DATA-DRIVEN SUPPLY CHAIN MONITORING AND OPTIMIZATION

DATA-DRIVEN SUPPLY CHAIN MONITORING AND OPTIMIZATION

by

JING WANG, M.SC., B.ENG.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2022) McMaster University

(Computational Science and Engineering) Hamilton, Ontario, Canada

TITLE:	Data-driven Supply Chain Monitoring and Optimization
AUTHOR:	Jing Wang
	M.Sc. (Electrical Engineering),
	Zhejiang University, Hangzhou, China
	B.Eng. (Electrical Engineering and Automation),
	Zhejiang University, Hangzhou, China
SUPERVISORS:	Dr. Christopher L.E. Swartz, Dr. Kai Huang
NUMBER OF PAGES:	xiii, 233

Abstract

In the era of Industry 4.0, conventional supply chains are undergoing a transformation into digital supply chains with the wide application of digital technologies such as big data, cloud computing, and Internet of Things. A digital supply chain is an intelligent and value-driven process that has superior features such as speed, flexibility, transparency, and real-time inventory monitoring and management. This concept is further included in the framework of Supply Chain 4.0, which emphasizes the connection between supply chain and Industry 4.0. In this context, data analytics for supply chain management presents a promising research opportunity. This thesis aims to investigate the use of data analytics in supply chain decision-making, including modelling, monitoring, and optimization.

First, this thesis investigates supply chain monitoring (SCMo) using data analytics. The goal of SCMo is to raise an alarm when abnormal supply chain events occur and identify the potential reason. We propose a framework of SCMo based on a data-driven method, principal component analysis (PCA). Within this framework, supply chain data such as inventory levels and customer demand are collected, and the normal operating conditions of a supply chain are characterized using PCA. Fault detection and diagnosis are implemented by examining the monitoring statistics and variable contributions. A supply chain simulation model is developed to carry out the case studies. The results show that dynamic PCA (DPCA) successfully detected abnormal behaviour of the supply chain, such as transportation delay, low production rate, and supply shortage. Moreover, the contribution plot is shown to be effective in interpreting the abnormality and identify the fault-related variables. The method of using data-driven methods for SCMo is named data-driven SCMo in this work.

Then, a further investigation of data-driven SCMo based on another statistical process monitoring method, canonical variate analysis (CVA), is conducted. CVA utilizes the state-space model of a system and determines the canonical states by maximizing the correlation between the combination of past system outputs and inputs and the combination of future outputs. A state-space model of supply chain is developed, which forms the basis of applying CVA to detect supply chain faults. The performance of CVA and PCA are assessed and

compared in terms of dimensionality reduction, false alarm rate, missed detection rate, and detection delay. Case studies show that CVA identifies a smaller system order than PCA and achieves comparable performance to PCA in a lower-dimensional latent space.

Next, we investigate data-driven supply chain control under uncertainty with risk taken into account. The method under investigation is reinforcement learning (RL). Within the RL framework, an agent learns an optimal policy that maps the state to action during the process of interacting with the non-deterministic environment, such that a numerical reward is maximized. The current literature regarding supply chain control focuses on conventional RL that maximizes the expected return. However, this may be not the best option for risk-averse decision makers. In this work, we explore the use of safe RL, which takes into account the concept of risk in the learning process. Two safe RL algorithms, $\hat{Q}$-learning and β-pessimistic Q-learning, are investigated. Case studies are carried out based on the supply chain simulator developed using agent-based modelling. Results show that Q-learning has the best performance under normal scenarios, while safe RL algorithms perform better under abnormal scenarios and are more robust to changes in the environment. Moreover, we find that the benefits of safe RL are more pronounced in a closed-loop supply chain.

Finally, we investigate real-time supply chain optimization. The operational optimization problems for supply chains of realistic size are often large and complex, and solving them in real time can be challenging. This work aims to address the problem by using a deep learning-based model predictive control (MPC) technique. The MPC problem for supply chain operation is formulated based on the state space model of a supply chain, and the optimal state-input pairs are precomputed in the offline phase. Then, a deep neural network is built to map the state to input, which is then used in the online phase to reduce solution time. We propose an approach to implement the deep learning-based MPC method when there are delayed terms in the system, and a heuristic approach to feasibility recovery for mixed-integer MPC, with binary decision variables taken into account. Case studies show that compared with solving the nominal MPC problem online, deep learning-based MPC can provide near-optimal solution at a lower computational cost.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Dr. Christopher Swartz and Dr. Kai Huang, who provided their kind supports and guidance throughout my PhD study. Their extensive expertise and valuable insights into the research area are critical to the completion of this work. Their excellent mentorship helped me move forward with my goal as a researcher. Working together with them has been an enjoyable experience.

