the program from being stuck in a continuous loop. The number of neighbors is the number of branches the program chooses to diversify its potential solutions pool. Increasing the number of neighbors increases the total amount of differing solutions. Lastly, the tabu list size parameter stores a limited amount of previously visited solutions and the best solution. Once the tabu list size parameter is full, it ejects older solutions. It brings in

newly visited solutions to keep a running list of solutions that the program may not revisit. The program uses a static tabu list.

A flowchart, depicted in Figure 4-1, helps visualize the steps taken in creating the Tabu Search algorithm:

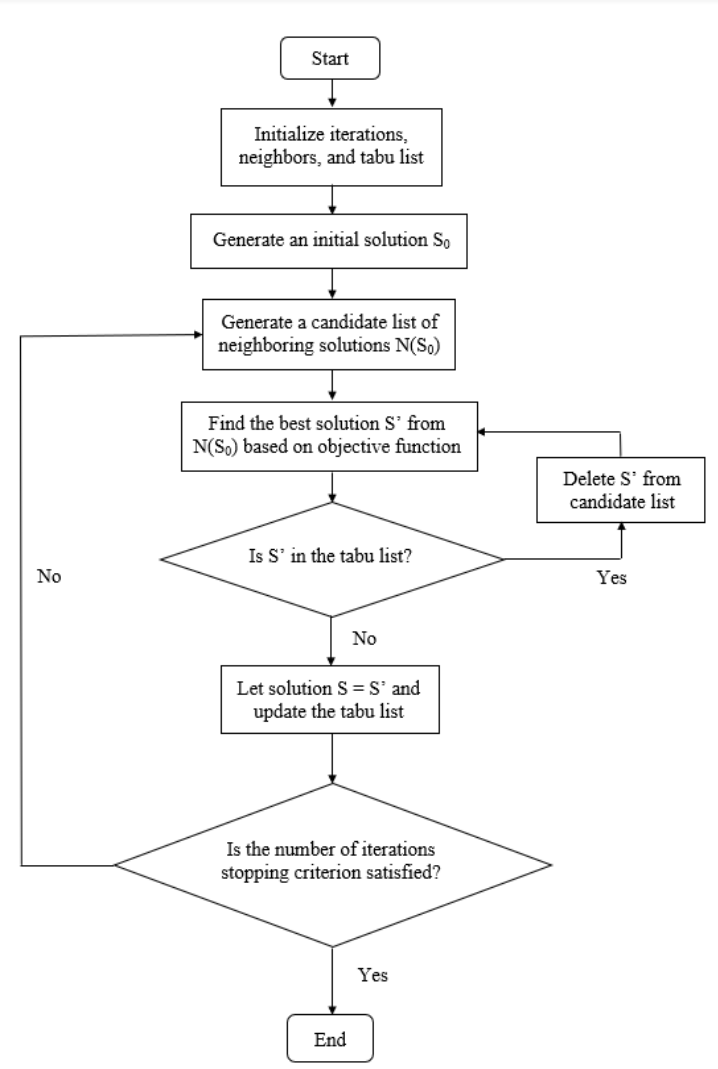

Figure 4-1: Tabu Search Flowchart

4.2 Tabu Search Pseudocode

First, we start with developing an initial solution. Then, we execute the Tabu Search algorithm to explore the local search.

4.2.1 Algorithm 1: Develop a Solution

```
Read in node and edge data.
Initialize solution parameters
    For each market:
        While market demand ≠ 0, choose random path:
            If warehouse path already chosen, select new path
            If market demand > warehouse capacity, take available capacity and select path
            Else, satisfy remaining market demands and update warehouse quantities
    For each warehouse:
        While warehouse demand ≠ 0, choose a random path:
            If plant path already chosen, select new path
            If warehouse demand > plant capacity, take available capacity and select path
            Else, satisfy remaining warehouse demands and update plant quantities
    For each plant:
        Choose random suppliers and update supplier quantities
Calculate single-objective function
Return solution
```

4.2.2 Algorithm 2: Tabu Search

```
Initialize Tabu Search parameters: iterations, neighbors, and tabu list size.
Generate initial solution S_0
Run Tabu Search algorithm
    While iterations ≤ stopping criterion:
        Generate neighborhood of solutions N(S_0)
        Select best, unique solution (S') and add to tabu list
        Update tabu list with most recent solutions and current best solution
Return current best solution
```

4.3 Tabu Search Problem Instances

In addition to solving the three problem instances in the Excel Solver model, larger cases of a similar supply chain reflect reality more closely. Scaling the supply chain network

proves difficult for a mathematical model that is solved using Excel Solver as only 200 decision variables can be considered. Even for other mathematical model solvers, scaling the variables, parameters, and constraints can be repetitive and time-consuming. As a result, a computer program would be highly beneficial for scaling this problem to any supply chain network size or variation. A Tabu Search computer program demonstrates a reliable solution methodology of any supply chain size.

In addition to the previous problem instances (#1-3), the Tabu Search computer program uses ten other instances that exceed 200 decision variables. Each problem instance runs with differing values for iterations, neighbors, and tabu list size. Each problem instance runs with ten replications. Table 4-1 presents a comprehensive list of problem instances below:

Table 4-1: Tabu Search Problem Instances

Problem Instance	SD	SO	P	W	M	Iterations	Neighbors	Tabu List Size
1.1	1	1	1	1	1	100	150	50
1.2	1	1	1	1	1	200	300	100
1.3	1	1	1	1	1	400	600	200
2.1	3	2	2	3	6	100	150	50
2.2	3	2	2	3	6	200	300	100
2.3	3	2	2	3	6	400	600	200
3.1	6	6	5	7	15	100	150	50
3.2	6	6	5	7	15	200	300	100
3.3	6	6	5	7	15	400	600	200
4.1	9	9	7	10	22	100	150	50
4.2	9	9	7	10	22	200	300	100
4.3	9	9	7	10	22	400	600	200

5.1	13	13	10	15	33	100	150	50
5.2	13	13	10	15	33	200	300	100
5.3	13	13	10	15	33	400	600	200
6.1	19	19	15	22	49	100	150	50
6.2	19	19	15	22	49	200	300	100
6.3	19	19	15	22	49	400	600	200
7.1	28	28	22	33	73	100	150	50
7.2	28	28	22	33	73	200	300	100
7.3	28	28	22	33	73	400	600	200
8.1	42	42	33	49	109	100	150	50
8.2	42	42	33	49	109	200	300	100
8.3	42	42	33	49	109	400	600	200
9.1	63	63	49	73	163	100	150	50
9.2	63	63	49	73	163	200	300	100
9.3	63	63	49	73	163	400	600	200
10.1	94	94	73	109	244	100	150	50
10.2	94	94	73	109	244	200	300	100
10.3	94	94	73	109	244	400	600	200
11.1	141	141	109	163	366	100	150	50
11.2	141	141	109	163	366	200	300	100
11.3	141	141	109	163	366	400	600	200
12.1	211	211	163	244	549	100	150	50
12.2	211	211	163	244	549	200	300	100
12.3	211	211	163	244	549	400	600	200
13.1	316	316	244	366	823	100	150	50
13.2	316	316	244	366	823	200	300	100
13.3	316	316	244	366	823	400	600	200

4.4 Tabu Search Data Parameters

Two types of comma-separated values files make up the Tabu Search data input. They are node and edge data. First, the node data files contain data pertinent to each specific node, including exchange rate, plant max capacity, warehouse max capacity, market demand, market variance, market goodwill loss cost, market excess inventory cost, and plant production cost per unit. Second, the edge data files contain data pertinent to the

arcs made between each supply chain node, including edge cost per unit, probability of supply, reliability of supply, and exchange rate at all levels.

We manually generate node data comma-separated values, but edge data files pose difficulty due to their massive factorial scaling. Therefore, a comma-separated value program coded in Python generates the edge data files.

Data values are chosen randomly according to a preset parameter range. This parameter range closely mirrors a range of maximum and minimum data values drawn from literature. In addition, market demand fluctuates with each run of the program. Table 4-2 summarizes the range of numerical outcomes for node and edge data parameters.

Table 4-2: Tabu Search Data Parameters

Parameter	Parameter Range
Exchange Rate	U [0.10; 2.50]
Plant Max Capacity	U [10000; 10000]
Warehouse Max Capacity	U [7000; 10000]
Market Demand Mean	U [1800; 2500]
Market Demand Variance	U [100; 300]
Market Goodwill Loss cost	U [3; 5]
Market Excess Inventory Cost	U [3; 5]
Plant Production Cost	U [20; 50]
Edge Cost	U [10; 60]
Probability	U [0.60; 0.95]
Reliability	U [0.80;1.00]

4.5 Tabu Search Results

A computer program using Python version 3.9.1, an open-source programming language, generated optimal solutions utilizing the Tabu Search algorithm (Liang, 2020). Several

generated instances of the problem tested the capability of the Tabu Search model. A Windows laptop with 16 GB RAM supported by an AMD Ryzen 4700U processor (8 cores and eight threads with max boost clock of up to 4.1 GHz and 4 MB cache size) ran the computer program (AMD, 2020).

The 13 problem instances have differing supply chain network sizes and three different parameter values: iterations, neighbors, and tabu list size. In addition, each problem instance ran with ten replications displaying a minimum, maximum, and mean total supply chain cost ($) and run time (s). This makes 13*3*10 = 390 total runs. Table 4-3 presents a comprehensive list of results for the Tabu Search program below:

Table 4-3: Tabu Search Results

Problem Instance	Total Supply Chain Cost ($) Minimum	Total Supply Chain Cost ($) Maximum	Total Supply Chain Cost ($) Avg.	Run Time (s) Minimum	Run Time (s) Maximum	Run Time (s) Avg.
1.1	$305,460	$326,294	$314,422	0.33	0.38	0.34
1.2	$304,364	$327,735	$320,471	0.83	1.42	1.10
1.3	$306,968	$324,564	$318,132	4.06	6.02	4.87
2.1	$1,077,963	$1,171,826	$1,117,712	0.63	1.08	0.94
2.2	$1,053,617	$1,144,854	$1,094,059	2.52	4.21	2.97
2.3	$1,055,294	$1,109,314	$1,079,856	11.92	14.65	13.00
3.1	$2,503,313	$2,938,476	$2,709,974	1.45	2.42	1.84
3.2	$2,355,350	$2,716,719	$2,562,071	5.90	9.45	6.58
3.3	$2,383,256	$2,573,289	$2,464,563	25.65	28.11	26.14
4.1	$3,899,179	$4,412,494	$4,194,970	2.17	3.74	2.62
4.2	$3,741,042	$4,222,904	$4,020,527	9.52	12.70	9.94
4.3	$3,522,076	$4,027,994	$3,827,620	37.88	39.99	38.64
5.1	$6,226,103	$6,936,944	$6,610,386	3.31	3.81	3.69
5.2	$6,285,264	$6,653,756	$6,493,455	14.43	16.13	15.28
5.3	$5,939,232	$6,355,414	$6,172,603	60.60	66.27	63.25
6.1	$10,338,557	$11,150,236	$10,832,441	5.61	6.74	6.46
6.2	$10,264,711	$10,944,850	$10,566,577	25.76	29.31	26.66
6.3	$10,083,162	$10,693,942	$10,372,145	99.62	144.87	113.69

7.1	$16,391,687	$17,269,007	$16,984,429	11.35	13.37	12.22
7.2	$16,246,583	$16,910,571	$16,630,000	44.26	51.10	46.90
7.3	$15,416,666	$16,642,086	$16,070,246	186.04	268.39	222.37
8.1	$26,201,790	$27,264,122	$26,620,216	26.82	45.18	38.16
8.2	$25,345,720	$26,385,159	$25,985,267	96.29	123.56	106.58
8.3	$24,864,534	$26,084,910	$25,518,667	319.14	420.11	342.77
9.1	$39,839,482	$42,361,909	$41,112,681	45.23	53.76	48.30
9.2	$40,208,504	$41,413,436	$40,801,821	182.97	244.63	207.55
9.3	$39,604,987	$40,735,450	$40,207,018	671.26	815.27	712.78
10.1	$62,945,738	$64,668,437	$63,961,829	94.49	96.86	95.87
10.2	$61,624,101	$64,393,809	$63,023,551	377.49	627.91	474.63
10.3	$61,348,360	$63,217,919	$62,330,256	1434.08	1505.77	1464.51
11.1	$95,394,999	$98,786,202	$97,603,802	182.93	193.27	190.39
11.2	$94,585,572	$98,231,104	$96,512,490	733.60	965.25	838.62
11.3	$94,735,479	$97,703,745	$95,880,657	3056.28	3657.82	3325.13
12.1	$148,203,306	$151,274,641	$150,316,410	411.75	570.55	475.83
12.2	$147,235,790	$150,375,322	$148,938,278	1639.41	2159.51	1800.40
12.3	$144,892,645	$149,403,846	$146,851,716	6500.93	7219.12	6658.20
13.1	$225,188,483	$238,267,389	$231,626,673	928.72	948.13	936.04
13.2	$225,338,416	$230,066,993	$228,642,490	3458.73	3741.92	3625.64
13.3	$221,615,252	$229,373,830	$226,103,637	14831.43	18926.27	16969.45

To quantify the improvement in total supply chain cost ($), we calculate a difference value in percentage. We calculate this by subtracting the previous total supply chain cost average from the new total supply chain cost average, then dividing it all by the previous total supply chain cost. We then multiply by 100 to represent the fraction as a percentage. Using the total cost difference (%) result, we analyze the improvements in every problem instance. The Tabu Search algorithm found an improving total cost average after increasing the Tabu Search input parameters in almost every problem instance. However, results show that an exception occurred in problem instance one. Problem instance one acts as a proof-of-concept with one node of each level. As a result, the random market demand in a tiny supply chain network highly influences the total cost. Because only one

market is present in problem instance one, the algorithm struggles to find a better solution for an ever-changing market demand when there is only one random path it must take.

CHAPTER 5. STATISTICAL ANALYSIS

5.1 Statistical Analysis Software and Data

A statistical analysis method using the Tabu Search results identifies variation in this section. Minitab version 19.2020.1 (64-bit) conducted the statistical analysis (Minitab, 2019). A Windows laptop with 16 GB of RAM supported by an AMD Ryzen 4700U processor (eight cores and eight threads, a max boost clock up to 4.1 GHz, and a 4 MB cache size) ran the statistical model (AMD, 2020).

We conducted a statistical analysis using a different experimental problem that contains six domestic suppliers, six global suppliers, five plants, seven warehouses, and 15 markets. The results of the program contain every combination of iterations (100, 200, 400), neighbors (150, 300, 600), and tabu list size (50, 100, 200). There are three different outcomes of three varying input parameters (3*3*3 = 27 combinations). Each outcome contains ten replications, equaling 270 total runs in one problem.

5.2 Regression Analysis of Total Supply Chain Cost

A quadratic multi-regression approach describes the relationship between the input parameters (iterations, neighbors' number, and tabu list size) and the response (total supply chain cost ($)). The input parameters act as independent continuous predictors, while the response serves as a continuous dependent variable. Equation 29 describes the regression equation with both linear, quadratic, and interaction terms.

$$E(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_{11} X_1^2 + \beta_{22} X_2^2 + \beta_{33} X_3^2 + \beta_{12} X_1 X_2 + \beta_{13} X_1 X_3 + \beta_{23} X_2 X_3 + \beta_{123} X_1 X_2 X_3 \quad (29)$$

In Equation 29, $E(Y)$ represents the expected response value. β_0 is the constant intercept, β_i, where $i = 1, 2, 3$, are the coefficients of the non-interaction terms, β_{ii}, where $i = 1, 2, 3$, are the coefficients of the quadratic terms, β_{ij}, where $i = 1, 2, 3$ and $j = 1, 2, 3$, are the coefficients of the two-way interaction terms, and $\beta_{ijk} W$here $i = 1, 2, 3$ $j = 1, 2, 3$ $k = 1, 2, 3$, are the coefficients of the three-way interaction term. $X_{i,}$ where $i = 1, 2, 3$, represents the three Tabu Search input parameters. The method of least squares obtains the results for the regression analysis.

5.2.1 Quadratic Multi-Regression

First, we conduct a multi-regression analysis with quadratic and linear interaction terms shown in Table 5-1. Then, we analyze if the regression model is significant. We use Minitab version 19 to conduct the regression analysis with a significance level (denoted as α) of 0.05. The regression's p-value is less than 5%, meaning we have a significant regression model. In addition, we test for the significance of the model's constant. The model constant's p-value is less than 5%, meaning we have a non-zero constant intercept.

Table 5-1: Total Supply Chain Cost Regression Analysis

Regression Equation

Total Supply Chain Cost ($) = 2924406 - 1540 Iterations - 422 Neighbors - 386 Tabu List Size + 1.865 Iterations*Iterations + 0.151 Neighbors*Neighbors - 0.39 Tabu List Size*Tabu List Size - 0.058 Iterations*Neighbors + 1.79 Iterations*Tabu List Size - 0.29 Neighbors*Tabu List Size + 0.00145 Iterations*Neighbors*Tabu List Size

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	2924406	91301	32.03	0.000	
Iterations	-1540	452	-3.40	0.001	68.25
Neighbors	-422	302	-1.40	0.163	68.25
Tabu List Size	-386	905	-0.43	0.670	68.25
Iterations*Iterations	1.865	0.738	2.53	0.012	49.00
Neighbors*Neighbors	0.151	0.328	0.46	0.646	49.00
Tabu List Size*Tabu List Size	-0.39	2.95	-0.13	0.895	49.00
Iterations*Neighbors	-0.058	0.621	-0.09	0.926	36.00
Iterations*Tabu List Size	1.79	1.86	0.96	0.338	36.00
Neighbors*Tabu List Size	-0.29	1.24	-0.24	0.814	36.00
Iterations*Neighbors*Tabu List Size	0.00145	0.00469	0.31	0.758	48.25

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
112250	32.54%	29.93%	27.21%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	10	1.57404E+12	1.57404E+11	12.49	0.000
Iterations	1	1.45919E+11	1.45919E+11	11.58	0.001
Neighbors	1	24687330199	24687330199	1.96	0.163
Tabu List Size	1	2290034582	2290034582	0.18	0.670
Iterations*Iterations	1	80481369454	80481369454	6.39	0.012
Neighbors*Neighbors	1	2659595076	2659595076	0.21	0.646
Tabu List Size*Tabu List Size	1	221015477	221015477	0.02	0.895
Iterations*Neighbors	1	108465129	108465129	0.01	0.926
Iterations*Tabu List Size	1	11588120667	11588120667	0.92	0.338
Neighbors*Tabu List Size	1	696023913	696023913	0.06	0.814
Iterations*Neighbors*Tabu List Size	1	1202252366	1202252366	0.10	0.758
Error	259	3.26340E+12	12599997398		
Lack-of-Fit	16	2.20485E+11	13780288705	1.10	0.355
Pure Error	243	3.04291E+12	12522282744		
Total	269	4.83744E+12			

5.2.2 Predictor Association

We utilize the p-value to identify significant predictors. We compare the p-value for the term to the significance level to assess the null hypothesis. The null hypothesis is that there is no association between the term and the response. According to Table 5-1 and Figure 5-1, only the first term (iterations) and its quadratic form (iterations x iterations) show a statistically significant association with total cost as its p-value is less than alpha. We can conclude that the coefficients for iterations and iterations x iterations predictors do not equal zero. In addition, the coefficients of iterations and iterations x iterations are negatively correlated to the total cost, meaning they are highly influential in reducing the total cost function.

On the other hand, all other predictors are insignificant in association with total cost. Neighbors number, the next closest p-value, misses the mark with a large p-value. While neighbors' number shows no statistically significant association with the total cost, altering the accepted significance level can allow neighbors' number to associate with the total cost significantly. Tabu list size seems to have little effect on influencing the total cost value.

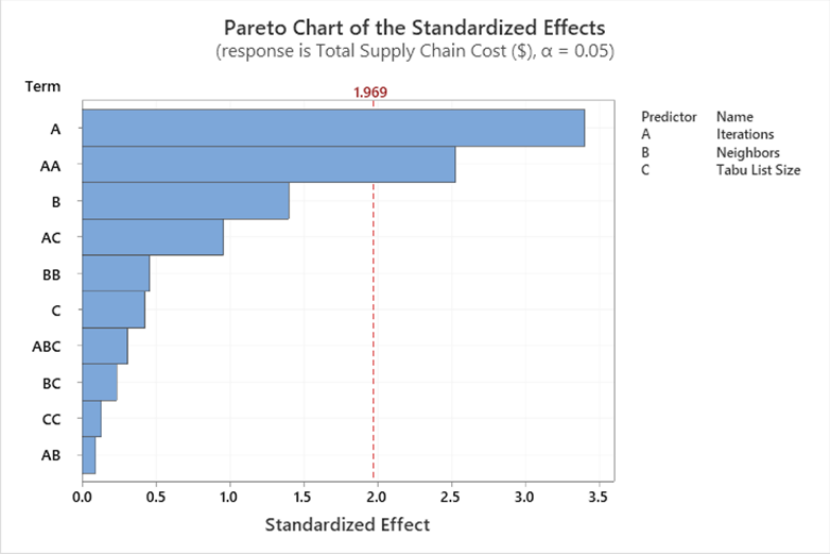

Figure 5-1: Total Supply Chain Cost Pareto Chart for Significant Predictors

None of the interaction effects are significant, so main effects and quadratic main effects become the next central area of focus. Figure 5-2 further explains with main effect plots for iterations, neighbors, and tabu list size. The graphs illustrate iterations and neighbors' contributions to reducing the total supply chain cost ($). In iterations and neighbors, the main effect plot reduces the mean total cost, while tabu list size virtually stays flat. Iterations repeatedly loop, generating more neighbors in the Tabu Search algorithm. Tabu list size serves as an external, unrelated array of continuously changing solutions, so this confirms our predictions.

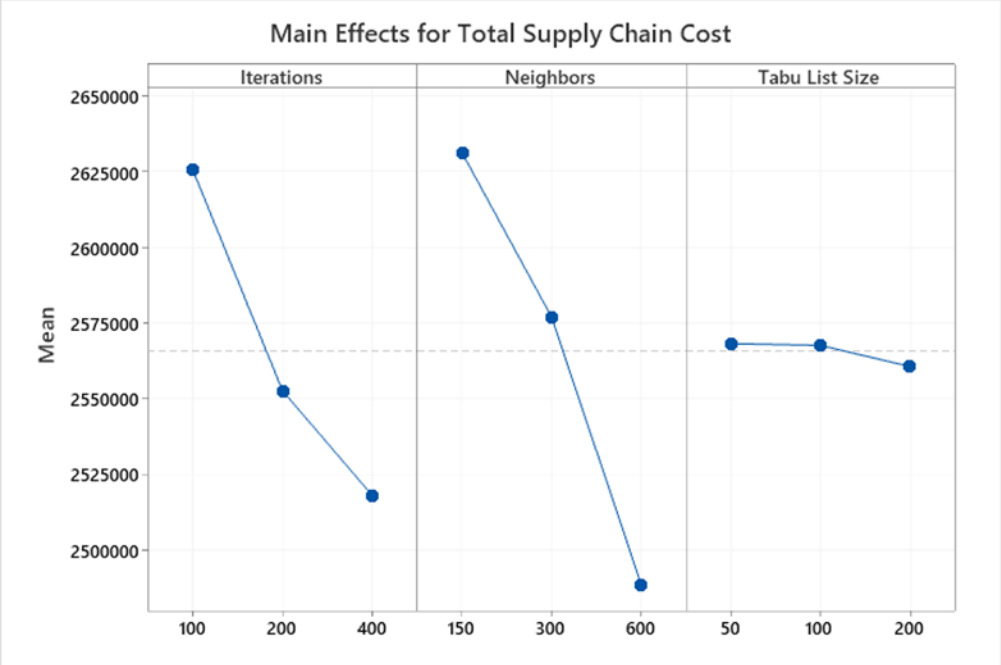

Figure 5-2: Run Time Main Effects Plot

5.2.3 VIF and Goodness of Fit

According to Table 5-1, each predictor contains a variance inflation factor (VIF). With high VIF values, we lose reliability amongst the regression results. The results display a VIF above ten which indicates a high correlation and is cause for concern. This concern applies more to prediction (not used in this paper) rather than estimation of predictors. However, we address this concern in the stepwise analysis.

Next, we look at the goodness of fit values. The goodness of fit values, $R^2 = .3254$ and $R^2(adj) = .2993$, imply that the joint presence of independent variables (iterations, neighbors, and tabu list size) explains the reported percentage of the dependent variable Y (total cost) variability in the model. The higher the percentage value, the better the

model fits the data (Frost, 2020). In a practical case, this $R^2(adj)$ value would not be acceptable as it is less than 50%. As a result, the model's small sample size does not fit the data well in the conducted experiment. As expected, $R^2(adjusted)$ is always a few percentage points lower than $R^2(theoretical)$.

5.2.4 Collinearity and Outliers

Now, we check for collinearity amongst the independent variables. When using the same 0.05 as the significance level, we see from the results in Table 5-2 that the p-values are all greater than 5%, meaning there is inconclusive evidence about the significance of the association between the variables. In addition, the near-zero correlation coefficients for all three variables do not allow us to conclude any correlation between iterations, neighbors, and tabu list size.

Table 5-2: Correlation Analysis of Independent Variables

Pairwise Pearson Correlations

Sample 1	Sample 2	N	Correlation	95% CI for ρ	P-Value
Neighbors	Iterations	270	-0.000	(-0.119, 0.119)	1.00000
Tabu List Size	Iterations	270	-0.000	(-0.119, 0.119)	1.00000
Tabu List Size	Neighbors	270	-0.000	(-0.119, 0.119)	1.00000

Following the multi-regression output, Cook's Distance provides interesting data to identify potential outliers. An outlier must have a value greater than 0.50. None of Cook's Distance values were larger than 0.50 than this value, so no outliers present. However, Minitab's regression output displays unusual observations gathered from data shown in Table 5-3. While these unusual observations diminish the validity of the regression model and skew results, they are vital data points in Tabu Search's goal to

minimize total cost. Observations with large distance values relative to other observations can be influential. Unpredictability and random variability within Tabu Search algorithms contribute as well.

Table 5-3: Unusual Observations of Total Supply Chain Cost ($)

Fits and Diagnostics for Unusual Observations

Obs	Total Supply Chain Cost ($)	Fit	Resid	Std Resid	
10	2938476	2697767	240709	2.17	R
31	2336459	2648843	-312383	-2.79	R
58	2782644	2554157	228487	2.07	R
65	2794657	2550995	243662	2.21	R
72	2326729	2597138	-270409	-2.42	R
77	2304209	2597138	-292929	-2.62	R
91	2355350	2594719	-239369	-2.12	R
119	2386180	2642380	-256200	-2.28	R
122	2879316	2632311	247005	2.20	R
124	2343080	2632311	-289230	-2.57	R
143	2251411	2499398	-247987	-2.21	R
161	2268141	2629454	-361313	-3.24	R
231	2311227	2551020	-239793	-2.16	R
269	2196313	2503795	-307482	-2.75	R

R Large residual

5.2.5 Assumptions Check

Typically, a linear regression analysis has two purposes: to predict the value of the dependent variable for individuals or to estimate the effect of some explanatory variable on the dependent variable. We do not wish to use regression to predict values but to analyze the effect on iterations, neighbors, and tabu list size on total supply chain cost ($). As a result, we must check the multi-regression model assumptions.

First, we check for linearity and additivity between the dependent and independent variables. Figure 5-4, Figure 5-5, and Figure 5-6 display residuals versus predictor

values. The three plots display symmetrically distributed data points around the horizontal line. In addition, a roughly constant variance validating linearity and additivity.

Second, we check the model's residuals normality. The residuals probability plot dissatisfies the normality of residuals assumption shown in Figure 5-3. Although most of the data points lie close to the red diagonal line, the Anderson-Darling p-value was less than 0.005, meaning the data does not come from a normal distribution. In addition, the bow-shaped pattern of deviations from the diagonal indicates that the residuals have excessive skewness (they are not symmetrically distributed, with too many large errors in one direction). The non-normality of residuals poses a significant concern for any regression model by reducing its validity. Tabu Search's essential goal of minimizing total cost may be a leading reason and the limited sample size of ten replications per combination. Therefore, we require a data transformation of the response variable to make the residuals demonstrate a normal distribution. We conducted a Box-Cox data transformation to adjust the normality of residuals' p-value to be greater than our 5% significance level (Bland & Altman, 1996). This adjustment significantly alters the statistical analysis results and will be depicted in the stepwise regression section.

Third, we check for homoscedasticity. We verify this assumption by observing the residuals versus fits plot in Figure 5-3. The data points in the plot show even distributions above and below the horizontal line.

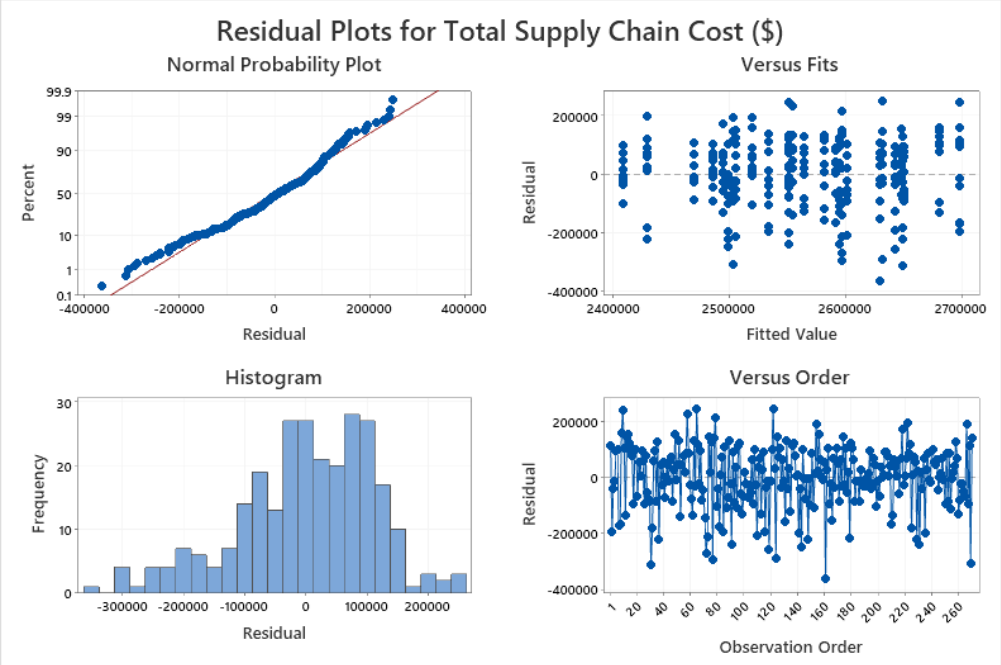

Figure 5-3: Four-in-One Plots for Total Supply Chain Cost ($)

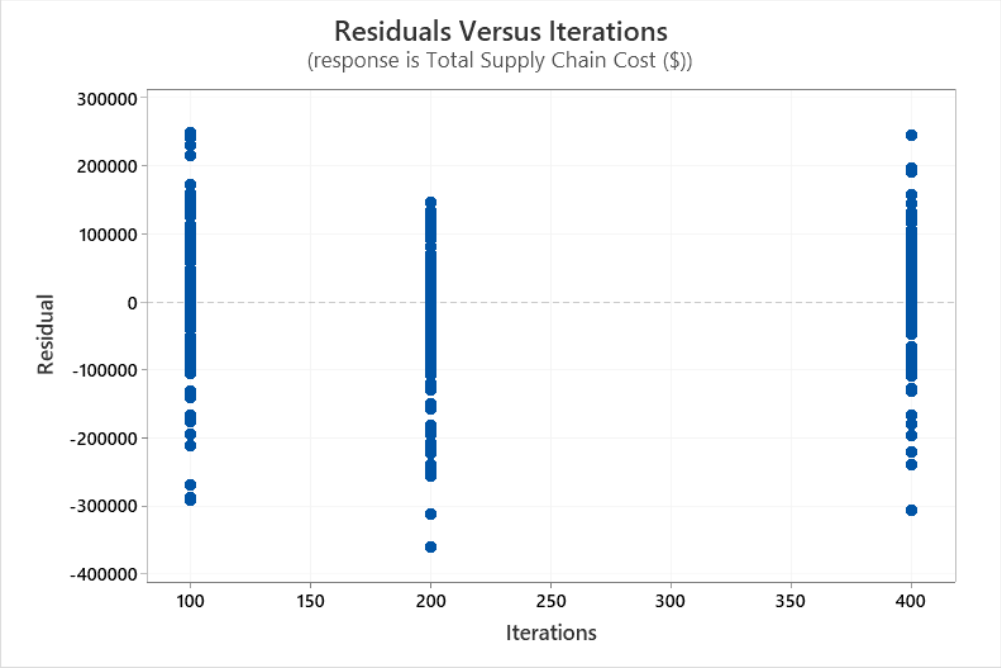

Figure 5-4: Total Supply Chain Cost Residuals vs. Iterations Predictor Plot

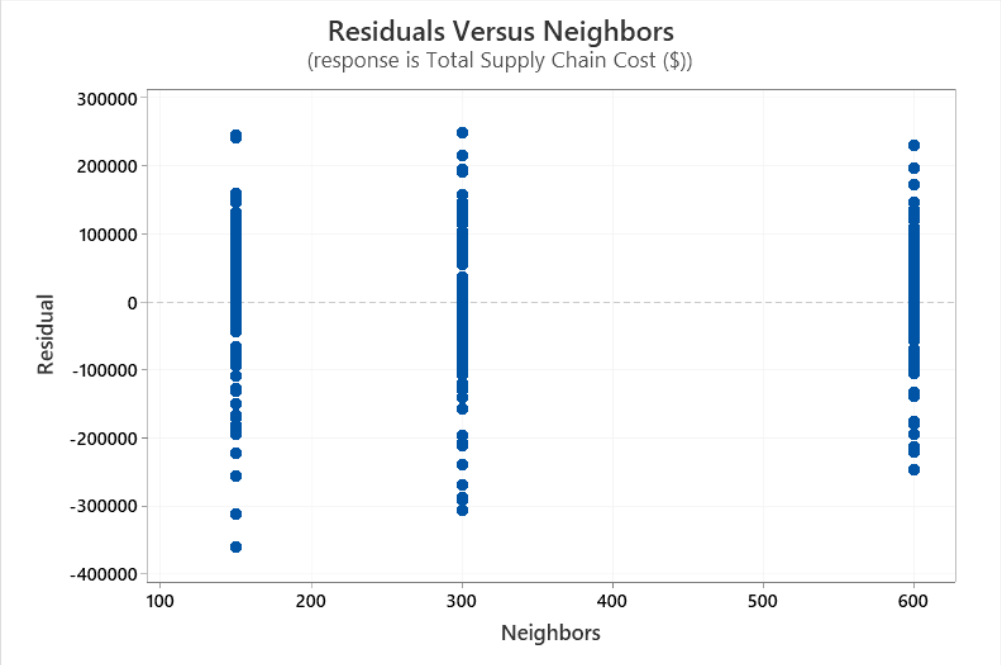

Figure 5-5: Total Supply Chain Cost Residuals vs. Neighbors Predictor Plot

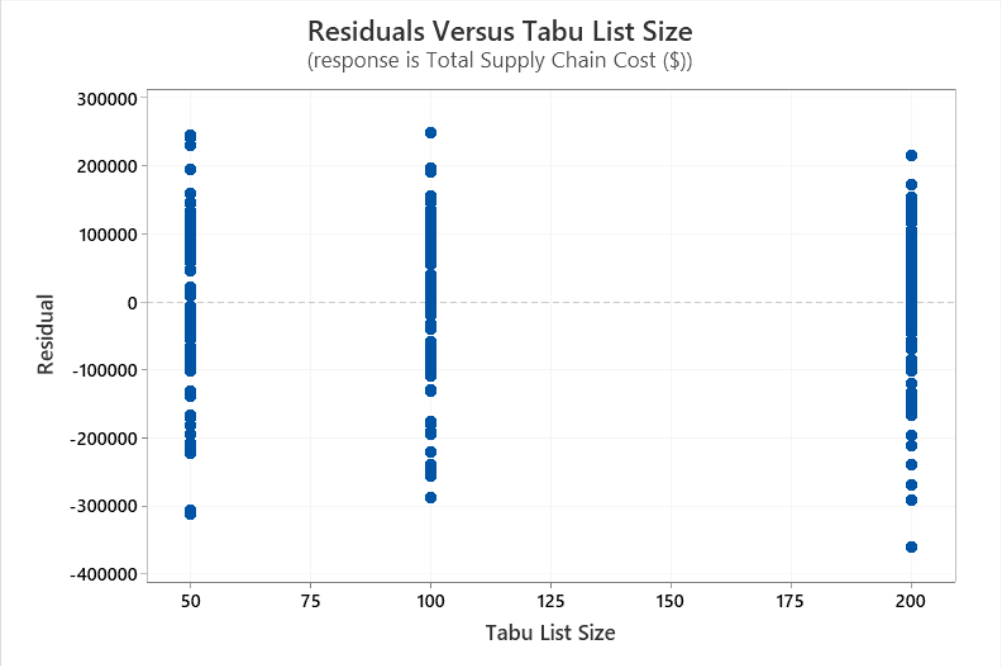

Figure 5-6: Total Supply Chain Cost Residuals vs. Tabu List Size Predictor Plot

5.2.6 Stepwise Analysis

Model reduction is an effective tool to enhance the statistical significance of a term. The elimination of statistically insignificant terms increases the precision of predictions from the model. An alpha to enter and an alpha to remove of 0.15 or 15% is recommended in the statistical significance criterion (Minitab, 2019). We apply the statistical significance criterion automatically with Minitab's algorithmic procedure, known as stepwise regression. Stepwise regression improves the validity of our regression analysis; however, it is essential to acknowledge that it does not always produce the best model. In addition, we set the stepwise to require a hierarchy model.

To normalize the residuals, we conducted a Box-Cox data transformation of the response variable, total supply chain cost ($). First, we conducted a Box-Cox data transformation using natural log (where lambda = 0), but this did not change the residuals' normality. After increasing lambda significantly (where lambda = 4.25), we obtain an Anderson-Darling normality test p-value of 0.051, meaning we fail to reject the null hypothesis that the residuals follow a normal distribution. Figure 5-7 depicts the Anderson-Darling normality results after data transformation.