I would like to thank my supervisory committee members, Dr. Prashant Mhaskar and Dr. Antoine Deza. Their inspiring research questions and helpful suggestions improved the quality of this work.

I would like to thank ProSensus Inc. and Dr. Brandon Corbett for getting involved with the research project presented in Chapter 2. Dr. Corbett's opinions and inputs helped strengthen the work.

I would like to acknowledge the support funding provided by the MacData Institute for the research project presented in Chapter 4. Working as a MacDATA Graduate Fellow was a valuable experience.

I would like to acknowledge the support from the McMaster Advanced Control Consortium (MACC). My sincere thanks go to my colleagues and the MACC members. I benefited from the collaborative work environment in MACC.

Last but not least, I gratefully appreciate the support from my family and friends.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we first introduce some basics of supply chain systems. Then, the main research contributions and the outline of this thesis are presented.

1.1 Supply Chain

A supply chain (SC) is a set of entities that are directly involved in the upstream and downstream flows of products, services, finances and information from sources to customers (Mentzer et al., 2001). These entities include suppliers, manufacturers, distributors, and retailers. The structure of a supply chain network is shown in Figure 1.1. The supply chain members are typically interlinked through a material flow from the upstream suppliers to

the downstream customers and an information flow in the opposite direction. With the cooperation between agents, a supply chain deals with the procurement of raw materials, the manufacture of products, and the distribution of products to satisfy customers' demands.