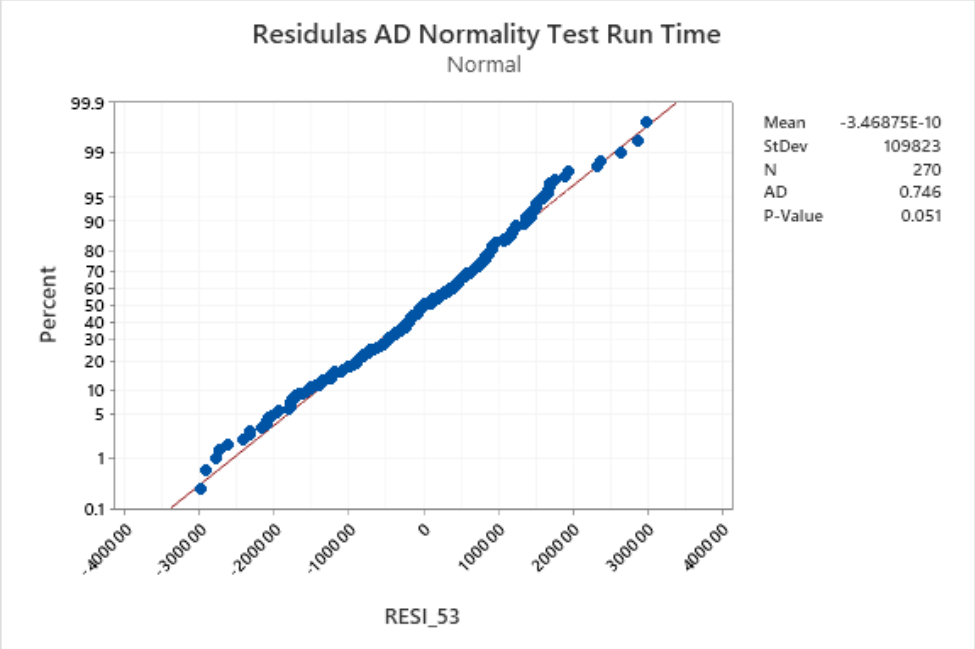

Figure 5-7: Anderson-Darling Residuals Normality Test Box-Cox Transformation

Table 5-4 and Figure 5-8 show the final multi-regression stepwise results. The algorithm chose to reduce the model down to iterations x iterations, neighbors, and iterations as the only remaining significant factors. Iterations x iterations contain a positive coefficient, while iterations and neighbors alone contain a negative coefficient. In addition, their VIF values remained the same as we expect collinearity amongst iterations and iterations x iterations. The R^2 and $R^2(adjusted)$ values increased by approximately 2% proving that the data now fits the model better than before; however, we would still consider an $R^2(adjusted)$ value lower than 50% to be unacceptable in practical use (Frost, 2020). Overall, iterations, neighbors, and iterations x iterations have a significant impact in influencing total supply chain cost ($). The stepwise analysis displays the final regression model equation in Table 5-4.

Table 5-4: Total Supply Chain Cost Stepwise Regression Analysis Results

Method

Box-Cox transformation	λ = 4.25

Stepwise Selection of Terms

α to enter = 0.15, α to remove = 0.15

The stepwise procedure added terms during the procedure in order to maintain a hierarchical model at each step.

Regression Equation

(Total Supply Chain Cost ($)^λ-1)/(λ×g^(λ-1)) = 915233 - 1412 Iterations - 321.1 Neighbors + 2.063 Iterations*Iterations

(λ = 4.25, g = 2562046 is the geometric mean of Total Supply Chain Cost ($))

Coefficients for Transformed Response

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	915233	41635	21.98	0.00000000	
Iterations	-1412	383	-3.68	0.00027739	49.00
Neighbors	-321.1	36.5	-8.80	0.00000000	1.00
Iterations*Iterations	2.063	0.738	2.80	0.00553735	49.00

Model Summary for Transformed Response

S	R-sq	R-sq(adj)	R-sq(pred)
112219	32.20%	31.44%	30.22%

Analysis of Variance for Transformed Response

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	3	1.59105E+12	5.30350E+11	42.11	0.000
Iterations	1	1.70977E+11	1.70977E+11	13.58	0.000
Neighbors	1	9.74154E+11	9.74154E+11	77.36	0.000
Iterations*Iterations	1	98506062749	98506062749	7.82	0.006
Error	266	3.34975E+12	12593033484		
Lack-of-Fit	23	3.18507E+11	13848118100	1.11	0.334
Pure Error	243	3.03124E+12	12474239467		
Total	269	4.94080E+12			

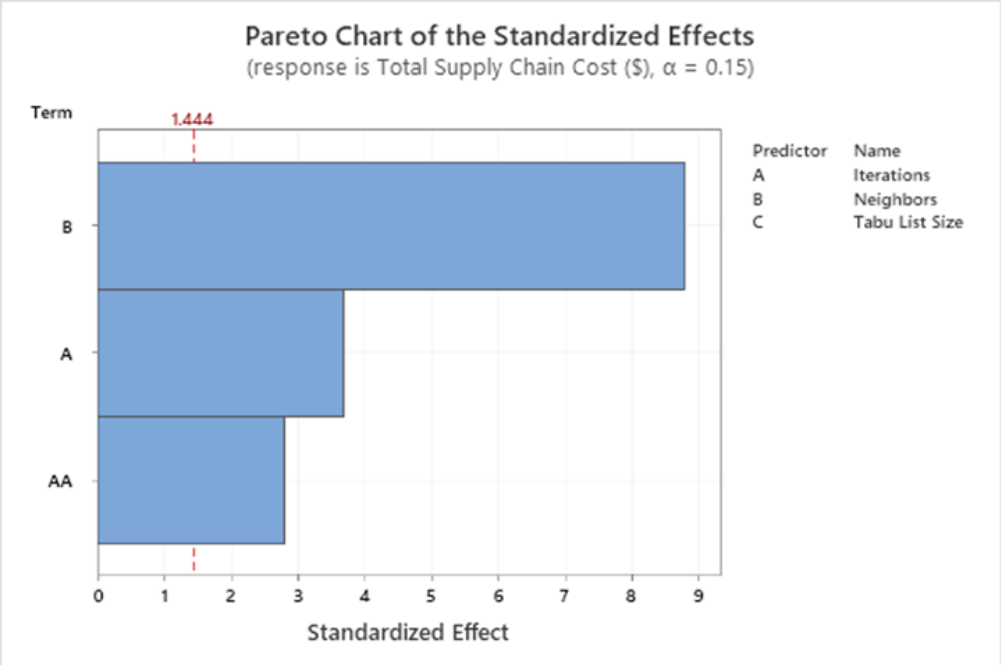

Figure 5-8: Total Supply Chain Cost Pareto Chart with Stepwise

5.3 Regression Analysis of Run Time

A quadratic multi-regression approach describes the relationship between the input parameters (iterations, neighbors' number, and tabu list size) and the response (run time (s)). The input parameters act as independent continuous predictors, while the response serves as a continuous dependent variable. Equation 30 describes the regression equation with both linear, quadratic, and interaction terms.

$$E(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_{11} X_1^2 + \beta_{22} X_2^2 + \beta_{33} X_3^2 + \beta_{12} X_1 X_2 + \beta_{13} X_1 X_3 + \beta_{23} X_2 X_3 + \beta_{123} X_1 X_2 X_3 \quad (30)$$

In Equation 30, $E(Y)$ represents the expected response value. β_0 is the constant intercept, β_i, where $i = 1, 2, 3$, are the coefficients of the non-interaction terms, β_{ii}, where $i = 1, 2, 3$, are the coefficients of the quadratic terms, β_{ij}, where $i = 1, 2, 3$ and $j = 1, 2, 3$, are the coefficients of the two-way interaction terms, and β_{ijk} where $i = 1, 2, 3$ $j =$

1, 2, 3 $k = 1, 2, 3$, are the coefficients of the three-way interaction term. $X_{i,}$ where $i = 1, 2, 3,$ represents the three Tabu Search input parameters. The method of least squares obtains the results for the regression analysis.

5.3.1 Quadratic Multi-Regression

First, we conduct a multi-regression analysis with quadratic and linear interaction terms. Then, we analyze if the regression model is significant. We use Minitab version 19 to conduct the regression analysis with a significance level (denoted as α) of 0.05. The regression's p-value is less than 5%, meaning we have a significant regression model. In addition, we test for the significance of the model's constant. The model constant's p-value is less than 5%, meaning we have a non-zero constant intercept.

Table 5-5: Run Time Regression Analysis

Regression Equation

Run Time (s) = 0.656 - 0.00174 Iterations - 0.00244 Neighbors
- 0.00020 Tabu List Size
- 0.000003 Iterations*Iterations + 0.000001 Neighbors*Neighbors
- 0.000019 Tabu List Size*Tabu List Size
+ 0.000117 Iterations*Neighbors
+ 0.000020 Iterations*Tabu List Size
+ 0.000013 Neighbors*Tabu List Size
- 0.000000 Iterations*Neighbors*Tabu List Size

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	0.656	0.575	1.14	0.255	
Iterations	-0.00174	0.00285	-0.61	0.542	68.25
Neighbors	-0.00244	0.00190	-1.28	0.201	68.25
Tabu List Size	-0.00020	0.00570	-0.04	0.972	68.25
Iterations*Iterations	-0.000003	0.000005	-0.60	0.547	49.00
Neighbors*Neighbors	0.000001	0.000002	0.59	0.555	49.00
Tabu List Size*Tabu List Size	-0.000019	0.000019	-1.01	0.315	49.00
Iterations*Neighbors	0.000117	0.000004	29.81	0.000	36.00
Iterations*Tabu List Size	0.000020	0.000012	1.72	0.087	36.00
Neighbors*Tabu List Size	0.000013	0.000008	1.66	0.099	36.00
Iterations*Neighbors*Tabu List Size	-0.000000	0.000000	-1.74	0.083	48.25

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.706531	99.09%	99.06%	99.01%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	10	14112.8	1411.28	2827.16	0.000
Iterations	1	0.2	0.19	0.37	0.542
Neighbors	1	0.8	0.82	1.65	0.201
Tabu List Size	1	0.0	0.00	0.00	0.972
Iterations*Iterations	1	0.2	0.18	0.36	0.547
Neighbors*Neighbors	1	0.2	0.17	0.35	0.555
Tabu List Size*Tabu List Size	1	0.5	0.51	1.01	0.315
Iterations*Neighbors	1	443.7	443.66	888.78	0.000
Iterations*Tabu List Size	1	1.5	1.47	2.94	0.087
Neighbors*Tabu List Size	1	1.4	1.37	2.74	0.099
Iterations*Neighbors*Tabu List Size	1	1.5	1.51	3.03	0.083
Error	259	129.3	0.50		
Lack-of-Fit	16	9.8	0.61	1.24	0.235
Pure Error	243	119.5	0.49		
Total	269	14242.1			

5.3.2 Predictor Association

We utilize the p-value to identify significant predictors. A 5% significance level indicates the percent risk of concluding that an association exists when there is no actual association. We compare the p-value for the term to the significance level to assess the null hypothesis. The null hypothesis is that there is no association between the term and the response. A significance level of 0.05 indicates a 5% risk of concluding that an association exists when there is no actual association.

According to Table 5-5 and Figure 5-9, only the first interaction term (iterations x neighbors association) shows a statistically significant association with run time (s) as its p-value is less than alpha. We can conclude that this interaction's coefficient does not equal zero. Its presence indicates that iterations on run time (s) vary at different predictor variable values (neighbors). In other words, the unique effect of iterations on run time (s) is not limited to iterations but also depends on neighbors' values.

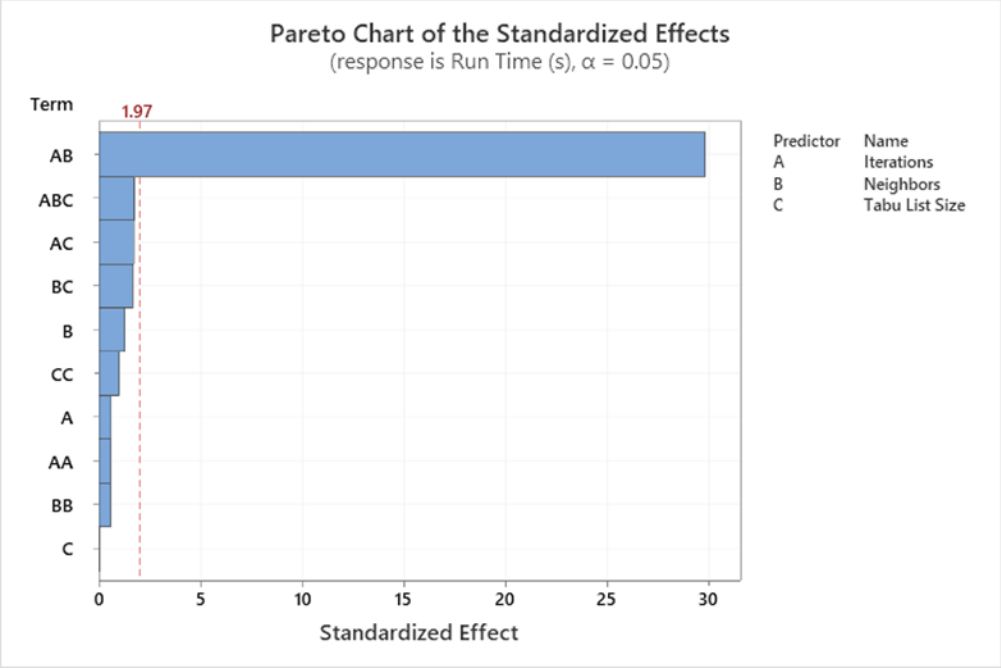

Figure 5-9: Run Time Pareto Chart for Significant Predictors

As a result of one of the interaction effects being significant, we ignore the main effects and explain the interaction effect. Figure 5-10 further illustrates an interaction plot between iterations and neighbors. The combination of iterations x neighbors' number dramatically increases the run time (s). Although other interactions also increase run time (s), iterations x neighbors impact the response exponentially because iterations loop repeatedly, generating more neighbors in the Tabu Search algorithm. Tabu list size serves as an external, unrelated array of continuously changing solutions.

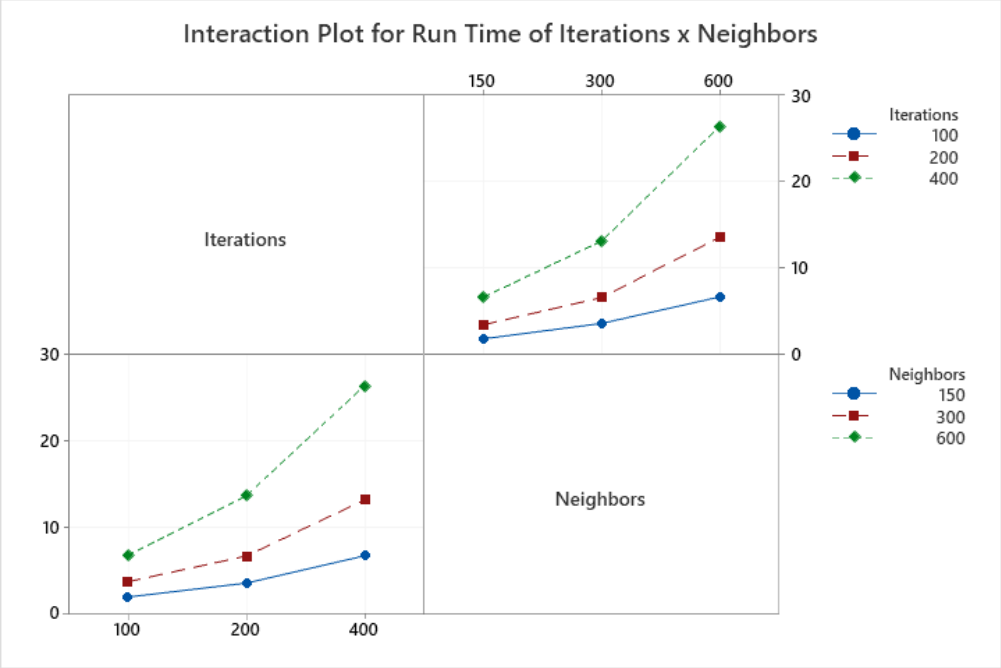

Figure 5-10: Run Time Interactions x Neighbors Interaction Plot

5.3.3 VIF and Goodness of Fit

According to Table 5-5, each predictor contains a variance inflation factor (VIF). With high VIF values, we lose reliability amongst the regression results. The results display a VIF above ten which indicates a high correlation and is cause for concern. This concern applies more to prediction (not used in this paper) rather than estimation of predictors. However, we address this concern in the stepwise analysis.

Next, we look at the goodness of fit values. The goodness of fit value, $R^2 = .9909$ and $R^2(adj) = .9906$, imply that the joint presence of independent variables (iterations, neighbors, and tabu list size) explains the reported percentage of the dependent variable Y (run time) variability in the model. The higher the percentage value, the better the model fits the data (Frost, 2020). In a practical case, this $R^2(adj)$ would be acceptable.

As a result, the model's small sample size does fit the data well in the conducted experiment. As expected, $R^2(adjusted)$ is always a few percentage points lower than $R^2(theoretical)$.

5.3.4 Collinearity and Outliers

Now, we check for collinearity amongst the independent variables. When using the same 0.05 as the significance level, we see from the results in Table 5-6 that the p-values are all greater than 5%, meaning there is inconclusive evidence about the significance of the association between the variables. In addition, the near-zero correlation coefficients for all three variables do not allow us to conclude any correlation between iterations, neighbors, and tabu list size.

Table 5-6: Correlation Analysis of Independent Variables

Pairwise Pearson Correlations

Sample 1	Sample 2	N	Correlation	95% CI for ρ	P-Value
Neighbors	Iterations	270	-0.000	(-0.119, 0.119)	1.00000
Tabu List Size	Iterations	270	-0.000	(-0.119, 0.119)	1.00000
Tabu List Size	Neighbors	270	-0.000	(-0.119, 0.119)	1.00000

Following the multi-regression output, Cook's Distance provides interesting data to identify potential outliers. An outlier must have a value greater than 0.50. None of Cook's Distance values were larger than 0.50 than this value, so no outliers present. However, Minitab's regression output displays unusual observations gathered from data shown in Table 5-7. While these unusual observations diminish the validity of the regression model and skew results, they are vital data points in Tabu Search's goal of minimizing total cost. Observations with large residual values relative to other

observations can be influential. Unpredictability and random variability within Tabu Search algorithms contribute as well.

Table 5-7: Unusual Observations of Run Time (s)

Fits and Diagnostics for Unusual Observations

Obs	Run Time (s)	Fit	Resid	Std Resid
21	5.330	3.590	1.741	2.49 R
22	5.250	3.590	1.661	2.38 R
23	5.511	3.590	1.922	2.75 R
51	9.018	6.754	2.264	3.29 R
61	9.072	6.485	2.587	3.76 R
81	9.156	6.838	2.317	3.33 R
91	9.453	6.782	2.670	3.80 R
101	9.248	6.792	2.457	3.50 R
151	14.984	13.233	1.751	2.50 R
171	15.103	13.375	1.728	2.48 R
181	28.114	26.387	1.727	2.55 R
201	9.039	6.769	2.270	3.33 R
211	9.125	7.007	2.119	3.10 R
221	28.788	26.541	2.247	3.24 R
231	14.986	13.308	1.678	2.42 R
241	15.487	13.467	2.021	2.91 R
251	9.111	6.580	2.531	3.63 R
261	14.916	13.196	1.720	2.47 R

R Large residual

5.3.5 Assumptions Check

Like total supply chain cost ($), we must check the multi-regression model assumptions for run time (s). First, we check for linearity and additivity between the dependent and independent variables. Figure 5-13 and Figure 5-14 display residuals versus predictor values. The three plots display some points not symmetrically distributed around the horizontal line. Most points lie below zero. Occasionally, long run times occurred, which may be due to external factors, such as the hardware or software environment. These factors must be acknowledged but are challenging to eliminate. Therefore, the linearity and additivity assumption restriction can be slightly relaxed.

Second, we check the model's residuals normality. The residuals probability plot dissatisfies the normality of the residuals assumption shown in Figure 5-11. Most of the data points deviate significantly from the red diagonal line. The Anderson-Darling p-value was less than 0.005, meaning the data does not come from a normal distribution. In addition, the bow-shaped pattern of deviations from the red diagonal indicates that the residuals have excessive skewness (they are not symmetrically distributed, with too many large errors in one direction). The non-normality of residuals poses a significant concern for any regression model. Tabu Search's essential goal of minimizing total cost may be a leading reason and the limited sample size of ten replications per combination. The first replication in each set of 10 replications per problem instance outputted a run time (s) result that was an unusual observation compared to the other nine replications' values. This issue stems from the files scanning and still being in the operating system's file cache system, so it does not require as much disk access as the first run (Clements & Singhal, 2013). Therefore, we identified these outliers and more to be removed because they negatively influence the data results. When eliminating 62 out of the 270 data points, the p-value of the Anderson-Darling normality test raises above 5%, meaning that the residuals now come from a normal distribution (Bland & Altman, 1996). This adjustment significantly alters the statistical analysis results and will be depicted in the stepwise regression section. Note: Data transformation methods were considered before removing outliers but did not affect making the run time (s) residuals normal.