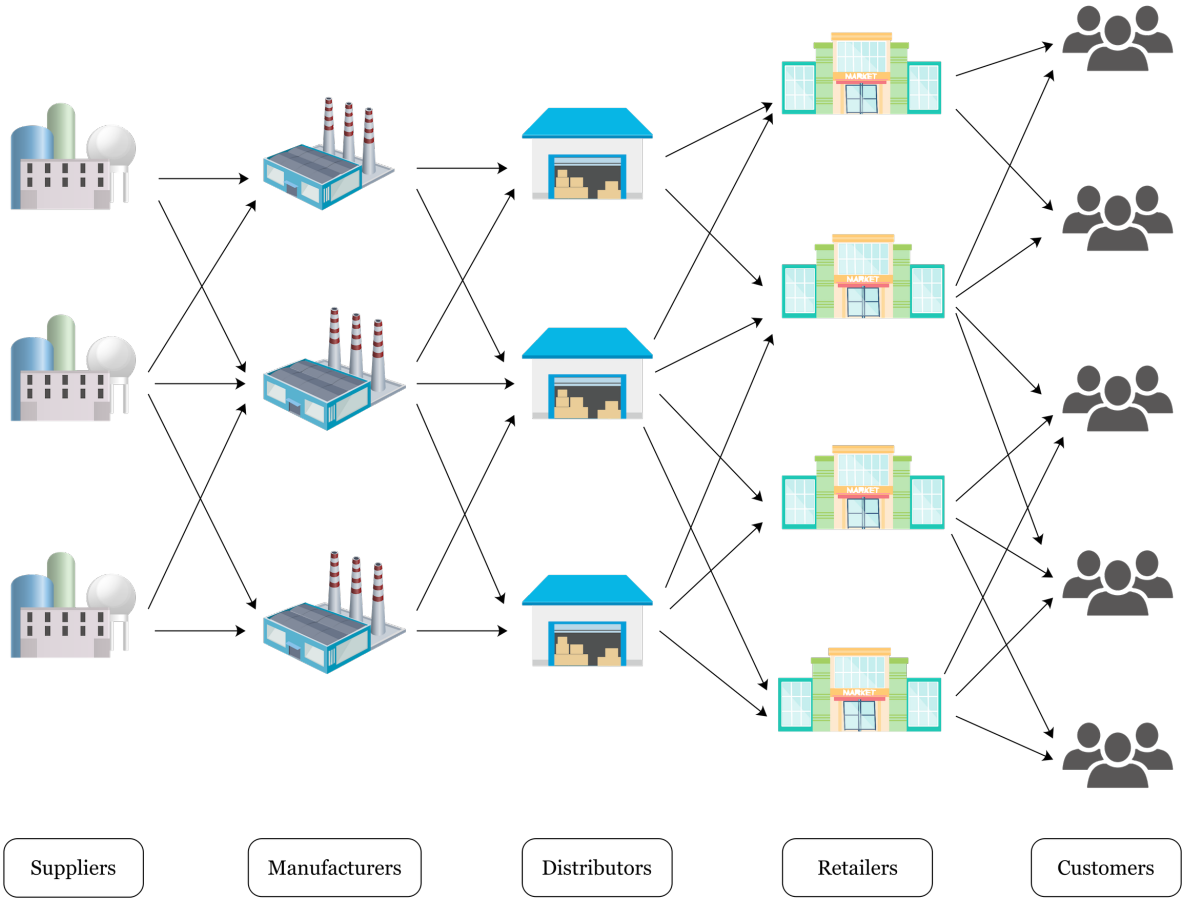


Figure 1.1. A supply chain network

A well-functioning supply chain is important to the general economic performance of an enterprise. *Supply chain management* (SCM) involves systemic and strategic coordination of business activities across business functions and across businesses within the supply chain (Mentzer et al., 2001; Papageorgiou, 2009). Supply chain management and optimization aim at satisfying customers' demands at minimum overall cost (Al-Othman et al., 2008; Mohammadi Bidhandi and Mohd Yusuff, 2011; Benyoucef et al., 2013; Castillo-Villar et al., 2017), bringing greater profit (Awudu and Zhang, 2013; Ma and Li, 2018; Zhou et al., 2019), and/or improving the customer service (Gupta et al., 2000; Mastragostino et al., 2014), through efficient coordination and integration of network design, procurement, production,

transportation, and distribution. Good SCM improves the long-term performance of the supply chain participants and the entire supply chain, and therefore helps maintain the competitiveness of supply chain systems (Mentzer et al., 2001; Mastragostino et al., 2014; Patel and Swartz, 2019).

Generally, decision-making for supply chains is categorized into the following three classes according to the planning horizon and the importance of decisions (Gupta and Maranas, 2003; Santoso et al., 2005; Stadtler et al., 2015; Silver et al., 2017):

(1) Long-term design and planning, or *strategic* planning. Planning at this level determines the acquisition of resources and typically aims to identify an optimal supply chain network configuration over a relatively long time horizon, e.g., more than 2 years. Examples of strategic decisions are the number, location, and capacity of facilities, the products to sell, the technology to be establish, long-term raw material and energy contracts, and nature of production planning and inventory management decision systems. Strategic decisions are the prerequisites for the future development of a supply chain and their effects are perceptible over several years. Strategic supply chain planning is investigated by, e.g., Tsiakis et al., 2001; Guillén et al., 2005; Azaron et al., 2008; Azaron, 2013; Singh et al., 2013; Tong et al., 2014; Hamta et al., 2015; Badri et al., 2016; Badri et al., 2017; Chatzikontidou et al., 2017; Jerbia et al., 2018; Zhen et al., 2019; Zhou et al., 2019.

(2) Mid-term planning, which is called either *tactical* planning (Silver et al., 2017) or *operational* planning (Stadtler et al., 2015) in the literature. Planning at this level focuses on an intermediate time horizon, e.g., 6–24 months, and determines the utilization of resources within the scope of strategic plans. Examples of mid-term decisions are rough quantities and times for the flows and resources in the supply chain, operation hours of plants, work force sizes, inventory levels, subcontracting levels, output rates, and transportation modes. Mid-term supply chain planning is investigated by, e.g., Gupta et al., 2000; Gupta and Maranas, 2000; Gupta and Maranas, 2003; Al-Othman et al., 2008; You et al., 2009; Awudu and Zhang, 2012; Mirzapour Al-e-hashem et al., 2013; Azadeh et al., 2014; Azadeh and Vafa Arani, 2016; Esmaeilikia et al., 2016; Shabani et al., 2014.

(3) Short-term planning, or *operational* planning/control. Planning at this level deals with

specifying the detailed instructions for instant execution and control in a relatively short time horizon, e.g., from a few days to 3 months. Examples of short-term decisions are what to produce or procure, when, on what machine (from which vendor), in what quantity, and in what order. Operational supply chain control is investigated by, e.g., Schütz and Tomasgard, 2011 and Mastragostino et al., 2014.

The environment under which a supply chain is operated is not always stable. The presence of uncertainty in supply chains, such as uncertain demand (Gupta et al., 2000; Gupta and Maranas, 2003), supply (Shabani et al., 2014; Hamdan and Diabat, 2019), process yield (Mastragostino et al., 2014), and product price (Awudu and Zhang, 2013; Azadeh et al., 2014), makes SCM a complex task (Tsiakis et al., 2001). Modern supply chain networks are prone to uncertainty due to their increasing complexity and interrelation. Various type of risks exist due to uncertainties that are ubiquitous within modern supply chains. Supply chain risks can be categorized into two classes (Tang, 2006): (1) Operational risks that are caused by the inherent uncertainties, for example, uncertainties in demand, supply, yield and transportation; (2) Disruption risks such as natural or man-made disasters, which have a greater impact on supply chains. Unexpected events may disrupt supply chain operations and cause substantial negative effects that propagate across the supply chain (Bansal et al., 2005; Fernández et al., 2015; Fernández et al., 2016). The uncertainties and risks in supply chain systems are the main motivation of this work. In this work, we aim to develop data-driven methods to support supply chain modelling and decision-making under uncertainty.