Third, we check for homoscedasticity. The residuals versus fits plot in Figure 5-11 shows most points below or above the horizontal line, but occasionally we see long run times with large residuals. Once again, this may be due to external factors, such as the hardware or software environment.

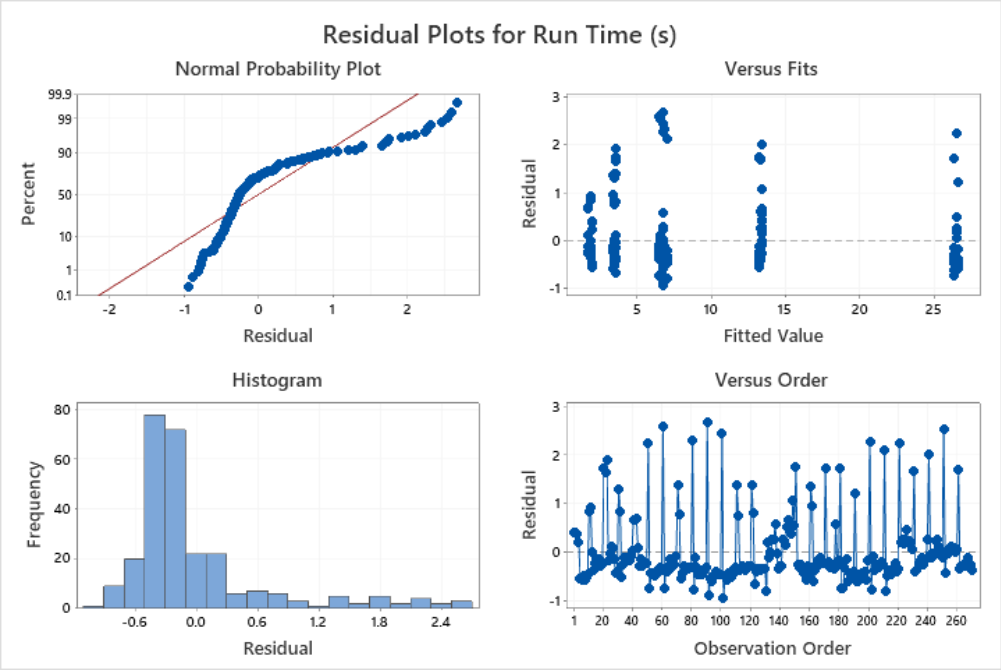

Figure 5-11: Run Time Four-in-One Plots

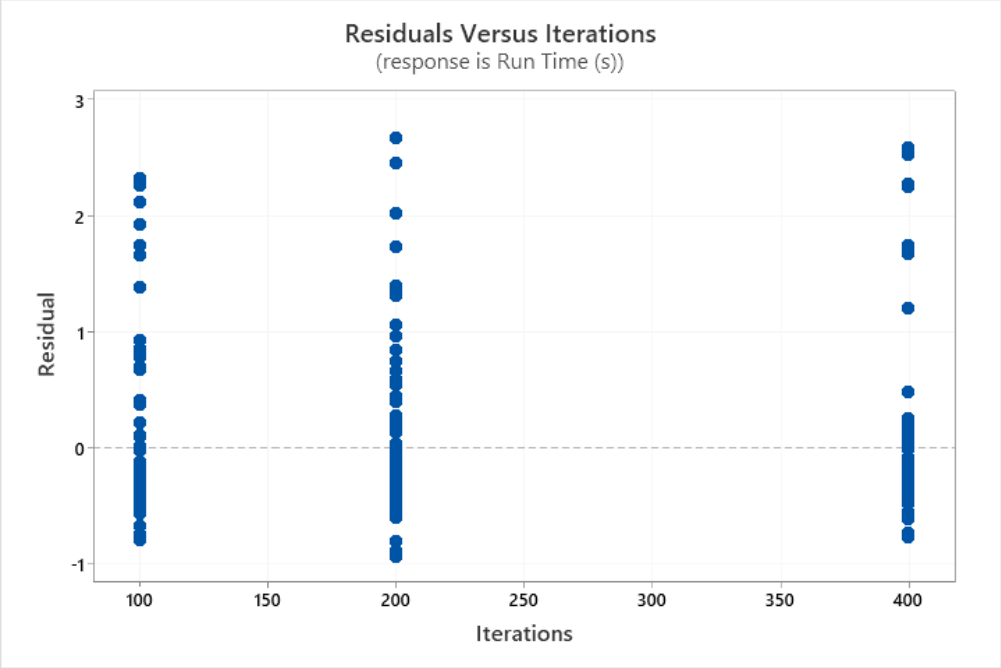

Figure 5-12: Run Time Residuals vs. Iterations Predictor Plot

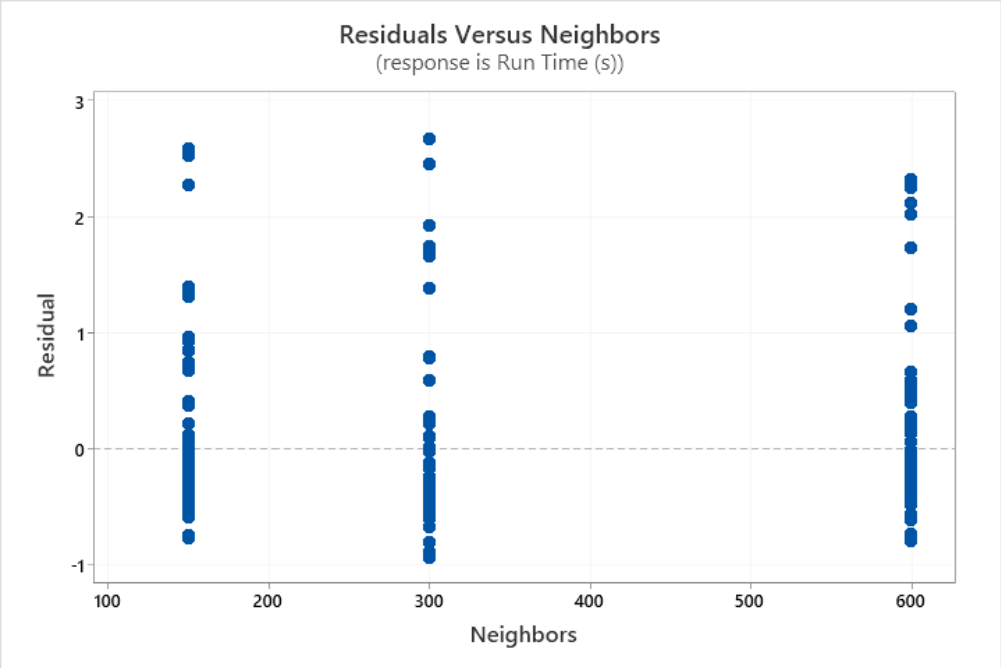

Figure 5-13: Run Time Residuals vs. Neighbors Predictor Plot

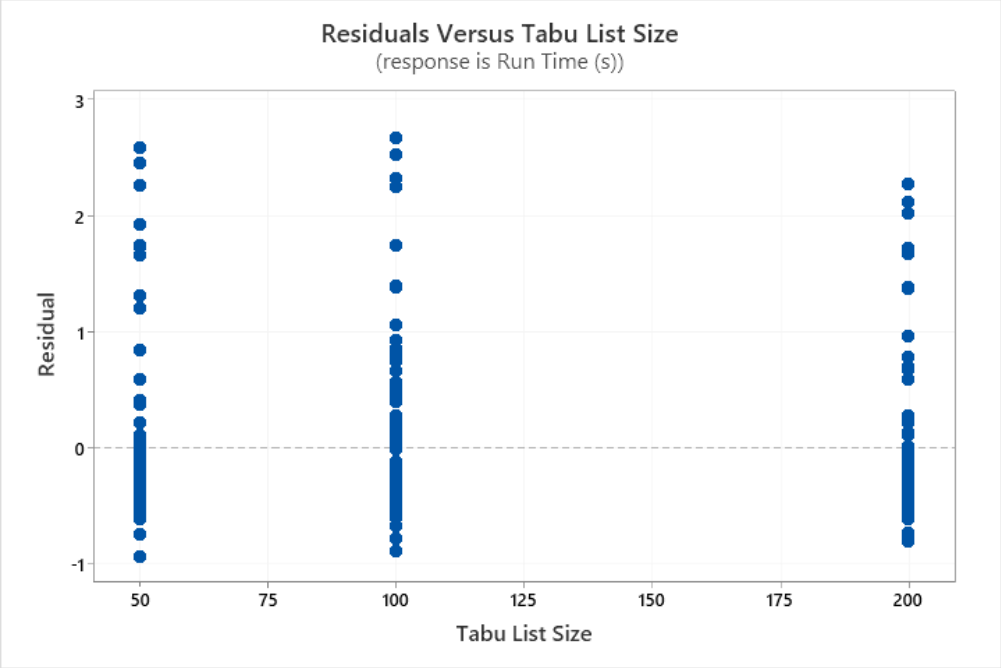

Figure 5-14: Run Time Residuals vs. Tabu List Size Predictor Plot

5.3.6 Stepwise Analysis

Like total supply chain cost ($), the run time (s) utilizes the same stepwise analysis with the same statistical significance criterion. To normalize the residuals, we eliminated 62 unusual observations out of the 270 total data points before conducting stepwise. Figure 5-15 depicts an Anderson-Darling p-value of 0.129, meaning that the run time (s) residuals now come from a normal distribution.

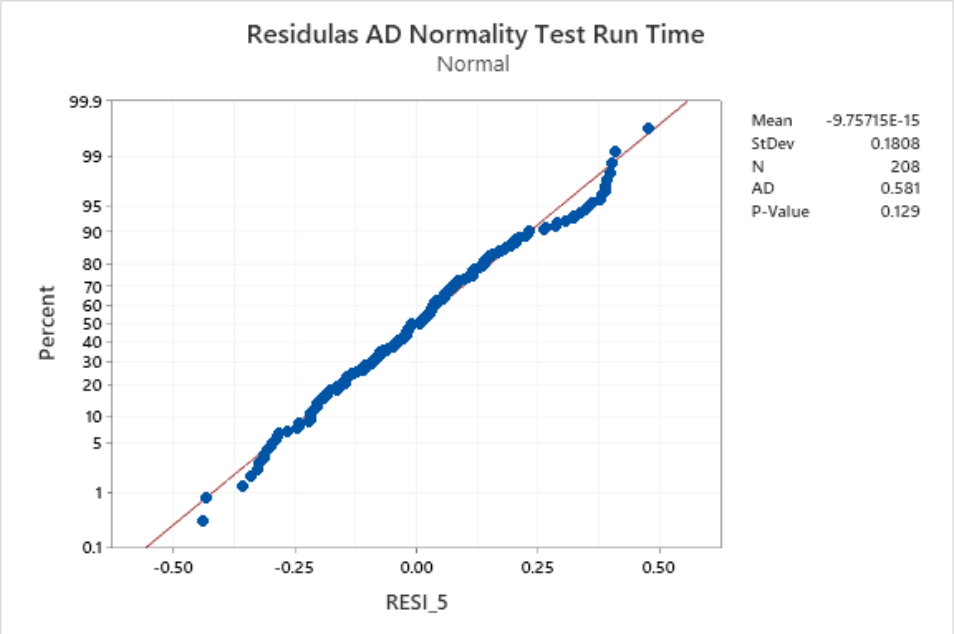

Figure 5-15: Anderson-Darling Normality Test of Residuals After Outlier Removal

Table 5-8 and Figure 5-16 show the multi-regression stepwise results. The algorithm chose to reduce the model down to the three-way interaction of iterations x neighbors x tabu list size as the remaining significant factor of the model. The hierarchy principle forces the appearance of the main and interaction effects for iterations, neighbors, and tabu list size because of the three-way interaction significance of iterations x neighbors x tabu list size. Because we removed 62 data points, the VIF values increased, and the R^2 and $R^2(adjusted)$ values remain relatively high indicating strong data that fits the regression model well (Frost, 2020). The stepwise analysis displays the final regression model equation in Table 5-8.

Table 5-8: Run Time Stepwise Regression Analysis Results

Regression Equation

Run Time (s) = 0.092 + 0.000102 Iterations - 0.003177 Neighbors
+ 0.00221 Tabu List Size
- 0.000004 Iterations*Iterations + 0.000002 Neighbors*Neighbors
- 0.000021 Tabu List Size*Tabu List Size
+ 0.000116 Iterations*Neighbors
+ 0.000015 Iterations*Tabu List Size
+ 0.000012 Neighbors*Tabu List Size
- 0.000000 Iterations*Neighbors*Tabu List Size

Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	0.092	0.184	0.50	0.616	
Iterations	0.000102	0.000889	0.11	0.909	76.33
Neighbors	-0.003177	0.000567	-5.60	0.000	68.80
Tabu List Size	0.00221	0.00176	1.25	0.211	72.57
Iterations*Iterations	-0.000004	0.000001	-3.03	0.003	52.87
Neighbors*Neighbors	0.000002	0.000001	3.85	0.000	49.99
Tabu List Size*Tabu List Size	-0.000021	0.000006	-3.83	0.000	49.31
Iterations*Neighbors	0.000116	0.000001	101.20	0.000	34.82
Iterations*Tabu List Size	0.000015	0.000004	4.12	0.000	36.32
Neighbors*Tabu List Size	0.000012	0.000002	5.29	0.000	36.43
Iterations*Neighbors*Tabu List Size	-0.000000	0.000000	-5.19	0.000	46.00

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.185377	99.94%	99.94%	99.93%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	10	11087.1	1108.71	32262.99	0.000
Iterations	1	0.0	0.00	0.01	0.909
Neighbors	1	1.1	1.08	31.35	0.000
Tabu List Size	1	0.1	0.05	1.57	0.211
Iterations*Iterations	1	0.3	0.32	9.19	0.003
Neighbors*Neighbors	1	0.5	0.51	14.83	0.000
Tabu List Size*Tabu List Size	1	0.5	0.51	14.70	0.000
Iterations*Neighbors	1	352.0	351.97	10242.18	0.000
Iterations*Tabu List Size	1	0.6	0.58	16.94	0.000
Neighbors*Tabu List Size	1	1.0	0.96	28.03	0.000
Iterations*Neighbors*Tabu List Size	1	0.9	0.92	26.90	0.000
Error	197	6.8	0.03		
Lack-of-Fit	16	4.2	0.26	18.39	0.000
Pure Error	181	2.6	0.01		
Total	207	11093.9			

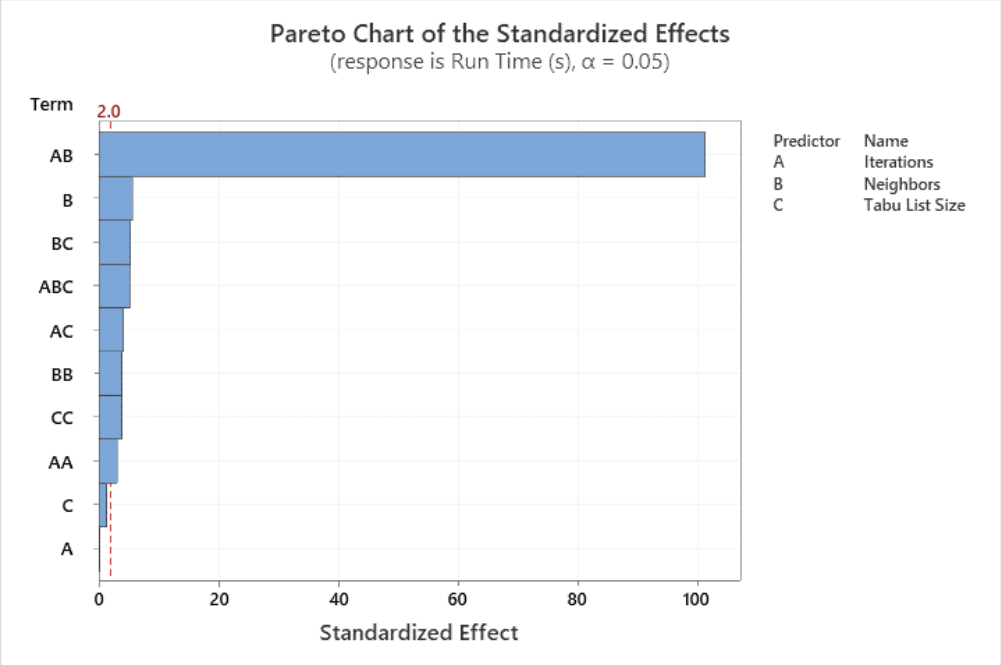

Figure 5-16: Run Time Pareto Chart with Stepwise

5.4 Statistical Analysis Key Takeaways

To summarize the steps in the statistical analysis, we first identified the significance of the regression model and its intercept coefficient. Then, we analyzed the association of the predictors with two different responses, computed various values, checked assumptions, transformed data or removed outliers, and performed stepwise analysis to improve results displaying a final regression model. After running Minitab's software, the model was significant, and we learned about potential concerns to investigate further.

First, we learned that iterations, neighbors, and iterations x iterations influence the minimization of total supply chain cost the most. Then, we learned that the interaction between iterations x neighbors x tabu list size influences the run time the most. These discoveries mean that increasing iterations and neighbors will decrease the total supply

chain cost, and increasing iterations, neighbors, and tabu list size will increase run time. From a design and practical sense, these discoveries align with our predictions and are not surprising. Iterations loop repeatedly to find more solutions and works cohesively with neighbors' number to explore new search spaces and locate more optimal solutions. Tabu list size keeps track of previously visited solutions to ensure the program makes an improving move. All inputs increase run time.

This paper's Tabu Search algorithm identifies a random route initial solution, branches to other neighboring route solutions, and repeats over a specified number of iterations. Iterations prove vital to the Tabu Search algorithm because it explores and exploits the possible search spaces providing ample attempts to obtain a good solution. In addition, neighbors' number exponentially increases the pool of potential solutions exploring search spaces never visited before. The tabu list serves as the cornerstone in constructing any Tabu Search program. The treasured tabu list prevents recently seen solutions from being revisited, saving time from a practical sense. From a statistical and design sense, all inputs are vital to the performance of the Tabu Search program. In almost any real-world scenario, paying a small price of longer run times seems like a decent tradeoff to decrease total supply chain costs.

CHAPTER 6. CONCLUSION AND FUTURE RESEARCH

This chapter provides an overall summary, a brief narrative of conclusions, contributions, and suggestions for future research.

6.1 Conclusion

The primary motivation behind this research is the rise of the COVID-19 pandemic. COVID-19 impacted every supply chain around the world. Although the virus slowed or shut down supply chains worldwide, it was highly beneficial for companies to view their supply chains differently. Overall, the pandemic accentuated the existing issues within supply chains and increased efforts in risk management to reduce costs.

This paper successfully created a multi-echelon supply chain network using domestic and global suppliers with embedded risk cost functions. A mixed-integer linear model illustrates these networks. First, the mathematical model was programmed and solved in Excel Solver for three smaller problems. Then, we modeled the problem with a Tabu Search algorithm for larger problem instances. While exact methods like Excel Solver can solve problems to optimality, they struggle with scaled problems. Given data for nodes and edges, the Tabu Search algorithm can solve any size problems and does a great job finding a quality solution amongst a large pool of possible solutions within a relatively short time. The purpose of using a commercial solver and Tabu Search was not to compare the two results of similar problem instances. Instead, the objective was to illustrate that a commercial solver like Excel Solver is incapable of solving large-scale

supply chain optimization problems. Thus, Tabu Search is a viable alternative to good results with a relatively fast run time, illustrating scalability and replicability.