With the wide application of digital technologies such as big data, cloud computing and Internet of Things (IoT), conventional supply chains are undergoing a transformation into digital supply chains (Büyüközkan and Göçer, 2018; He et al., 2020). A digital supply chain is an intelligent and value-driven process that has superior features such as speed, flexibility, transparency, and real-time inventory monitoring and management. Although in its infancy, a digital supply chain possesses great potential in value creation. This concept is further included in the framework of *Supply Chain 4.0* to emphasize the connection between supply chain and *Industry 4.0*, which is a framework involving cyber-physical integration in supply chains (Frederico et al., 2020; Ivanov and Dolgui, 2021; Govindan et al., 2022).

In the era of Supply Chain 4.0, data are an important resource that can be harnessed. Data analytics for supply chain management and optimization presents a promising opportunity, and has gained much attention in both academic research and industrial applications (Nguyen et al., 2018; Ning and You, 2019). Data analytics has been applied to different supply chain functions, for example, procurement, manufacturing, logistics and transportation, warehousing, and demand management (Nguyen et al., 2018). The process of using information and analytical tools to improve supply chain decision-making is defined as *supply chain analytics* (SCA) (Souza, 2014). SCA is generally categorized into the following three types (Souza, 2014; Wang et al., 2016; Tiwari et al., 2018):

(1) Descriptive analytics, which focuses on explaining what is happening or has happened in the supply chain and why.

(2) Predictive analytics, which aims to predict what will happen or will be likely to happen in the supply chain.

(3) Prescriptive analytics, which explores what should be happening and provides decision recommendations.

In this thesis, we focus on investigating the use of data analytics in operational supply chain decision-making, including modelling, monitoring and optimization. The methods adopted can be classified as descriptive and prescriptive SCA.

1.2 Main Contributions

In this thesis, several research gaps in the current literature are identified and addressed. The main research contributions of this thesis are presented as follows:

1. We proposed a data-driven supply chain monitoring (SCMo) framework based on principal component analysis (PCA). This method can be used for fault detection and diagnosis of supply chain systems. It can be classified as descriptive SCA.

2. We developed a state-space model of supply chain, and further extended the scope of data-driven SCMo using canonical variate analysis (CVA). This method can be classified as descriptive SCA. Comparative study was carried out to assess the performance of CVA and PCA in SCMo.

3. We developed a supply chain simulator using agent-based modelling. This simulator not only helps understand and evaluate the supply chain under different scenarios, but also enables the implementation of supply chain control policies in this work.

4. We developed a risk-averse supply chain control method using safe reinforcement learning (RL). Compared with conventional RL, the safe RL policies takes into account the concept of risk by incorporating the worst-case return in the optimization criterion. This method can be classified as prescriptive SCA. The performance of safe RL polices is examined and compared with a classical Order-Up-To inventory policy and a conventional RL policy. This study extends the supply chain control from forward supply chains to closed-loop supply chains using RL, and from conventional RL to safe RL.

5. We developed a deep learning-based model predictive control (MPC) method for real-time supply chain optimization. We proposed an approach to implement the method in the case where there are delayed decisions in the system. Moreover, a heuristic method was proposed to deal with the binary decision variables for deep learning-based mixed-integer MPC. Furthermore, we discussed the approach to data generation. Compared with MPC, this method can reduce the online computation time. This method can be classified as prescriptive SCA.

1.3 Thesis Outline

The research work in this thesis is composed of four components that are presented in Chapters 2–5. The remainder of this thesis is structured into the following chapters.

In Chapter 2, we first present a data-driven SCMo framework. The method of using PCA to detect and diagnose supply chain faults is described. Then, we present the details of the development of a supply chain simulator using agent-based modelling. Next, two case studies are carried out to validate the proposed method and the results are given, followed by the conclusions.

In Chapter 3, we start with a comprehensive literature review on SCMo and data-driven process monitoring. Then, the development of the state-space model of supply chain is described and data-driven SCMo based on CVA is presented. Hyperparameter tuning is also discussed. Next, two case studies are carried out to compare the performance of CVA and PCA, followed by the conclusions.

In Chapter 4, we first present a comprehensive literature review on RL-based supply chain control. Then, we present a risk-averse supply chain control method based on safe RL. The development of the supply chain simulator and the details of the simulation procedure are described. Next, two case studies are carried out to validate the safe RL polices. The performance of the safe RL polices is compared against a classical Order-Up-To policy and a conventional RL policy.

In Chapter 5, we begin by developing the MPC model for operational control of supply chain systems. Then, we present a deep learning-based MPC method for real-time supply chain control. Moreover, we propose an approach to implement the method for applications where there are delayed decisions in the system, and a heuristic method to deal with the binary decision variables for deep learning-based mixed-integer MPC. Next, two case studies are carried out to validate and assess the proposed control method.

In Chapter 6, a brief summary and concluding remarks on the contributions of this thesis are presented. Then, future research directions are identified and presented.

References

Awudu, I. and J. Zhang (2012). "Uncertainties and sustainability concepts in biofuel supply chain management: A review". In: *Renewable and Sustainable Energy Reviews* 16.2, pp. 1359–1368.

— (2013). "Stochastic production planning for a biofuel supply chain under demand and price uncertainties". In: *Applied Energy* 103, pp. 189–196.

Azadeh, A. and H. Vafa Arani (2016). "Biodiesel supply chain optimization via a hybrid system dynamics-mathematical programming approach". In: *Renewable Energy* 93, pp. 383–403.

Azadeh, A., H. Vafa Arani, and H. Dashti (2014). "A stochastic programming approach towards optimization of biofuel supply chain". In: *Energy* 76, pp. 513–525.

Azaron, A., K. Brown, S. Tarim, and M. Modarres (2008). "A multi-objective stochastic programming approach for supply chain design considering risk". In: *International Journal of Production Economics* 116.1, pp. 129–138.

Azaron, A. (2013). "A multi-objective stochastic programming model for designing supply chain networks". In: *IFAC Proceedings Volumes* 46.9, pp. 1956–1960.

Badri, H., S. Fatemi Ghomi, and T. Hejazi (2016). "A two-stage stochastic programming model for value-based supply chain network design". In: *Scientia Iranica* 23.1, pp. 348–360.

Badri, H., S. Fatemi Ghomi, and T.-H. Hejazi (2017). "A two-stage stochastic programming approach for value-based closed-loop supply chain network design". In: *Transportation Research Part E: Logistics and Transportation Review* 105, pp. 1–17.

Bansal, M., A. Adhitya, R. Srinivasan, and I. Karimi (2005). "An online decision support framework for managing abnormal supply chain events". In: ed. by L. Puigjaner and A. Espuña. Vol. 20. Computer Aided Chemical Engineering, pp. 985–990.

Benyoucef, L., X. Xie, and G. A. Tanonkou (2013). "Supply chain network design with unreliable suppliers: a Lagrangian relaxation-based approach". In: *International Journal of Production Research* 51.21, pp. 6435–6454.

Büyüközkan, G. and F. Göçer (2018). "Digital Supply Chain: Literature review and a proposed framework for future research". In: *Computers in Industry* 97, pp. 157–177.

Castillo-Villar, K. K., S. Eksioglu, and M. Taherkhorsandi (2017). "Integrating biomass quality variability in stochastic supply chain modeling and optimization for large-scale biofuel production". In: *Journal of Cleaner Production* 149, pp. 904–918.

Chatzikontidou, A., P. Longinidis, P. Tsiakis, and M. C. Georgiadis (2017). "Flexible supply chain network design under uncertainty". In: *Chemical Engineering Research and Design* 128, pp. 290–305.

Esmaeilikia, M., B. Fahimnia, J. Sarkis, K. Govindan, A. Kumar, and J. Mo (2016). "Tactical supply chain planning models with inherent flexibility: definition and review". In: *Annals of Operations Research* 244, pp. 407–427.

Fernández, E., V. Bogado, E. Salomone, and O. Chiotti (2016). "Framework for modelling and simulating the supply process monitoring to detect and predict disruptive events". In: *Computers in Industry* 80, pp. 30–42.

Fernández, E., C. M. Toledo, M. R. Galli, E. Salomone, and O. Chiotti (2015). "Agent-based monitoring service for management of disruptive events in supply chains". In: *Computers in Industry* 70, pp. 89–101.

Frederico, G. F., J. A. Garza-Reyes, A. Anosike, and V. Kumar (2020). "Supply Chain 4.0: concepts, maturity and research agenda". In: *Supply Chain Management: An International Journal* 25.2, pp. 262–282.

Govindan, K., D. Kannan, T. B. Jørgensen, and T. S. Nielsen (2022). "Supply Chain 4.0 performance measurement: A systematic literature review, framework development, and empirical evidence". In: *Transportation Research Part E: Logistics and Transportation Review* 164, p. 102725.