Based on the statistical analysis results, we learned that iterations, neighbors, and iterations x iterations influence the minimization of total supply chain costs. Then, we learned that the interaction between iterations x neighbors x tabu list size influences the run time the most. These discoveries mean increasing iterations and neighbors will decrease the total supply chain cost, and increasing iterations, neighbors, and tabu list size will increase run time. From a design and practical sense, these discoveries align with our predictions and are not surprising. Iterations repeatedly loop, giving neighbors the chance to explore unique search spaces for more optimal solutions. In comparison, the tabu list holds the better solutions and prevents already visited solutions from being tried again. Overall, all three inputs take up time.

6.2 Contributions

This paper offers research, practical, business, and scientific contributions to current literature. The following section organizes these contributions.

6.2.1 Research

This paper makes several contributions from a research perspective:

- From a research point of view, this paper identifies gaps found in research. These gaps include using an Excel Solver linear program and Tabu Search algorithm to

solve a single-objective supply chain cost function with embedded risk and outsourcing.

- This paper relaxes many of the complex assumptions found in literature. The supply chain network problems only consider one static planning horizon. In addition, the problems only consider one SKU finished good made at a 1:1 ratio with parts. The model for this paper does not evaluate quality management principles such as scrap rate, defective parts, or logistical anomalies.
- In addition, we simplify the construction of the supply chain network leaving only suppliers (domestic and global), plants, warehouses, and markets. Literature often depicts other supply chain networks, including ones with distribution centers, retailers, and manufacturers.
- Both nodes and edges within the supply chain network incur risks and costs. A single-objective function analyzes the supply chain from a holistic perspective by summing each echelon and inter-echelon cost with embedded risk.

6.2.2 Practical

This paper makes several contributions from a practical perspective:

- This paper offers an exact solution methodology that applies to problems consisting of less than 200 decision variables from a practical perspective. Supply chain managers in the industry will have experience working with Excel Solver and easily understand the model from a high-level perspective.
- Only node data is needed to generate edge data using the edge data Python program. Both node and edge data are required to generate Tabu Search results.

The Tabu Search may not provide the most optimal results, but the program applies to any size problems.

- This work allows users to estimate the total supply chain cost of their unique supply chain network efficiently and effectively. It also enables supply chain managers to analyze specific level costs higher or lower than expected.
- Also, the mathematical model allows supply chain managers to pick more advantageous suppliers in terms of cost or risk. The same applies to plants and warehouses.
- This paper offers the ability for users to adjust for risk depending on external factors impacting their supply chain. Users can increase risk percentages to reflect a buoyant economy or decrease risk percentages to reflect a struggling economy.
- Lastly, the multi-regression statistical analysis informs programmers of the most critical Tabu Search input parameters. A programmer should allocate equal amounts of time in practice when incorporating iterations, neighbors, and tabu list size.

6.2.3 Business

This paper makes several contributions from a business perspective:

- Various departments within a company can utilize the paper's work. First, the supply chain department can use the supplier selection process to identify cheaper and more dependable suppliers in the production of their products. In addition, the product development team plays a key role as they determine the product build complexity. As the complexity of the product increases, the business will require

more dependable suppliers, but for a simple product, the cheapest supplier may suffice. Product development determines a resilient process to reduce the time to market for products to remain competitive.

- Second, businesses could integrate the logistics and inventory management departments. The logistics team could identify cheaper or more dependable transportation methods to transfer material/products between echelons efficiently. In addition, inventory management is concerned with ensuring the right stock at the right cost and time. Lastly, the inventory management team could identify warehousing solutions to reduce work-in-process (WIP) or inventory holding costs and increase salvage values or inventory turns.
- Third, customer service management interprets the relationship between a company and its customers. This department ties into this paper's supply chain principles because they act as the main source of customer information providing real-time information for product availability through its supply chain interfaces. Successful organizations establish and maintain customer rapport and induce positive feelings in customer purchases.

6.2.4 Scientific

This paper makes several contributions from a scientific perspective:

- This paper embeds risk management into a single-objective total supply chain cost function with a Tabu Search solution.
- In addition, this paper contributes to outsourcing by considering two different types of suppliers: domestic and global suppliers, each with their unique cost

function calculations and data parameters. Global suppliers carry higher risk and lower material costs than domestic suppliers with lower risk but higher material costs.

- Also, this paper develops several test problems to investigate the improvements in total supply cost and run time. We accomplish this by running small, medium, and large problem sizes with varying Tabu Search input parameters to identify cost savings and run time increases.
- Test results and statistical analysis of the metaheuristic's performance are numerically and mathematically interesting. For this paper's model only, increasing iterations and neighbors will decrease the total supply chain cost, and increasing iterations, neighbors, and tabu list size will increase run time.

6.3 Future Research

A few future research directions could expand the contributions presented in this paper:

- In this paper's studied problems, only one SKU of product moves from level to level. Only one part is needed to manufacture the one finished good. It would be interesting to expand on this assumption and perform a similar analysis on a supply chain network with multiple parts and finished goods.
- Only suppliers (domestic and global), plants, warehouses, and markets make up the supply chain network in this paper. It would be interesting also to consider other supply chain states such as retailers, distribution centers, or manufacturers.
- 100% of materials/goods transfer to the next echelon in the supply chain network in the studied problems. It would be interesting to account for quality, specifically

incorporating scrapped or missing transit parts to simulate real-world logistical issues.

- The model in this paper identifies supply chain network paths based on the cost it incurs to take such a path. It would be interesting to assume that the cost for multiple routes is equal, and some other deciding factor must be considered, such as loyalty, convenience, or locality of products.
- The solution methodologies chosen to solve this problem include Excel Solver and Tabu Search. It would be interesting to solve this problem with other methods such as CPLEX, Gurobi, LINGO, MATLAB, or SAS combined with simulation, simulated annealing, genetic algorithm, or particle Swarm optimization. In addition, a results comparison of alternative strategies with Tabu Search would be a valuable contribution.

REFERENCES

[1] AMD Ryzen™ 7 4700U Specs. Advanced Micro Devices. (2020, June 1). https://www.amd.com/en/products/apu/amd-ryzen-7-4700u.

[2] Aqlan, F., & Lam, S. S. (2015). A fuzzy-based integrated framework for supply chain risk assessment. International Journal of Production Economics, 161, 54–63. https://doi.org/10.1016/j.ijpe.2014.11.013

[3] Behzadi, G., O'Sullivan, M. J., Olsen, T. L., & Zhang, A. (2018). Agribusiness supply chain risk management: A review of quantitative decision models. Omega, 79, 21–42. https://doi.org/10.1016/j.omega.2017.07.005

[4] Behzadi, G., O'Sullivan, M. J., Olsen, T. L., Scrimgeour, F., & Zhang, A. (2017). Robust and resilient strategies for managing supply disruptions in an agribusiness supply chain. International Journal of Production Economics, 191, 207–220. https://doi.org/10.1016/j.ijpe.2017.06.018

[5] Bland, J. M., & Altman, D. G. (1996). Transformations, means, and confidence intervals. BMJ (Clinical research ed.), 312(7038), 1079. https://doi.org/10.1136/bmj.312.7038.1079

[6] Braido, G. M., Borenstein, D., & Casalinho, G. D. (2016). Supply chain network optimization using a Tabu Search based heuristic. Gestão & Produção, 23(1), 3- 17.

[7] C. Lee, Y. Yeung, and Z. Hong, (2018). "Managing the risks of outsourcing in supply chain networks," First International Technology Management Conference, San Jose, CA, 2011, pp. 488-494, doi: 10.1109/ITMC.2011.5996017.

[8] Cha, H., Pingry, D., & Thatcher, M. (2008). Managing the Knowledge Supply Chain: An Organizational Learning Model of Information Technology Offshore Outsourcing. MIS Quarterly, 32(2), 281. https://doi.org/10.2307/25148841

[9] Clements, J., & Singhal, P. (2013, October 30). Why do Python programs run very slow the first time? [web log]. https://stackoverflow.com/questions/19684408/why-do-python-programs-run- very-slow-the-first-time.

[10] Fatehi Kivi, A., Mehdizadeh, E., Tavakkoli-Moghaddam, R., & Najafi, S. E. (2021). Solving a Multi-Item Supply Chain Network Problem by Three Meta-heuristic Algorithms. Journal of Optimization in Industrial Engineering, 14(2), 145-151.

[11] Frost, J. (2020, April 23). How To Interpret R-squared in Regression Analysis. Statistics by Jim. https://statisticsbyjim.com/regression/interpret-r-squared-regression/.

[12] Gendreau, M. (2003). An Introduction to Tabu Search in Handbook of Metaheuristics (pp. 37-54). Springer, Boston, MA.

[13] Glover, F., Laguna, M., & Marti, R. (2007). Principles of tabu search. Approximation algorithms and metaheuristics, 23, 1-12.

[14] González-Zapatero, C., González-Benito, J., Lannelongue, G., & Ferreira, L. M. (2020). Using fit perspectives to explain supply chain risk management efficacy. International Journal of Production Research, 1–12. https://doi.org/10.1080/00207543.2020.1776412

[15] Heckmann, I., Comes, T., & Nickel, S. (2015). A critical review on supply chain risk – Definition, measure, and modeling. Omega, 52, 119–132. https://doi.org/10.1016/j.omega.2014.10.004

[16] Hernandez, D. F., & Haddud, A. (2018). Value creation via supply chain risk management in global fashion organizations outsourcing production to China. Journal of Global Operations and Strategic Sourcing, 11(2), 250–272. https://doi.org/10.1108/jgoss-09-2017-0037

[17] Kasyanenko, S. (2019, July 22). iPhone Made in India: Apple Outsourcing Strategy. Medium. https://medium.com/@Ralabs/iphone-made-in-india-apple-outsourcing- strategy-fff490580cf9.

[18] König, A., & Spinler, S. (2016). The effect of logistics outsourcing on the supply chain vulnerability of shippers. The International Journal of Logistics Management, 27(1), 122–141. https://doi.org/10.1108/ijlm-03-2014-0043

[19] Kouvelis, P., & Milner, J. M. (2002). Supply chain capacity and outsourcing decisions: the dynamic interplay of demand and supply uncertainty. IIE Transactions, 34(8), 717–728. https://doi.org/10.1080/07408170208928907

[20] Kroes, J. R., & Ghosh, S. (2009). Outsourcing congruence with competitive priorities: Impact on supply chain and firm performance. Journal of Operations Management, 28(2), 124–143. https://doi.org/10.1016/j.jom.2009.09.004

[21] Kumar, S. K., Tiwari, M. K., & Babiceanu, R. F. (2010). Minimization of supply chain cost with embedded risk using computational intelligence

approaches. International Journal of Production Research, 48(13), 3717-3739.

[22] Lee, K., & Ozsen, L. (2020). Tabu search heuristic for the network design model with lead time and safety stock considerations. Computers & Industrial Engineering, 148, 106717.

[23] Lee, Y. H., & Kwon, S. G. (2010). The hybrid planning algorithm for the distribution center operation using tabu search and decomposed optimization. Expert systems with applications, 37(4), 3094-3103.

[24] Lee, Y. H., Jeong, C. S., & Moon, C. (2002). Advanced planning and scheduling with outsourcing in the manufacturing supply chain. Computers & Industrial Engineering, 43(1-2), 351–374. https://doi.org/10.1016/s0360-8352(02)00079-7

[25] Liang, F. (2020, July 27). Optimization Techniques - Tabu Search. https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25.

[26] Melo, M. T., Nickel, S., & Saldanha-da-Gama, F. (2012). A tabu search heuristic for redesigning a multi-echelon supply chain network over a planning horizon. International Journal of Production Economics, 136(1), 218-230.

[27] Data Analysis, Statistical & Process Improvement Tools. Minitab. (2019, January 19). https://www.minitab.com/en-us/.

[28] Mohammed, A. M., & Duffuaa, S. O. (2020). A tabu search based algorithm for the optimal design of multi-objective multi-product supply chain networks. Expert Systems with Applications, 140, 112808.

[29] Mohammed, A., & Duffuaa, S. (2019, January). A meta-heuristic algorithm based on simulated annealing for designing multi-objective supply chain systems. In 2019 Industrial & systems engineering conference (ISEC) (pp. 1-6). IEEE.

[30] Mohib, A. M. N., & Deif, A. M. (2019). Supply chain multi-state risk assessment using universal generating function. Production Planning & Control, 31(9), 699–708. https://doi.org/10.1080/09537287.2019.1680891

[31] Mokrini, A. E., Dafaoui, E., Berrado, A., & Mhamedi, A. E. (2016). An approach to risk Assessment for Outsourcing Logistics: Case of Pharmaceutical Industry. IFAC-PapersOnLine, 49(12), 1239–1244. https://doi.org/10.1016/j.ifacol.2016.07.681

[32] Olson, D. L., & Wu, D. (2011). Risk management models for supply chain: a scenario analysis of outsourcing to China. Supply Chain Management: An International Journal, 16(6), 401–408. https://doi.org/10.1108/13598541111171110

[33] Python Release 3.9.1. Python.org. (2020, December 7). https://www.python.org/downloads/release/python-391/.

[34] Ravindran, A. R., & Warsing, D. P. (2017). Supply chain engineering: models and applications. CRC Press.

[35] Shahraki, M., & Sharifi, A. (2019). Multi-period Multi-level Supply Chain Network Design in Agile Manufacturing with Tabu Search Algorithm. Roshd-E-Fanavari. https://www.sid.ir/en/journal/ViewPaper.aspx?id=666076

[36] Shangquan, G. (2000). Economic globalization: trends, risks and risk prevention. Economic & Social Affairs, CDP Background Paper, 1, 1-8.

[37] Solver Add-in in Excel. Microsoft Office Support. (2019). https://support.microsoft.com/en-us/office/load-the-solver-add-in-in-excel-612926fc-d53b-46b4-872c-e24772f078ca#OfficeVersion=Windows.

[38] Williamson, O. E. (2008). Outsourcing: Transaction Cost Economics and Supply Chain Management. The Journal of Supply Chain Management, 44(2), 5–16. https://doi.org/10.1111/j.1745-493x.2008.00051.

[39] Yan, B., Wang, X., & Shi, P. (2017). Risk assessment and control of agricultural supply chains under Internet of Things. Agrekon, 56(1), 1–12. https://doi.org/10.1080/03031853.2017.1284680.

APPENDICES

This section displays pertinent information in creating this paper's solution methodologies, including complete Tabu Search statistical analysis results, Tabu Search Python code, and the Edge Data Generator Python code used in this paper's solution methodologies.

A. Tabu Search Results for Statistical Analysis

Problem Instance	Iterations	Neighbors	Tabu List Size	Total Supply Chain Cost ($)	Run Time (s)
3.1.1	100	150	50	$ 2,810,381.94	2.422032
3.1.2	100	150	50	$ 2,503,313.15	2.422190
3.1.3	100	150	50	$ 2,658,991.69	2.390801
3.1.4	100	150	50	$ 2,683,623.54	2.227513
3.1.5	100	150	50	$ 2,791,746.96	1.463760
3.1.6	100	150	50	$ 2,798,885.56	1.454639
3.1.7	100	150	50	$ 2,527,585.65	1.540746
3.1.8	100	150	50	$ 2,530,405.61	1.456596
3.1.9	100	150	50	$ 2,856,329.46	1.500123
3.1.10	100	150	50	$ 2,938,476.14	1.552166
3.2.1	100	150	100	$ 2,785,597.27	2.770565
3.2.2	100	150	100	$ 2,548,811.82	2.844427
3.2.3	100	150	100	$ 2,826,139.63	1.928448
3.2.4	100	150	100	$ 2,836,203.61	1.564704
3.2.5	100	150	100	$ 2,806,060.04	1.702705
3.2.6	100	150	100	$ 2,779,321.15	1.793548
3.2.7	100	150	100	$ 2,757,536.76	1.725237
3.2.8	100	150	100	$ 2,585,702.81	1.779196
3.2.9	100	150	100	$ 2,781,563.78	1.626938
3.2.10	100	150	100	$ 2,783,214.94	1.659796
3.3.1	100	300	50	$ 2,583,148.07	5.330456
3.3.2	100	300	50	$ 2,707,031.07	5.250223
3.3.3	100	300	50	$ 2,656,928.18	5.511345
3.3.4	100	300	50	$ 2,743,718.28	3.432258
3.3.5	100	300	50	$ 2,710,679.33	3.566567
3.3.6	100	300	50	$ 2,557,045.94	3.709438
3.3.7	100	300	50	$ 2,727,201.85	3.694258

3.3.8	100	300	50	$	2,597,751.23	3.444149
3.3.9	100	300	50	$	2,574,924.56	3.181881
3.3.10	100	300	50	$	2,566,719.43	3.207320
3.4.1	200	150	50	$	2,336,459.41	4.812177
3.4.2	200	150	50	$	2,467,533.42	4.346443
3.4.3	200	150	50	$	2,707,441.90	2.998491
3.4.4	200	150	50	$	2,743,199.61	3.250366
3.4.5	200	150	50	$	2,565,692.35	3.391022
3.4.6	200	150	50	$	2,777,280.82	3.334674
3.4.7	200	150	50	$	2,426,568.84	3.401417
3.4.8	200	150	50	$	2,693,755.28	3.331921
3.4.9	200	150	50	$	2,622,745.61	3.429552
3.4.10	200	150	50	$	2,706,258.75	3.540490
3.5.1	100	150	200	$	2,577,641.02	2.406997
3.5.2	100	150	200	$	2,649,209.04	2.411347
3.5.3	100	150	200	$	2,694,720.76	2.440887
3.5.4	100	150	200	$	2,634,197.62	1.844236
3.5.5	100	150	200	$	2,709,502.57	1.462712
3.5.6	100	150	200	$	2,722,061.14	1.603703
3.5.7	100	150	200	$	2,562,377.81	1.449252
3.5.8	100	150	200	$	2,676,152.70	1.494683
3.5.9	100	150	200	$	2,801,060.77	1.509804
3.5.10	100	150	200	$	2,640,666.65	1.525093
3.6.1	100	600	50	$	2,598,842.62	9.018140
3.6.2	100	600	50	$	2,686,933.38	6.010422
3.6.3	100	600	50	$	2,413,944.04	6.332910
3.6.4	100	600	50	$	2,598,368.78	6.328052
3.6.5	100	600	50	$	2,632,535.62	6.275338
3.6.6	100	600	50	$	2,573,282.19	6.349621
3.6.7	100	600	50	$	2,640,776.15	6.351651
3.6.8	100	600	50	$	2,782,643.76	6.373649
3.6.9	100	600	50	$	2,641,375.05	6.440240
3.6.10	100	600	50	$	2,528,329.21	6.383341
3.7.1	400	150	50	$	2,474,724.20	9.072139
3.7.2	400	150	50	$	2,418,617.64	5.743211
3.7.3	400	150	50	$	2,682,122.49	6.124894
3.7.4	400	150	50	$	2,518,730.90	6.108091
3.7.5	400	150	50	$	2,794,656.79	6.196726
3.7.6	400	150	50	$	2,668,468.29	6.340147
3.7.7	400	150	50	$	2,530,315.25	6.189712
3.7.8	400	150	50	$	2,645,657.61	6.330341
3.7.9	400	150	50	$	2,471,161.80	6.244940
3.7.10	400	150	50	$	2,507,308.80	6.401817