Guillén, G., F. Mele, M. Bagajewicz, A. Espuña, and L. Puigjaner (2005). "Multiobjective supply chain design under uncertainty". In: *Chemical Engineering Science* 60.6, pp. 1535–1553.

Gupta, A. and C. D. Maranas (2000). "A two-stage modeling and solution framework for multisite midterm planning under demand uncertainty". In: *Industrial & Engineering Chemistry Research* 39.10, pp. 3799–3813.

— (2003). "Managing demand uncertainty in supply chain planning". In: *Computers & Chemical Engineering* 27.8, pp. 1219–1227.

Gupta, A., C. D. Maranas, and C. M. McDonald (2000). "Mid-term supply chain planning under demand uncertainty: customer demand satisfaction and inventory management". In: *Computers & Chemical Engineering* 24.12, pp. 2613–2621.

Hamdan, B. and A. Diabat (2019). "A two-stage multi-echelon stochastic blood supply chain problem". In: *Computers & Operations Research* 101, pp. 130–143.

Hamta, N., M. A. Shirazi, S. F. Ghomi, and S. Behdad (2015). "Supply chain network optimization considering assembly line balancing and demand uncertainty". In: *International Journal of Production Research* 53.10, pp. 2970–2994.

He, L., M. Xue, and B. Gu (2020). "Internet-of-things enabled supply chain planning and coordination with big data services: Certain theoretic implications". In: *Journal of Management Science and Engineering* 5.1, pp. 1–22.

Ivanov, D. and A. Dolgui (2021). "A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0". In: *Production Planning & Control* 32.9, pp. 775–788.

Jerbia, R., M. Kchaou Boujelben, M. A. Sehli, and Z. Jemai (2018). "A stochastic closed-loop supply chain network design problem with multiple recovery options". In: *Computers & Industrial Engineering* 118, pp. 23–32.

Ma, H. and X. Li (2018). "Closed-loop supply chain network design for hazardous products with uncertain demands and returns". In: *Applied Soft Computing* 68, pp. 889–899.

Mastragostino, R., S. Patel, and C. L. Swartz (2014). "Robust decision making for hybrid process supply chain systems via model predictive control". In: *Computers & Chemical Engineering* 62, pp. 37–55.

Mentzer, J. T., W. DeWitt, J. S. Keebler, S. Min, N. W. Nix, C. D. Smith, and Z. G. Zacharia (2001). "Defining supply chain management". In: *Journal of Business Logistics* 22.2, pp. 1–25.

Mirzapour Al-e-hashem, S., A. Baboli, and Z. Sazvar (2013). "A stochastic aggregate production planning model in a green supply chain: Considering flexible lead times, nonlinear purchase and shortage cost functions". In: *European Journal of Operational Research* 230.1, pp. 26–41.

Mohammadi Bidhandi, H. and R. Mohd Yusuff (2011). "Integrated supply chain planning under uncertainty using an improved stochastic approach". In: *Applied Mathematical Modelling* 35.6, pp. 2618–2630.

Nguyen, T., L. Zhou, V. Spiegler, P. Ieromonachou, and Y. Lin (2018). "Big data analytics in supply chain management: A state-of-the-art literature review". In: *Computers and Operations Research* 98, pp. 254–264.

Ning, C. and F. You (2019). "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming". In: *Computers & Chemical Engineering* 125, pp. 434–448.

Al-Othman, W. B., H. M. Lababidi, I. M. Alatiqi, and K. Al-Shayji (2008). "Supply chain optimization of petroleum organization under uncertainty in market demands and prices". In: *European Journal of Operational Research* 189.3, pp. 822–840.

Papageorgiou, L. G. (2009). "Supply chain optimisation for the process industries: Advances and opportunities". In: *Computers & Chemical Engineering* 33.12, pp. 1931–1938.

Patel, S. and C. L. Swartz (2019). "Supply chain design with time-limited transportation contracts". In: *Computers & Chemical Engineering*, p. 106579.

Santoso, T., S. Ahmed, M. Goetschalckx, and A. Shapiro (2005). "A stochastic programming approach for supply chain network design under uncertainty". In: *European Journal of Operational Research* 167.1, pp. 96–115.

Schütz, P. and A. Tomasgard (2011). "The impact of flexibility on operational supply chain planning". In: *International Journal of Production Economics* 134.2, pp. 300–311.

Shabani, N., T. Sowlati, M. Ouhimmou, and M. Rönnqvist (2014). "Tactical supply chain planning for a forest biomass power plant under supply uncertainty". In: *Energy* 78, pp. 346–355.

Silver, E. A., D. F. Pyke, and D. J. Thomas (2017). *Inventory and Production Management in Supply Chains*. 4th. CRC Press, Taylor & Francis Group, Boca Raton.