3.8.1	100	300	200	$	2,455,601.31	4.873938
3.8.2	100	300	200	$	2,326,728.62	4.282253
3.8.3	100	300	200	$	2,384,932.09	2.955189
3.8.4	100	300	200	$	2,743,330.71	3.064344
3.8.5	100	300	200	$	2,617,624.39	3.211437
3.8.6	100	300	200	$	2,726,602.44	3.188015
3.8.7	100	300	200	$	2,304,209.31	3.195366
3.8.8	100	300	200	$	2,738,844.15	3.170059
3.8.9	100	300	200	$	2,811,122.80	3.193635
3.8.10	100	300	200	$	2,495,153.01	3.176357
3.9.1	100	600	100	$	2,492,896.13	9.155627
3.9.2	100	600	100	$	2,357,303.25	6.067322
3.9.3	100	600	100	$	2,546,345.13	6.729661
3.9.4	100	600	100	$	2,608,123.96	6.541926
3.9.5	100	600	100	$	2,339,061.81	6.387528
3.9.6	100	600	100	$	2,643,026.42	6.385821
3.9.7	100	600	100	$	2,427,653.43	6.401028
3.9.8	100	600	100	$	2,463,567.20	6.435246
3.9.9	100	600	100	$	2,668,377.40	6.512711
3.9.10	100	600	100	$	2,513,287.41	6.541953
3.10.1	200	300	100	$	2,355,349.99	9.452547
3.10.2	200	300	100	$	2,688,100.00	5.904876
3.10.3	200	300	100	$	2,505,374.00	6.254898
3.10.4	200	300	100	$	2,631,223.46	6.186296
3.10.5	200	300	100	$	2,487,275.44	6.276699
3.10.6	200	300	100	$	2,709,000.73	6.346617
3.10.7	200	300	100	$	2,716,718.89	6.403676
3.10.8	200	300	100	$	2,535,900.94	6.332244
3.10.9	200	300	100	$	2,465,511.87	6.341187
3.10.10	200	300	100	$	2,526,252.42	6.324001
3.11.1	200	300	50	$	2,622,660.16	9.248458
3.11.2	200	300	50	$	2,578,416.93	5.855817
3.11.3	200	300	50	$	2,528,363.45	6.315549
3.11.4	200	300	50	$	2,575,598.12	6.234230
3.11.5	200	300	50	$	2,547,165.11	6.371948
3.11.6	200	300	50	$	2,663,986.17	6.290070
3.11.7	200	300	50	$	2,507,939.47	6.290531
3.11.8	200	300	50	$	2,702,915.78	6.434482
3.11.9	200	300	50	$	2,573,957.71	6.382848
3.11.10	200	300	50	$	2,393,401.16	6.485961
3.12.1	200	150	100	$	2,679,370.52	4.864567
3.12.2	200	150	100	$	2,709,499.23	4.218264
3.12.3	200	150	100	$	2,514,521.60	3.114843

3.12.4	200	150	100	$	2,671,714.35	3.151644
3.12.5	200	150	100	$	2,733,057.21	3.192018
3.12.6	200	150	100	$	2,450,673.57	3.156667
3.12.7	200	150	100	$	2,605,927.14	3.119183
3.12.8	200	150	100	$	2,627,852.19	3.179243
3.12.9	200	150	100	$	2,386,180.25	3.203338
3.12.10	200	150	100	$	2,632,026.80	3.208221
3.13.1	100	300	100	$	2,733,885.78	4.946941
3.13.2	100	300	100	$	2,879,315.60	4.360466
3.13.3	100	300	100	$	2,665,578.70	2.896233
3.13.4	100	300	100	$	2,343,080.42	3.171965
3.13.5	100	300	100	$	2,776,589.56	3.236274
3.13.6	100	300	100	$	2,705,236.45	3.201621
3.13.7	100	300	100	$	2,735,510.91	3.192358
3.13.8	100	300	100	$	2,598,266.02	3.219644
3.13.9	100	300	100	$	2,697,958.03	3.212792
3.13.10	100	300	100	$	2,618,188.05	3.212682
3.14.1	200	300	200	$	2,424,195.27	5.966114
3.14.2	200	300	200	$	2,713,630.72	6.989409
3.14.3	200	300	200	$	2,541,561.17	6.654664
3.14.4	200	300	200	$	2,613,425.84	6.777885
3.14.5	200	300	200	$	2,461,820.66	7.042371
3.14.6	200	300	200	$	2,696,927.70	7.033228
3.14.7	200	300	200	$	2,706,361.65	7.357241
3.14.8	200	300	200	$	2,661,452.84	6.738442
3.14.9	200	300	200	$	2,586,789.90	6.417601
3.14.10	200	300	200	$	2,589,789.62	6.487606
3.15.1	200	600	100	$	2,303,832.37	13.140545
3.15.2	200	600	100	$	2,461,927.07	13.686624
3.15.3	200	600	100	$	2,251,410.87	13.658180
3.15.4	200	600	100	$	2,601,904.00	13.568366
3.15.5	200	600	100	$	2,498,018.57	13.945855
3.15.6	200	600	100	$	2,424,001.20	14.075522
3.15.7	200	600	100	$	2,494,949.96	13.849741
3.15.8	200	600	100	$	2,278,009.35	13.802432
3.15.9	200	600	100	$	2,496,427.30	14.473623
3.15.10	200	600	100	$	2,399,554.38	13.974960
3.16.1	400	300	100	$	2,608,371.93	14.983908
3.16.2	400	300	100	$	2,412,999.71	12.996577
3.16.3	400	300	100	$	2,532,628.90	12.918465
3.16.4	400	300	100	$	2,709,477.49	12.989075
3.16.5	400	300	100	$	2,542,261.21	12.968303
3.16.6	400	300	100	$	2,675,507.76	12.804771

3.16.7	400	300	100	$	2,583,572.57	12.794361
3.16.8	400	300	100	$	2,520,342.23	12.683185
3.16.9	400	300	100	$	2,512,246.18	12.962710
3.16.10	400	300	100	$	2,572,986.85	12.767680
3.17.1	200	150	200	$	2,268,141.13	4.788189
3.17.2	200	150	200	$	2,658,704.26	4.372335
3.17.3	200	150	200	$	2,734,683.03	2.824300
3.17.4	200	150	200	$	2,536,923.09	3.048129
3.17.5	200	150	200	$	2,609,190.88	3.175019
3.17.6	200	150	200	$	2,478,897.16	3.246758
3.17.7	200	150	200	$	2,698,681.32	3.173235
3.17.8	200	150	200	$	2,592,546.08	3.296063
3.17.9	200	150	200	$	2,734,856.80	3.220226
3.17.10	200	150	200	$	2,681,645.52	3.198278
3.18.1	200	600	50	$	2,421,911.76	15.103373
3.18.2	200	600	50	$	2,568,316.81	13.122953
3.18.3	200	600	50	$	2,476,326.32	13.197468
3.18.4	200	600	50	$	2,652,075.24	13.056958
3.18.5	200	600	50	$	2,480,639.07	13.067214
3.18.6	200	600	50	$	2,608,794.41	13.056877
3.18.7	200	600	50	$	2,450,895.33	13.037468
3.18.8	200	600	50	$	2,523,542.63	13.973391
3.18.9	200	600	50	$	2,291,338.39	13.109920
3.18.10	200	600	50	$	2,630,471.30	13.226190
3.19.1	400	600	200	$	2,573,288.62	28.113967
3.19.2	400	600	200	$	2,534,113.29	25.656367
3.19.3	400	600	200	$	2,383,586.90	25.648063
3.19.4	400	600	200	$	2,457,460.92	26.082668
3.19.5	400	600	200	$	2,456,477.75	26.244642
3.19.6	400	600	200	$	2,383,256.43	25.782938
3.19.7	400	600	200	$	2,498,216.35	25.979803
3.19.8	400	600	200	$	2,457,963.11	25.928905
3.19.9	400	600	200	$	2,441,306.40	25.912183
3.19.10	400	600	200	$	2,459,960.98	26.006798
3.20.1	400	600	50	$	2,402,626.55	27.828020
3.20.2	400	600	50	$	2,378,852.49	26.027953
3.20.3	400	600	50	$	2,426,135.66	26.011958
3.20.4	400	600	50	$	2,506,914.04	26.050097
3.20.5	400	600	50	$	2,371,448.72	26.148700
3.20.6	400	600	50	$	2,387,831.15	26.187899
3.20.7	400	600	50	$	2,480,280.17	26.168214
3.20.8	400	600	50	$	2,456,152.16	26.230785
3.20.9	400	600	50	$	2,307,104.35	26.064329

3.20.10	400	600	50	$	2,397,993.11	26.451970
3.21.1	400	150	200	$	2,525,802.03	9.039482
3.21.2	400	150	200	$	2,629,470.60	6.006670
3.21.3	400	150	200	$	2,592,912.11	6.681447
3.21.4	400	150	200	$	2,602,752.10	6.396171
3.21.5	400	150	200	$	2,681,695.74	6.467551
3.21.6	400	150	200	$	2,681,492.56	6.528052
3.21.7	400	150	200	$	2,671,478.93	6.473262
3.21.8	400	150	200	$	2,597,993.00	6.486534
3.21.9	400	150	200	$	2,633,149.28	6.574112
3.21.10	400	150	200	$	2,425,275.45	6.522104
3.22.1	100	600	200	$	2,361,846.37	9.125489
3.22.2	100	600	200	$	2,559,156.72	6.215641
3.22.3	100	600	200	$	2,530,189.34	6.528771
3.22.4	100	600	200	$	2,502,524.02	6.656929
3.22.5	100	600	200	$	2,554,020.27	6.617924
3.22.6	100	600	200	$	2,437,275.31	6.568041
3.22.7	100	600	200	$	2,564,987.56	6.696728
3.22.8	100	600	200	$	2,665,846.15	6.792324
3.22.9	100	600	200	$	2,467,454.64	6.623905
3.22.10	100	600	200	$	2,498,409.61	6.666246
3.23.1	400	600	100	$	2,497,731.69	28.787656
3.23.2	400	600	100	$	2,624,613.42	26.752624
3.23.3	400	600	100	$	2,518,654.44	26.803709
3.23.4	400	600	100	$	2,549,754.71	26.721809
3.23.5	400	600	100	$	2,248,329.83	26.806524
3.23.6	400	600	100	$	2,439,887.09	27.023788
3.23.7	400	600	100	$	2,500,693.31	26.740969
3.23.8	400	600	100	$	2,485,715.24	26.794943
3.23.9	400	600	100	$	2,208,553.80	26.763962
3.23.10	400	600	100	$	2,452,374.24	26.611413
3.24.1	400	300	200	$	2,311,227.04	14.986168
3.24.2	400	300	200	$	2,571,873.54	12.908228
3.24.3	400	300	200	$	2,510,382.14	12.956515
3.24.4	400	300	200	$	2,620,125.67	13.079292
3.24.5	400	300	200	$	2,353,557.28	13.140552
3.24.6	400	300	200	$	2,578,068.24	13.200511
3.24.7	400	300	200	$	2,636,858.48	13.075453
3.24.8	400	300	200	$	2,541,697.59	13.188831
3.24.9	400	300	200	$	2,534,808.53	13.176685
3.24.10	400	300	200	$	2,649,114.93	13.570455
3.25.1	200	600	200	$	2,586,203.00	15.487477
3.25.2	200	600	200	$	2,542,025.94	13.416498

3.25.3	200	600	200	$	2,532,678.52	13.377644
3.25.4	200	600	200	$	2,551,324.20	13.457172
3.25.5	200	600	200	$	2,440,895.17	13.408860
3.25.6	200	600	200	$	2,486,853.52	13.694931
3.25.7	200	600	200	$	2,499,751.98	13.742482
3.25.8	200	600	200	$	2,555,402.13	13.604627
3.25.9	200	600	200	$	2,476,261.54	13.457328
3.25.10	200	600	200	$	2,393,743.81	13.357771
3.26.1	400	150	100	$	2,650,593.76	9.110517
3.26.2	400	150	100	$	2,481,359.68	6.160920
3.26.3	400	150	100	$	2,650,269.50	6.609956
3.26.4	400	150	100	$	2,454,511.02	6.574247
3.26.5	400	150	100	$	2,563,116.57	6.624607
3.26.6	400	150	100	$	2,585,760.57	6.700196
3.26.7	400	150	100	$	2,604,732.79	6.675251
3.26.8	400	150	100	$	2,691,456.70	6.673651
3.26.9	400	150	100	$	2,633,488.18	6.589311
3.26.10	400	150	100	$	2,436,400.20	6.639394
3.27.1	400	300	50	$	2,422,117.42	14.916207
3.27.2	400	300	50	$	2,481,003.25	12.866817
3.27.3	400	300	50	$	2,482,540.47	12.930660
3.27.4	400	300	50	$	2,456,033.54	12.916416
3.27.5	400	300	50	$	2,437,548.01	12.944416
3.27.6	400	300	50	$	2,696,919.73	12.919948
3.27.7	400	300	50	$	2,410,111.34	13.082640
3.27.8	400	300	50	$	2,619,625.40	12.957825
3.27.9	400	300	50	$	2,196,312.74	12.938493
3.27.10	400	300	50	$	2,647,211.78	12.842269

B. Tabu Search Python Code

This section displays the Tabu Search Python code used in this paper's solution methodologies.

B.1 Edge.py

```
from typing import List

# Constants for Edge data csv
COLS = 8
```

```
TYPE = 0
NAME = 3
EDGE_COST_PER_UNIT = 4
PROBABILITY = 5
RELIABILITY = 6
EXCHANGE_RATE = 7

# Might have to save edge data at node level
SUPPLIER_TO_PLANT = 0
PLANT_TO_WAREHOUSE = 1
WAREHOUSE_TO_MARKET = 2


class Edge:
    def __init__(self, row_data: List):
        if len(row_data) < COLS:
            raise EdgeDataError
        self.name = row_data[NAME]
        self.edge_cpu = row_data[EDGE_COST_PER_UNIT]
        self.probability = row_data[PROBABILITY]
        self.reliability = row_data[RELIABILITY]
        self.exchange_rate = row_data[EXCHANGE_RATE]  # exchange rate
of A in AB
        self.edge_coefficient = (
            self.edge_cpu * self.probability * self.reliability *
self.exchange_rate
        )

    def __str__(self):
        return f"Edge: {self.name}"

    def __repr__(self):
        return f"Edge(name:{self.name}, edge_cpu:{self.edge_cpu},
probability:{self.probability}, reliability:{self.reliability},
exchange_rate:{self.exchange_rate})"

    def __eq__(self, other):
        return (
            isinstance(other, Edge)
            and other.name == self.name
            and other.edge_cpu == self.edge_cpu
            and other.probability == self.probability
            and other.reliability == self.reliability
            and other.exchange_rate == self.exchange_rate
        )

    @classmethod
    def create_edge(cls, row_data: List):
        """List -> Edge
        creates the appropriate edge"""
        type_ = row_data[TYPE]

        if type_ == SUPPLIER_TO_PLANT:
            return SupplierToPlantEdge(row_data)
        elif type_ == PLANT_TO_WAREHOUSE:
            return PlantToWareEdge(row_data)
        elif type_ == WAREHOUSE_TO_MARKET:
```

```
            return WareToMarkEdge(row_data)
        else:
            raise InvalidEdgeError


class WareToMarkEdge(Edge):
    def __init__(self, row_data: List):
        super().__init__(row_data)
        # warehouse to market transportation cost
        # warehouse to market risk cost
        # warehouse to market total cost
        self.wh_to_mk_trans_cost = self.edge_cpu * self.probability *
self.exchange_rate
        self.wh_to_mk_risk_cost = (
            self.edge_cpu * (1 - self.probability) * self.exchange_rate
        )
        self.wh_to_mk_total_cost = self.wh_to_mk_trans_cost +
self.wh_to_mk_risk_cost

    def __str__(self):
        return f"WareToMark: {self.name} {self.wh_to_mk_total_cost}\n"

    def __repr__(self):
        return f"WareToMarkEdge(wh_to_mk_trans
cost:{self.wh_to_mk_trans_cost},
wh_to_mk_risk_cost:{self.wh_to_mk_risk_cost},
wh_to_mk_total_cost{self.wh_to_mk_total_cost} {super().__repr__()})"

    def __eq__(self, other):
        return (
            isinstance(other, WareToMarkEdge)
            and other.wh_to_mk_trans_cost == self.wh_to_mk_trans_cost
            and other.wh_to_mk_risk_cost == self.wh_to_mk_risk_cost
            and other.wh_to_mk_total_cost == self.wh_to_mk_total_cost
            and super(WareToMarkEdge, self).__eq__(other)
        )


class PlantToWareEdge(Edge):
    def __init__(self, row_data: List):
        super().__init__(row_data)
        self.pl_to_wh_trans_cost = self.edge_cpu * self.probability *
self.exchange_rate
        self.pl_to_wh_risk_cost = (
            self.edge_cpu * (1 - self.probability) * self.exchange_rate
        )
        self.pl_to_wh_total_cost = self.pl_to_wh_trans_cost +
self.pl_to_wh_risk_cost

    def __str__(self):
        return f"PlantToWare: {self.name}"

    def __repr__(self):
        return f"PlantToWareEdge(wh_to_mk_trans cost:
{self.pl_to_wh_trans_cost}, pl_to_wh_risk_cost:
{self.pl_to_wh_risk_cost}, pl_to_wh_total_cost:
{self.pl_to_wh_total_cost}\n\t{super().__repr__()})\n"
```

```
    def __eq__(self, other):
        return (
            isinstance(other, PlantToWareEdge)
            and other.pl_to_wh_trans_cost == self.pl_to_wh_trans_cost
            and other.pl_to_wh_risk_cost == self.pl_to_wh_risk_cost
            and other.pl_to_wh_total_cost == self.pl_to_wh_total_cost
            and super(PlantToWareEdge, self).__eq__(other)
        )


class SupplierToPlantEdge(Edge):
    def __init__(self, row_data: List):
        super().__init__(row_data)
        self.supplier_raw_material_cost = (
            self.edge_cpu * self.probability * self.exchange_rate
        )
        self.supplier_quality_risk_cost = (
            self.probability * self.reliability * self.exchange_rate
        )
        self.supplier_failure_cost = (
            self.edge_cpu * (1 - self.probability) * self.exchange_rate
        )
        self.supplier_total_cost = None
        # TODO: change later to NOne

    def __str__(self):
        return f"SupplierToPlantEdge: {self.name}"

    def __repr__(self):
        return f"SupplierToPlant(supplier_raw_material_cost:
{self.supplier_raw_material_cost}, supplier_quality_risk_cost:
{self.supplier_quality_risk_cost}, supplier_failure_cost:
{self.supplier_failure_cost} supplier_total_cost:
{self.supplier_total_cost}\n\t{super().__repr__()})\n"

    def __eq__(self, other):
        return (
            isinstance(other, SupplierToPlantEdge)
            and other.supplier_raw_material_cost ==
self.supplier_raw_material_cost
            and other.supplier_quality_risk_cost ==
self.supplier_quality_risk_cost
            and other.supplier_failure_cost ==
self.supplier_failure_cost
            and other.supplier_total_cost == self.supplier_total_cost
            and super(SupplierToPlantEdge, self).__eq__(other)
        )


class EdgeDataError(Exception):
    def __init__(self):
        self.message = f"Invalid arguements passed to Edge during
initialization"
        super().__init__(self.message)
```

```
class InvalidEdgeError(Exception):
    def __init__(self):
        self.message = f"Edge is not of valid type"
        super().__init__(self.message)
```