Singh, A. R., R. Jain, and P. K. Mishra (2013). "Capacities-based supply chain network design considering demand uncertainty using two-stage stochastic programming". In: *The International Journal of Advanced Manufacturing Technology volume* 69, pp. 555–562.

Souza, G. C. (2014). "Supply chain analytics". In: *Business Horizons* 57.5, pp. 595–605.

Stadtler, H., C. Kilger, and H. Meyr, eds. (2015). *Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies*. 5th. Springer Berlin, Heidelberg.

Tang, C. S. (2006). "Perspectives in supply chain risk management". In: *International Journal of Production Economics* 103.2, pp. 451–488.

Tiwari, S., H. Wee, and Y. Daryanto (2018). "Big data analytics in supply chain management between 2010 and 2016: Insights to industries". In: *Computers & Industrial Engineering* 115, pp. 319–330.

Tong, K., J. Gong, D. Yue, and F. You (2014). "Stochastic programming approach to optimal design and operations of integrated hydrocarbon biofuel and petroleum supply chains". In: *ACS Sustainable Chemistry & Engineering* 2.1, pp. 49–61.

Tsiakis, P., N. Shah, and C. C. Pantelides (2001). "Design of multi-echelon supply chain networks under demand uncertainty". In: *Industrial & Engineering Chemistry Research* 40.16, pp. 3585–3604.

Wang, G., A. Gunasekaran, E. W. Ngai, and T. Papadopoulos (2016). "Big data analytics in logistics and supply chain management: Certain investigations for research and applications". In: *International Journal of Production Economics* 176, pp. 98–110.

You, F., J. M. Wassick, and I. E. Grossmann (2009). "Risk management for a global supply chain planning under uncertainty: Models and algorithms". In: *AIChE Journal* 55.4, pp. 931–946.

Zhen, L., L. Huang, and W. Wang (2019). "Green and sustainable closed-loop supply chain network design under uncertainty". In: *Journal of Cleaner Production* 227, pp. 1195–1209.

Zhou, X., H. Zhang, R. Qiu, M. Lv, C. Xiang, Y. Long, and Y. Liang (2019). "A two-stage stochastic programming model for the optimal planning of a coal-to-liquids supply chain under demand uncertainty". In: *Journal of Cleaner Production* 228, pp. 10–28.

Chapter 2

Data-driven Supply Chain Monitoring Using Principal Component Analysis

The study in this chapter has been published and presented in:

[1] Jing Wang, Christopher L.E. Swartz, Brandon Corbett, and Kai Huang. "Supply chain monitoring using principal component analysis". *Industrial & Engineering Chemistry Research* 2020 59 (27), 12487-12503.

[2] Jing Wang, Christopher L.E. Swartz, and Kai Huang. "Supply chain monitoring based on principal component analysis". *2020 Virtual AIChE Annual Meeting*, November

16-20, 2020.

Various types of risks exist in a supply chain, and disruptions could lead to economic loss or even break-down of a supply chain without an effective mitigation strategy. The ability to detect disruptions early can help improve the resilience of the supply chain. In this paper, the application of principal component analysis (PCA) and dynamic PCA (DPCA) in fault detection and diagnosis of a supply chain system is investigated. In order to monitor the supply chain, data such as inventory levels, market demands and amount of products in transit are collected. PCA and DPCA are used to model the normal operating conditions (NOC). Two monitoring statistics, the Hotelling's T^2 and the squared prediction error (SPE), are used to detect abnormal operation of the supply chain. The confidence limits of these two statistics are estimated from the training data based on the χ^2- distributions. The contribution plots are used to identify the variables with abnormal behavior when at least one statistic exceeds its limit. Two case studies are presented - a multi-echelon supply chain for single product that includes a manufacturing process, and a gas bottling supply chain with multiple products. In order to validate the proposed method, supply chain simulation models are developed using the programming language `Python 3.7`, and simulated data is collected for analysis. PCA and DPCA are applied to the data using the `scikit-learn` machine learning library for Python. The results show that abnormal operation due to transportation delay, supply shortage, and poor manufacturing yield can be detected. The contribution plots are useful for interpreting and identifying the abnormality.

2.1 Introduction

A supply chain is a network of suppliers, manufacturing facilities, warehouses, retailers, and customers that deals with the procurement of raw materials, the manufacture of products, and the distribution of products to the customers. There are material flow from upstream suppliers to downstream customers, and information flow in the opposite direction (Patel and Swartz, 2019; Mastragostino et al., 2014). The importance of a well operating supply chain to the overall economic performance of an enterprise has underpinned a large body of research on supply chain operation and design across several disciplines. Reviews of supply chain studies from a process systems engineering perspective are given by Grossmann (2005), Shah (2005), and Papageorgiou (2009).