B.2 Node.py

```
import math
import random as rd
from datetime import datetime
from typing import Dict, List

from scipy.stats import norm

# Constants for Node data csv
MAX_COLS = 8

# Columns in each row -> see NodeDataCSV.csv
TYPE = 0
NAME = 1
EXCHANGE_RATE = 2
MAX_CAPACITY = 3
MEAN_DEMAND = 4
VARIANCE_OF_DEMAND = 5
GOODWILL_LOSS_COST_PER_UNIT = 6
EXCESS_INV_SUPPLY_COST_PER_UNIT = 7
PRODUCTION_COST_PER_UNIT = 8

OUTSOURCED = 1
DOMESTIC = 0
PLANT = 2
WAREHOUSE = 3
MARKET = 4


class Node:
    def __init__(self, row_data: List):
        if len(row_data) < MAX_COLS:
            raise NodeDataError
        self.exchange_rate = row_data[EXCHANGE_RATE]
        self.max_capacity = row_data[MAX_CAPACITY]
        self.mean_demand = row_data[MEAN_DEMAND]
        self.var_demand = row_data[VARIANCE_OF_DEMAND]
        self.g_cpu = row_data[GOODWILL_LOSS_COST_PER_UNIT]
        self.e_cpu = row_data[EXCESS_INV_SUPPLY_COST_PER_UNIT]
        self.prod_cost = row_data[PRODUCTION_COST_PER_UNIT]

    def __str__(self):
        return f"Node:{self.exchange_rate}, {self.max_capacity},
{self.mean_demand}, {self.var_demand}, {self.g_cpu}, {self.e_cpu},
{self.prod_cost}"

    def __repr__(self):
        return f"Node(exchange_rate:{self.exchange_rate},
max_capacity:{self.max_capacity}, mean_demand:{self.mean_demand},
```

```
var_demand:{self.var_demand}, g_cpu:{self.g_cpu}, e_cpu:{self.e_cpu},
{self.prod_cost})"

    def __eq__(self, other):
        return (
            isinstance(other, Node)
            and other.exchange_rate == self.exchange_rate
            and other.max_capacity == self.max_capacity
            and other.mean_demand == self.mean_demand
            and other.var_demand == self.var_demand
            and other.g_cpu == self.g_cpu
            and other.e_cpu == self.e_cpu
            and other.prod_cost == self.prod_cost
        )

    @classmethod
    def create_node(cls, row: List):
        """Returns appropriate Node based on the row data from Node CSV
data """
        type_: int = row[0]
        if type_ == DOMESTIC or type_ == OUTSOURCED:  # domestic
supplier
            return Supplier(row)
        elif type_ == PLANT:
            return Plant(row)
        elif type_ == WAREHOUSE:
            return Warehouse(row)
        elif type_ == MARKET:
            return Market(row)
        else:
            raise InvalidNodeError


class Supplier(Node):
    domestic_count = 0
    outsourced_count = 0

    def __init__(self, row_data: List):
        super().__init__(row_data)
        if row_data[TYPE] == DOMESTIC:
            Supplier.domestic_count += 1
            self.name = f"SD{Supplier.domestic_count}"
            self.is_domestic = True
        elif row_data[TYPE] == OUTSOURCED:
            Supplier.outsourced_count += 1
            self.name = f"SO{Supplier.outsourced_count}"
            self.is_domestic = False

    def __str__(self):
        return f"Supplier:{self.name}"

    def __repr__(self):
        return f"Supplier(name:{self.name},
is_domestic:{self.is_domestic},\n\t{super().__repr__()})\n"

    def __eq__(self, other):
        return (
```

```
            isinstance(other, Supplier)
            and other.name == self.name
            and other.is_domestic == self.is_domestic
            and super(Supplier, self).__eq__(other)
        )


class Plant(Node):
    plant_count = 0

    def __init__(self, row_data: List):
        super().__init__(row_data)
        Plant.plant_count += 1
        self.name: str = f"P{Plant.plant_count}"
        self.plant_mfg_cost = self.prod_cost
        self.warehouse_paths: Dict[str, int] = dict()
        self.supplier_paths: Dict[str, int] = dict()
        self.plant_risk_cost = None
        self.plant_total_cost = None

    def __str__(self):
        return f"Plant:{self.name}"

    def __repr__(self):
        return f"Plant(name:{self.name},
manufacturing_cost:{self.plant_mfg_cost},
risk_cost:{self.plant_risk_cost} {super().__repr__()})\n"

    def __eq__(self, other):
        return (
            isinstance(other, Plant)
            and other.name == self.name
            and other.plant_mfg_cost == self.plant_mfg_cost
            and other.warehouse_paths == self.warehouse_paths
            and other.supplier_paths == self.supplier_paths
            and other.plant_risk_cost == self.plant_risk_cost
            and other.plant_total_cost == self.plant_total_cost
            and super(Plant, self).__eq__(other)
        )

    def init_plant_node(self, suppliers: List, warehouses: List,
edge_data: Dict):
        """List, Dict -> None
        initializes the plant node dictionary, plant risk cost and
plant total cost
        """
        self._init_plant_to_warehouse_dict(warehouses, edge_data)
        self._init_supplier_to_plant_dict(suppliers, edge_data)
        self._init_plant_risk_cost(edge_data)
        # self._init_plant_total_cost()

    # TODO: make this more efficient and ensure its right, do something
like this

    def _init_plant_to_warehouse_dict(self, warehouses: List,
edge_data):
        myDict = dict()
```

```
        for node in warehouses:
            edge_name = f"{self.name}{node.name}"
            try:
                temp_edge_data = edge_data[edge_name]
            except:
                temp_edge_data = None
            myDict.update({edge_name:
temp_edge_data.pl_to_wh_total_cost})
        self.warehouse_paths = myDict

    def _init_supplier_to_plant_dict(self, suppliers: List, edge_data):
        myDict = dict()
        for node in suppliers:
            edge_name = f"{node.name}{self.name}"
            # print(edge_name)
            try:
                temp_edge_data = edge_data[edge_name]
            except:
                print(f"Failed to find: {edge_name}")
                temp_edge_data = None
            myDict.update({edge_name:
temp_edge_data.supplier_total_cost})
        self.supplier_paths = myDict

    def _init_plant_risk_cost(self, edge_data):
        counter = 0
        for path in self.warehouse_paths.keys():
            edge = edge_data[path]
            probability = edge.probability
            reliability = edge.reliability
            exchange_rate = edge.exchange_rate
            counter += probability * reliability
        self.plant_risk_cost = counter

    def _init_plant_total_cost(self):
        self.plant_total_cost = self.plant_mfg_cost /
self.plant_risk_cost


class Warehouse(Node):
    warehouse_count = 0

    def __init__(self, row_data: List):
        super().__init__(row_data)
        Warehouse.warehouse_count += 1
        self.name: str = f"W{Warehouse.warehouse_count}"
        self.market_paths: Dict[str, int] = dict()
        self.plant_paths: Dict[str, int] = dict()

    def __str__(self):
        return f"Warehouse:{self.name}\n"

    def __repr__(self):
        return f"Warehouse(name:{self.name}, {super().__repr__()}\n"

    def __eq__(self, other):
        return (
```

```
            isinstance(other, Warehouse)
            and other.name == self.name
            and super(Warehouse, self).__eq__(other)
        )

    def init_warehouse_node(self, plant_nodes, market_nodes,
edge_data):
        self._init_plant_to_warehouse_dict(plant_nodes, edge_data)
        self._init_warehouse_to_market_dict(market_nodes, edge_data)

    def _init_plant_to_warehouse_dict(self, plant_nodes: List,
edge_data):
        myDict = dict()
        for node in plant_nodes:
            edge_name = f"{node.name}{self.name}"
            try:
                temp_edge_data = edge_data[edge_name]
            except:
                temp_edge_data = None
            myDict.update({edge_name:
temp_edge_data.pl_to_wh_total_cost})
        self.plant_paths = myDict

    def _init_warehouse_to_market_dict(self, market_nodes: List,
edge_data):
        myDict = dict()
        for node in market_nodes:
            edge_name = f"{self.name}{node.name}"
            # print(edge_name)
            try:
                temp_edge_data = edge_data[edge_name]
            except:
                temp_edge_data = None
            myDict.update({edge_name:
temp_edge_data.wh_to_mk_total_cost})
        self.supplier_paths = myDict


class Market(Node):
    market_count = 0

    def __init__(self, row_data: List):
        super().__init__(row_data)
        Market.market_count += 1
        self.name: str = f"M{Market.market_count}"
        self.six_sigma = 6 * math.sqrt(self.var_demand)
        self.market_min = self.mean_demand - self.six_sigma  # market
demand range
        self.market_max = self.mean_demand + self.six_sigma
        self.market_pdf = 1 - norm.pdf(
            self.mean_demand, self.mean_demand,
math.sqrt(self.var_demand)
        )  # not sure
        self.market_demand = rd.randint(int(self.market_min),
int(self.market_max))
        self.surplus_supply_cost = (
```

```
            (self.market_demand - self.mean_demand) * self.market_pdf *
self.e_cpu
        )
        self.shortage_supply_cost = (
            (self.mean_demand - self.market_demand) * self.market_pdf *
self.g_cpu
        )
        self.is_surplus = True if (self.surplus_supply_cost > 0) else
False
        self.warehouse_paths: Dict[str, int] = dict()  # later will be
a dict
        # acts as a placeholder, need functions to check that sum of
values matches market_demand
        # TODO: this is dependent on markets being last in the Node
data csv
        # NOTE: IDK why we made this comment

    def __str__(self):
        return f"Market:{self.name} {self.surplus_supply_cost}
{self.shortage_supply_cost}\n"

    def __repr__(self):
        return f"Market(name:{self.name}, six-sigma:{self.six_sigma},
market-min:{self.market_min}, market-max:{self.market_max},
pdf:{self.market_pdf}, market demand:{self.market_demand}, surplus-
cost:{self.surplus_supply_cost}, shortage-
cost:{self.shortage_supply_cost}, is_surplus:
{self.is_surplus}\n\t{super().__repr__()})\n"

    def __eq__(self, other):
        return (
            isinstance(other, Market)
            and other.name == self.name
            and other.six_sigma == self.six_sigma
            and other.market_min == self.market_min
            and other.market_max == self.market_max
            and other.market_pdf == self.market_pdf
            and super(Market, self).__eq__(other)
        )

    def init_market_node(self, warehouse: List[Warehouse], edge_data:
Dict):
        self._init_market_available_paths(warehouse, edge_data)

    def _init_market_available_paths(self, warehouses: List[Warehouse],
edge_data):
        """initalizes the dictionary for the edges available to the
market node at the time."""
        # returns a dictionary
        myDict = dict()
        for node in warehouses:
            edge_name = f"{node.name}{self.name}"
            try:
                temp_edge_data = edge_data[edge_name]
            except:
                temp_edge_data = None
```

```
            myDict.update({edge_name:
temp_edge_data.wh_to_mk_total_cost})
        self.warehouse_paths = myDict


class NodeDataError(Exception):
    def __init__(self):
        self.message = f"Invalid arguements passed to Node during
creation"
        super().__init__(self.message)


class InvalidNodeError(Exception):
    def __init__(self):
        self.message = f"Node is not of type [Supplier, Plant,
Warehouse, Market]"
        super().__init__(self.message)
```

B.3 Graph.py

```
from typing import List, NewType, Optional, TypeVar, Union

import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
from networkx.drawing.nx_agraph import graphviz_layout

# from supply_network.edge import *
# from supply_network.node import *
from .edge import *
from .node import *

"""
Notes:
 __attribute -> private
_attribute -> protected
"""

SOURCE = "source"
SINK = "sink"


class SupplyGraph:
    # could probably do a json of all this
    supplier_nodes: List = []
    plant_nodes: List = []
    warehouse_nodes: List = []
    market_nodes: List = []

    supplier_to_plant_edges: List = []
    plant_to_warehouse_edges: List = []
    warehouse_to_market_edges: List = []

    paths: List = []
```

```
    edges: dict = {}

    G: nx.DiGraph = nx.DiGraph()

    def __init__(self, nodeDataFile: str, edgeDataFile: str):
        self.node_data_file = nodeDataFile
        self.edge_data_file = edgeDataFile

    def build_graph(self):
        """ builds graph and saves into G attribute """
        self._add_nodes_to_graph()
        self._build_edge_data_dict()  # TODO: make this special

        self._init_market_nodes(self.warehouse_nodes, self.edges)
        self._init_warehouse_nodes(self.plant_nodes, self.market_nodes,
self.edges)
        self._init_plant_nodes(self.supplier_nodes,
self.warehouse_nodes, self.edges)
        self.total_market_demand = self._calc_total_market_demand()
        # print(self.total_market_demand)

    def _add_nodes_to_graph(self):
        """ adds nodes from CSV data into the graph G """
        node_df = pd.read_csv(self.node_data_file)
        for row in node_df.values:
            node = Node.create_node(row)
            self.G.add_node(node.name, data=node, name=node.name)
            self._store_node(node)
        self._connect_source_to_suppliers()
        self._connect_suppliers_to_plants()
        self._connect_plants_to_warehouses()
        self._connect_warehouse_to_markets()
        self._connect_markets_to_sink()
        self.paths = list(nx.all_simple_paths(self.G, source=SOURCE,
target=SINK))
        # NOTE: consider leaving this as a generator object for later
tabu search

    def _build_edge_data_dict(self):
        """ reads in edge data CSV and builds a dictionary of their
names to the edge object with the data """
        edge_df = pd.read_csv(self.edge_data_file)
        for row in edge_df.values:
            edge = Edge.create_edge(row)
            self.edges.update({edge.name: edge})
            self._store_edge(edge)

    def _store_node(self, node):
        """ adds nodes to appropriate node list in graph """
        if type(node) == Supplier:
            self.supplier_nodes.append(node)
        elif type(node) == Plant:
            self.plant_nodes.append(node)
        elif type(node) == Warehouse:
            self.warehouse_nodes.append(node)
        elif type(node) == Market:
            self.market_nodes.append(node)
```

```
            # node._init_market_available_paths(self.warehouse_nodes,
self.edges)
        else:
            raise InvalidNodeError

    def _store_edge(self, edge):
        """ stores edges in the appropriate list for the graph """
        if type(edge) == SupplierToPlantEdge:
            self.supplier_to_plant_edges.append(edge)
        elif type(edge) == PlantToWareEdge:
            self.plant_to_warehouse_edges.append(edge)
        elif type(edge) == WareToMarkEdge:
            self.warehouse_to_market_edges.append(edge)
        else:
            raise InvalidEdgeError

    def get_node_by_name(self, nodeName: str):
        """
        nodeName : str -> Node
        returns the Node with the given name
        """
        first_char = nodeName[0]
        if nodeName[0] == "S":  # supplier
            for node in self.supplier_nodes:
                if nodeName == node.name:
                    return node
        if nodeName[0] == "P":  # plant
            for node in self.plant_nodes:
                if nodeName == node.name:
                    return node
        elif nodeName[0] == "W":  # warehouse
            for node in self.warehouse_nodes:
                if nodeName == node.name:
                    return node
        elif nodeName[0] == "M":  # market
            for node in self.market_nodes:
                if nodeName == node.name:
                    return node
        return None

    def get_edge_by_name(self, name):
        try:
            target = self.edges[name]
            return target
        except:
            return None

    def _get_node_constraint_data(self):
        """ returns a dictionary of the node names and their
constraints """
        constraint_dict = dict()
        for plant in self.plant_nodes:
            constraint_dict.update({plant.name: plant.max_capacity})

        for warehouse in self.warehouse_nodes:
            constraint_dict.update({warehouse.name:
warehouse.max_capacity})
```

```
        return constraint_dict

    def _get_market_demand_data(self):
        """ returns a dictionary of the node names and their demands
"""
        demand_dict = dict()
        for market in self.market_nodes:
            demand_dict.update({market.name: market.market_demand})
        return demand_dict

    def _calc_total_market_demand(self):
        """
        calculates the total market cost for the supply chain network
        Total market cost is broken down into excess cost and shortage
cost
        """
        total_market_demand = 0
        for market in self.market_nodes:
            total_market_demand += market.market_demand
        return total_market_demand

    def _connect_source_to_suppliers(self):
        for supplier in self.supplier_nodes:
            self.G.add_edge(SOURCE, supplier.name)

    def _connect_suppliers_to_plants(self):
        for supplier in self.supplier_nodes:
            for plant in self.plant_nodes:
                self.G.add_edge(supplier.name, plant.name)

    def _connect_plants_to_warehouses(self):
        for plant in self.plant_nodes:
            for warehouse in self.warehouse_nodes:
                self.G.add_edge(plant.name, warehouse.name)

    def _connect_warehouse_to_markets(self):
        for warehouse in self.warehouse_nodes:
            for market in self.market_nodes:
                self.G.add_edge(warehouse.name, market.name)

    def _connect_markets_to_sink(self):
        for market in self.market_nodes:
            self.G.add_edge(market.name, SINK)

    def show_graph(self):
        self.G.remove_node(SOURCE)
        self.G.remove_node(SINK)
        graph_pos = graphviz_layout(self.G, prog="dot", args="-
Grankdir=LR")
        nx.draw(self.G, with_labels=True, pos=graph_pos)
        plt.show()
        self.G.add_node(SOURCE)
        self.G.add_node(SINK)
        self._connect_source_to_suppliers()
        self._connect_markets_to_sink()

    def show_full_graph(self):
```

```
        graph_pos = graphviz_layout(self.G, prog="dot", args="-
Grankdir=LR")
        nx.draw(self.G, with_labels=True, pos=graph_pos)
        plt.show()

    def _print_all_paths(self):
        print(self.paths)

    def _get_edge(self, node1, node2):
        """ accesses the edge dictionary based on the corresponding
edge, returns an edge """
        target_edge = node1.name + node2.name
        return self.edges[target_edge]

    def _init_market_nodes(self, warehouse_nodes, edge_data):
        for market_node in self.market_nodes:
            market_node.init_market_node(warehouse_nodes, edge_data)

    def _init_warehouse_nodes(self, plant_nodes, market_nodes,
edge_data):
        for warehouse_node in self.warehouse_nodes:
            warehouse_node.init_warehouse_node(plant_nodes,
market_nodes, edge_data)

    def _init_plant_nodes(self, supplier_nodes, warehouse_nodes,
edge_data):
        for plant_node in self.plant_nodes:
            plant_node.init_plant_node(supplier_nodes, warehouse_nodes,
edge_data)
        # NOTE: O(N^2)
```

B.4 Solution.py

```
import random
from .graph import SupplyGraph


class Solution:
    def __init__(
        self, total_market_demand, market_demand_data,
node_constraint_data, sg
    ):
        self.total_market_demand = total_market_demand
        self.market_demand_data = market_demand_data
        self.constraints = node_constraint_data.copy()
        self.sg = sg
        self.edge_data = sg.edges
        # quantities at each edge
        # coefficient
        self.wares_to_markets_quantities = {}  # edge_name -> quant
        self.warehouse_demands = {}
        self.plants_to_wares_quantities = {}
        self.plant_demands = {}
        self.suppliers_to_plants_quantities = {}
```

```
        self.total_supply_cost = None
        self.total_plant_production_cost = None
        self.total_plant_warehouse_cost = None
        self.total_warehouse_market_cost = None
        self.total_market_cost = None
        self.solution = None

    def __str__(self):
        return f"\n \
                Solution:\n \
                Supplier->Plant:
{self.suppliers_to_plants_quantities}\n \
                Plant->Warehouse: {self.plants_to_wares_quantities}\n \
                Warehouse->Market: {self.wares_to_markets_quantities}\n
\
                Market Demand: {self.market_demand_data}\n\n \
                Total Supply Cost: {self.total_supply_cost}\n \
                Total Plant Production Cost:
{self.total_plant_production_cost}\n \
                Total Plant Warehouse Cost:
{self.total_plant_warehouse_cost}\n \
                Total Warehouse Market Cost:
{self.total_warehouse_market_cost}\n \
                Total Market Cost: {self.total_market_cost}\n \
                Objective Function: {self.solution}\n\n \
                Total Market Demand: {self.total_market_demand}\n"

    # TODO: test this
    def __gt__(self, other):
        return self.solution > other.solution

    def show_demands(self):
        print(
            f"\
        Demands:\n\n\
        Plant Demands:     {self.plant_demands}\n\
        Warehouse Demands: {self.warehouse_demands}\n\
        Market Demands:    {self.market_demand_data}\n\
        "
        )

    def calc_objective_function(self, market_nodes, plant_nodes,
edge_data):
        """ returns our objective function """
        self._calc_total_supply_cost(edge_data)
        self._calc_total_plant_production_cost(plant_nodes)
        self._calc_total_plant_warehouse_cost(edge_data)
        self._calc_total_warehouse_market_cost(edge_data)
        self._calc_total_market_cost(market_nodes)

        self.solution = (
            self.total_supply_cost
            + self.total_plant_production_cost
            + self.total_plant_warehouse_cost
            + self.total_warehouse_market_cost
            + self.total_market_cost
        )
```

```
        return self

    # TODO: break down, write tests
    def calc_ware_to_market_quants(self, market_nodes,
warehouse_nodes):
        """ setting the warehouse to market quantity amounts """
        random.shuffle(market_nodes)
        for market in market_nodes:
            market_demand = market.market_demand
            path = []
            while market_demand != 0:
                entries = list(market.warehouse_paths.items())
                path_choice = random.choice(entries)[0]
                if path_choice in path:
                    continue  # try again
                ware_node = path_choice.replace(market.name, "")  # W1
                available_quant = self.constraints[ware_node]
                if market_demand > available_quant:
                    self.wares_to_markets_quantities.update(
                        {path_choice: available_quant}
                    )
                    market_demand -= available_quant
                    self.constraints[ware_node] = 0
                else:
                    self.wares_to_markets_quantities.update(
                        {path_choice: market_demand}
                    )
                    self.constraints[ware_node] -= market_demand
                    market_demand = 0
                path.append(path_choice)

    def set_warehouse_demands(self, market_nodes, warehouse_nodes):
        wh_demand = dict()
        for node in warehouse_nodes:
            wh_demand.update({node.name: 0})

        for key in self.wares_to_markets_quantities.keys():
            target_warehouse = key[: key.find("M")]
            wh_demand[target_warehouse] +=
self.wares_to_markets_quantities[key]
        self.warehouse_demands = wh_demand

    # TODO: break down, write tests, abstract them too
    def calc_plant_to_ware_quants(self, plant_nodes, warehouse_nodes):
        """ setting the warehouse to market quantity amounts """
        random.shuffle(warehouse_nodes)
        for warehouse in warehouse_nodes:
            warehouse_demand = self.warehouse_demands[warehouse.name]
            path = []
            while warehouse_demand != 0:
                entries = list(warehouse.plant_paths.items())
                # entries = list(pal.warehouse_paths.items())
                path_choice = random.choice(entries)[0]
                if path_choice in path:
                    continue  # try again
                plant_node = path_choice.replace(warehouse.name, "")  #
P1
```

```
                # ware_node = path_choice.replace(market.name, "")  #
W1
                available_quant = self.constraints[plant_node]
                if warehouse_demand > available_quant:
                    self.plants_to_wares_quantities.update(
                        {path_choice: available_quant}
                    )
                    warehouse_demand -= available_quant
                    self.constraints[plant_node] = 0
                else:
                    self.plants_to_wares_quantities.update(
                        {path_choice: warehouse_demand}
                    )
                    self.constraints[plant_node] -= warehouse_demand
                    warehouse_demand = 0
                path.append(path_choice)

    def set_plant_demands(self, warehouse_nodes, plant_nodes):
        pl_demand = dict()
        for node in plant_nodes:
            pl_demand.update({node.name: 0})

        for key in self.plants_to_wares_quantities.keys():
            target_warehouse = key[: key.find("W")]
            pl_demand[target_warehouse] +=
self.plants_to_wares_quantities[key]
        self.plant_demands = pl_demand

    # TODO: break down, write tests, abstract them too
    def calc_supplier_to_plant_quants(self, supplier_nodes,
plant_nodes):
        """ setting the warehouse to market quantity amounts """
        random.shuffle(plant_nodes)
        for plant in plant_nodes:
            plant_demand = self.plant_demands[plant.name]
            entries = list(plant.supplier_paths.items())
            path_choice = random.choice(entries)[0]
            plant_node = path_choice.replace(plant.name, "")  # S1
            self.suppliers_to_plants_quantities.update({path_choice:
plant_demand})
            # plant_demand = 0

    def _calc_total_supply_cost(self, edge_data):
        self.total_supply_cost = 0
        for quant_edge in self.suppliers_to_plants_quantities.keys():
            quant_edge_data = self.sg.get_edge_by_name(quant_edge)  #
might break
            # SupplierToPlantEdge
            raw_material_coefficient =
quant_edge_data.supplier_raw_material_cost
            failure_coefficient = quant_edge_data.supplier_failure_cost
            quality_risk_cost =
quant_edge_data.supplier_quality_risk_cost
            quantity = self.suppliers_to_plants_quantities[quant_edge]
            self.total_supply_cost += (
                raw_material_coefficient * quantity / quality_risk_cost
            ) + (failure_coefficient * quantity)
```

```
    def _calc_total_plant_production_cost(self, plant_nodes):
        self.total_plant_production_cost = 0
        for node in self.plant_demands.keys():
            node_data = self.sg.get_node_by_name(node)
            # PlantNode
            coefficient = node_data.plant_mfg_cost
            quantity = self.plant_demands[node]
            self.total_plant_production_cost += (
                coefficient * quantity
            ) / node_data.plant_risk_cost

    def _calc_total_plant_warehouse_cost(self, edge_data):
        self.total_plant_warehouse_cost = 0
        for quant_edge in self.plants_to_wares_quantities.keys():
            quant_edge_data = self.sg.get_edge_by_name(quant_edge)  #
might break
            # PlantToWareEdge
            coefficient = quant_edge_data.pl_to_wh_total_cost
            quantity = self.plants_to_wares_quantities[quant_edge]
            self.total_plant_warehouse_cost += coefficient * quantity

    def _calc_total_warehouse_market_cost(self, edge_data):
        self.total_warehouse_market_cost = 0
        for quant_edge in self.wares_to_markets_quantities.keys():
            quant_edge_data = self.sg.get_edge_by_name(quant_edge)  #
might break
            # WareToMarkEdge
            coefficient = quant_edge_data.wh_to_mk_total_cost
            quantity = self.wares_to_markets_quantities[quant_edge]
            self.total_warehouse_market_cost += coefficient * quantity

    def _calc_total_market_cost(self, market_nodes):
        self.total_market_cost = 0
        for node in self.market_demand_data.keys():
            node_data = self.sg.get_node_by_name(node)
            # MarketNode
            quantity = self.market_demand_data[node]
            if node_data.is_surplus:
                coefficient = node_data.surplus_supply_cost
            else:
                coefficient = abs(node_data.shortage_supply_cost)
            self.total_market_cost += coefficient
```

B.5 Tabu.py

```
import sys
import time

# from supply_network.graph import SupplyGraph
from .graph import SupplyGraph
from .solution import Solution


class Tabu:
```

```
def __init__(
    self,
    node_data: str,
    edge_data: str,
    iterations: int,
    num_neighbors: int,
    tabu_list_size: int,
):
    """        node_data.csv       edge_data.csv   iterations """
    self.iterations = int(iterations)
    self.num_neighbors = int(num_neighbors)
    self.tabu_list_size = int(tabu_list_size)
    self.graph = SupplyGraph(node_data, edge_data)
    self.graph.build_graph()
    self.total_market_demand = self.graph.total_market_demand
    self.market_demand_data = self.graph._get_market_demand_data()
    self.node_constraints = self.graph._get_node_constraint_data()
    self.initial_solution = None
    self.best_solution = None
    self.tabu_list = None
    self.time = None

def __str__(self):
    return f"\n\
    Tabu Search Results:\n\
    Stopping Criterion:  {self.iterations}\n\
    Candidate list size: {self.num_neighbors}\n\
    Tabu list size:      {self.tabu_list_size}\n\
    Tabu Search Time:    {self.time} seconds\n\
    Best Solution:\n{self.best_solution}\n"

def run(self):
    start_time = time.time()
    self.initial_solution = self._calc_initial_solution()
    self.best_solution = self.run_tabu_search()
    end_time = time.time()
    self.time = end_time - start_time
    print(self)
    # print(self.graph.supplier_to_plant_edges)

def _calc_initial_solution(self):
    """ calculates the initial solution for tabu search """
    return self.calc_solution()

def calc_solution(self):
    solution = Solution(
        self.total_market_demand,
        self.market_demand_data,
        self.node_constraints,
        self.graph,
    )
    solution.calc_ware_to_market_quants(
        self.graph.market_nodes, self.graph.warehouse_nodes
    )
    solution.set_warehouse_demands(
        self.graph.market_nodes, self.graph.warehouse_nodes
    )
```

```
        solution.calc_plant_to_ware_quants(
            self.graph.plant_nodes, self.graph.warehouse_nodes
        )
        solution.set_plant_demands(self.graph.warehouse_nodes,
self.graph.plant_nodes)
        solution.calc_supplier_to_plant_quants(
            self.graph.supplier_nodes, self.graph.plant_nodes
        )
        solution.calc_objective_function(
            self.graph.market_nodes, self.graph.plant_nodes,
self.graph.edges
        )
        return solution

    def _get_neighbor_solutions(self, source_solution):
        """ returns a list of neighboring solutions """
        neighbor_solutions = []
        for i in range(self.num_neighbors):
            solution = self.calc_solution()
            neighbor_solutions.append(solution)
        return neighbor_solutions

    def run_tabu_search(self):
        tabu_list = {}
        iter_num = 0
        curr_solution = self.initial_solution
        while iter_num <= self.iterations:  # stopping criterion
            neighbors = self._get_neighbor_solutions(curr_solution)
            best_neighbor = min(neighbors)
            while best_neighbor in tabu_list:
                neighbors.pop(best_neighbor)
                best_neighbor = min(neighbors)
            self._update_tabu_list(tabu_list, best_neighbor)
            curr_solution = min(curr_solution, best_neighbor)
            iter_num += 1
        self.tabu_list = tabu_list
        self.best_solution = curr_solution
        return curr_solution

    def _update_tabu_list(self, tabu_dict: dict, solution: Solution):
        """ updates the tabu tenure of the tabu list """
        for key in list(tabu_dict):
            tabu_dict[key] -= 1
            if tabu_dict[key] <= 0:
                tabu_dict.pop(key)
        tabu_dict.update({solution: self.tabu_list_size})  # adding
        return tabu_dict
```

B.6 Tabu Test.py

```
import sys
import random
import time
from supply_network.tabu import Tabu
from supply_network.graph import SupplyGraph
```

```
random.seed()
ARGS = 6


def main(argv):

    # NOTE: revisit where to best place timing, class resets,
    # depending on how to more fairly time each iteration, then do it
efficiently i.e build graph each time
    times = 1
    try:
        if "-a" in argv:
            times = int(argv[-1])
    except:
        times = 1
    total_time = 0
    total_sum = 0
    total_time_start = time.time()
    for _ in range(times):
        start_time = time.time()
        # node data, edge data,  iterations, neighbor list size, tabu
list size
        tabu = Tabu(sys.argv[1], sys.argv[2], sys.argv[3], sys.argv[4],
sys.argv[5])
        tabu.run()
        total_time += time.time() - start_time
        total_sum += tabu.best_solution.solution

    print(
        f"Total program runtime: {time.time() - total_time_start}
seconds, average time: {total_time/times}, average answer:
{total_sum/times}"
    )
    if "-v" in argv:
        print("Visualizing graph")
        tabu.best_solution.visualize_solution()


if __name__ == "__main__":
    if len(sys.argv) < ARGS:
        print(
            "Invalid arguements:\nUsage: main.py <csvNodeData>
<csvEdgeData> <number of iterations> <neighbor list size> <tabu list
size> [[-v] | [-a <number>]]"
        )
    elif len(sys.argv) == ARGS and "-v" in sys.argv:
        print(
            "Invalid arguements:\nUsage: main.py <csvNodeData>
<csvEdgeData> <number of iterations> <neighbor list size> <tabu list
size> [[-v] | [-a <number>]]"
        )
    elif (len(sys.argv) == ARGS + 1 and sys.argv[-1] != "-v") or (
        len(sys.argv) == ARGS + 2 and sys.argv[-2] != "-a" and
sys.argv[-1].isnumeric()
    ):
        print(
```

```
            "Invalid arguements:\nUsage: main.py <csvNodeData>
<csvEdgeData> <number of iterations> <neighbor list size> <tabu list
size> [[-v] | [-a <number>]]"
        )
    else:
        main(sys.argv)
```

C. Edge Data Generator Python Code

```
import argparse
import random as rd
import sys
from typing import List, Tuple

import pandas as pd


class EdgeDataGenerator:
    """
    A class used for creating EdgeDataCSV.csv files from nodde data
csvs.

    ...

    Attributes
    ----------
    file : str
            The filepath of the Node Data csv file

    c_range : List[float]
        The range of values to randomly populate the Edge Cost/Unit
column in the Edge Data csv
        (default is [1, 100])

    p_range : List[float]
        The range of values to randomly populate the Probability column
in the Edge Data csv
        (default is [0.0, 1.0])

    r_range : List[float]
        The range of values to randomly populate the Reliability column
in the Edge Data csv
        (default is [0.0, 1.0])

    e_range : List[float]
        The range of values to randomly populate the ER (exchange rate)
column in the Edge Data csv
        (default is [0.0, 1.0])

    supply_nodes : List[str]
        Contains the names of all of the supply nodes in the Node data
csv file

    plant_nodes : List[str]
        Contains the names of all of the plant nodes in the Node data
csv file
```

```
    warehouse_nodes : List[str]
        Contains the names of all of the warehouse nodes in the Node
data csv file

    market_nodes  : List[str]
        Contains the names of all of the market nodes in the Node data
csv file

    """

    def __init__(
        self,
        file: str,
        c_range: List[int],
        p_range: List[float],
        r_range: List[float],
        e_range: List[float],
    ):
        """
        Parameters
        ----------
        file : str
            The filepath of the Node Data csv file

        c_range : List[float]
            The range of values to randomly populate the Edge Cost/Unit
column in the Edge Data csv
            (default is [1, 100])

        p_range : List[float]
            The range of values to randomly populate the Probability
column in the Edge Data csv
            (default is [0.0, 1.0])

        r_range : List[float]
            The range of values to randomly populate the Reliability
column in the Edge Data csv
            (default is [0.0, 1.0])

        e_range : List[float]
            The range of values to randomly populate the ER (exchange
rate) column in the Edge Data csv
            (default is [0.0, 1.0])
        """
        self.file = file
        self.c_range = sorted(c_range)
        self.p_range = sorted(p_range)
        self.r_range = sorted(r_range)
        self.e_range = sorted(e_range)

        self.supply_nodes: List[str] = []
        self.plant_nodes: List[str] = []
        self.warehouse_nodes: List[str] = []
        self.market_nodes: List[str] = []

    def create_csv(self):
```

```
        """
        Runs the Edge Data Generator Object creating appropriate csv
file
        """
        self.read_file(self.file)
        file_name = self.create_file_name(self.file)
        column_names = [
            "Type",
            "Start Node",
            "End Node",
            "Edge Name",
            "Edge Cost/Unit",
            "Probability",
            "Reliability",
            "ER",
        ]
        data = self.create_rows(
            self.supply_nodes,
            self.plant_nodes,
            self.warehouse_nodes,
            self.market_nodes,
            self.c_range,
            self.p_range,
            self.r_range,
            self.e_range,
        )
        edges = pd.DataFrame(data, columns=column_names)
        edges.to_csv(file_name, index=False)

    def create_file_name(self, nodeDataFile: str) -> str:
        """
        Creates the corresponding file name of the nodeDataFile
        """
        return nodeDataFile.replace("Node", "Edge")

    def create_rows(
        self,
        suppliers,
        plants,
        warehouses,
        markets,
        c_range: List[int],
        p_range: List[float],
        r_range: List[float],
        e_range: List[float],
    ) -> List[List]:
        dataframe = []
        for s in suppliers:
            for p in plants:
                entry = self.create_row(
                    0,
                    s,
                    p,
                    rd.randint(c_range[0], c_range[1]),
                    round(rd.uniform(p_range[0], p_range[1]), 2),
                    round(rd.uniform(r_range[0], r_range[1]), 2),
                    round(rd.uniform(e_range[0], e_range[1]), 2),
```

```
                )
                dataframe.append(entry)
        for p in plants:
            for w in warehouses:
                entry = self.create_row(
                    1,
                    p,
                    w,
                    rd.randint(c_range[0], c_range[1]),
                    round(rd.uniform(p_range[0], p_range[1]), 2),
                    round(rd.uniform(r_range[0], r_range[1]), 2),
                    round(rd.uniform(e_range[0], e_range[1]), 2),
                )
                dataframe.append(entry)
        for w in warehouses:
            for m in markets:
                entry = self.create_row(
                    2,
                    w,
                    m,
                    rd.randint(c_range[0], c_range[1]),
                    round(rd.uniform(p_range[0], p_range[1]), 2),
                    round(rd.uniform(r_range[0], r_range[1]), 2),
                    round(rd.uniform(e_range[0], e_range[1]), 2),
                )
                dataframe.append(entry)
        return dataframe

    def create_row(
        self,
        edge_type: int,
        node1: str,
        node2: str,
        cpu: int,
        probability: float,
        reliability: float,
        exchange: float,
    ) -> List:
        return [
            edge_type,
            node1,
            node2,
            node1 + node2,
            cpu,
            probability,
            reliability,
            exchange,
        ]

    def read_file(self, filepath: str) -> None:
        node_df = pd.read_csv(filepath)
        node_lists = self.get_node_names(node_df)
        self.supply_nodes = node_lists[0]
        self.plant_nodes = node_lists[1]
        self.warehouse_nodes = node_lists[2]
        self.market_nodes = node_lists[3]
```

```
    def get_node_names(
        self,
        dataframe: pd.DataFrame,
    ) -> List[List[str]]:
        """
        Reads Node Data CSV file and places the name of each node in
the appropriate list
        """
        supply_nodes: List[str] = []
        plant_nodes: List[str] = []
        warehouse_nodes: List[str] = []
        market_nodes: List[str] = []
        curr_line = 2
        for row in dataframe.values:
            nodeType, nodeName = row[0], row[1]
            if nodeType == 0 or nodeType == 1:
                supply_nodes.append(nodeName)
            elif nodeType == 2:
                plant_nodes.append(nodeName)
            elif nodeType == 3:
                warehouse_nodes.append(nodeName)
            elif nodeType == 4:
                market_nodes.append(nodeName)
            else:
                print("ERROR")
                raise Exception("Unknown node type on line " +
str(curr_line))
            curr_line += 1
        return [supply_nodes, plant_nodes, warehouse_nodes,
market_nodes]


def main():
    args = parse_args(sys.argv)
    e_gen = EdgeDataGenerator(
        args.nodeDataFile,
        args.cost_range,
        args.probability,
        args.reliability,
        args.exchange_rate,
    )
    e_gen.create_csv()
    print(f"Done generating edge data from file: {e_gen.file}\n")
    exit()


def parse_args(args) -> argparse.Namespace:
    parser = argparse.ArgumentParser()
    parser.add_argument("nodeDataFile", help="Node data csv file to
generate Edge Data")
    parser.add_argument(
        "-c",
        "--cost_range",
        nargs=2,
        default=[10, 60],
        required=False,
        help="Edge cost per unit range to be used",
```

```
        type=int,
    )
    parser.add_argument(
        "-p",
        "--probability",
        nargs=2,
        default=[0.60, 0.95],
        required=False,
        help="Probability range to be used",
        type=float,
    )
    parser.add_argument(
        "-r",
        "--reliability",
        nargs=2,
        default=[0.80, 1.0],
        required=False,
        help="Reliability range to be used",
        type=float,
    )
    parser.add_argument(
        "-e",
        "--exchange_rate",
        nargs=2,
        default=[0.1, 2.5],
        required=False,
        help="Exchange rate range to be used",
        type=float,
    )
    args = parser.parse_args()
    return args


if __name__ == "__main__":
    main()
```