There are various types of risks in supply chain operation, including delays, poor yield or quality at supply source, procurement failures, inaccurate forecasts, uncertain consumer demands, and disruptions like natural disaster (Chopra and Sodhi, 2004; Tang, 2006a). For example, in the case study of Carvalho et al. (2012), the supply delay is regarded as the main disturbance that negatively affects the automaker. Wilson (2007) investigates the impact of transportation disruptions on the supply chain performance. The disturbances may cause financial loss or even break-down of the supply chain without an effective mitigation strategy (Tang, 2006b). Improving the ability to detect disturbances in a supply chain quickly can help reduce the risks and substantially increase the resilience of the supply chain (Chopra and Sodhi, 2014; Sheffi, 2015). The purpose of this paper is to investigate the application of data analytics for supply chain monitoring, fault detection and diagnosis.

Data analytics is becoming more and more important in the era of big data. Data analytics in supply chain management (SCM) has gained much attention in both academic research and industrial applications (Souza, 2014; Wang et al., 2016a; Tiwari et al., 2018). In SCM, data is increasingly employed to capture the trends in costs and performance, monitor inventory, support process control and improve the process, and optimize production (Mishra et al., 2018). Supply chain analytics (SCA), defined by Souza (2014) as the use of information and analytical tools for improved supply chain decision-making, can generally be categorized into

three classes (Souza, 2014; Wang et al., 2016a; Tiwari et al., 2018): (1) descriptive analytics, which aims at explaining what has happened/is happening in the supply chain and why; (2) predictive analytics, which focuses on answering what will be happening or likely to happen in the supply chain; and (3) prescriptive analytics, which explores what should be happening and how to influence it. The relevant techniques involve statistics, programming, mathematical optimization, and simulation (Tiwari et al., 2018).

The applications of data analytics in SCM can also be classified by supply chain functions: procurement, manufacturing, logistics and transportation, warehousing, and demand management, or general SCM (Nguyen et al., 2018). For example, in research on procurement, Jain et al. (2014) investigate a data mining approach to discover the hidden relationships between data used for suppliers' selection and their overall rating based on prior performance. The approach helps in optimizing the selection process of suppliers. Mori et al. (2012) utilize machine learning techniques like support vector machine and logistic regression to build the prediction model of customer-supplier relationships, and help find potential business partners. As for manufacturing, Zhong et al. (2015) use radio frequency identification (RFID) production shop floor data to obtain better estimation of the arrival of customer orders and standard operation times. These parameters are then used to develop a two-level planning and scheduling model for the RFID-enabled real-time ubiquitous shop floor manufacturing. In logistics and transportation, Zhao et al. (2017) extract the upper and lower limits of uncertain parameters from historical data, and use them for the re-design of a green supply chain. Toole et al. (2015) develop a system to estimate the travel demand and infrastructure usage for transportation planning, using massive data generated by mobile computing. Li et al. (2015) employ Lasso Granger causality models to pick the most relevant data to build the traffic prediction model, thus achieving a good balance between model complexity and model performance. In terms of warehousing, Chiang et al. (2011) define an association index and propose a data mining-based storage assignment approach, which improves the efficiency of order picking. Tsai and Huang (2015) use data analytics to capture customer purchase and moving behaviours, and optimize the shelf space allocation. As for demand management, Salehan and Kim (2016) use a sentiment mining approach to investigate the predictors of performance of online consumer reviews; Alain Yee et al. (2016) employ a neural network

to investigate which variables, like online reviews, promotion strategies and sentiments, are important predictors of product sales. Ma et al. (2014) propose a demand trend mining technique for data-driven product design.

A promising application of SCA is the data-driven supply chain optimization under uncertainty, in which techniques of data analytics and machine learning are used to characterize the uncertainty based on process data in order to more accurately represent the uncertainty and reduce conservativeness in the optimization. Ning and You (2017) and Ning and You (2018a) present formulations for data-driven adaptive robust optimization with case studies that include robust planning of chemical process networks under uncertainty. Shang and You (2018) develop a formulation for distributionally robust optimization under uncertainty, which seeks to hedge against inexactness of the probability distribution of the uncertainty. In Gao et al. (2019), the scheme is extended and applied to robust optimization of shale gas supply chains under uncertainty. A recent review of data-driven optimization under uncertainty with perspectives on future research directions is given in Ning and You (2019).

The various applications of SCA reviewed by Wang et al. (2016a), Nguyen et al. (2018), and Tiwari et al. (2018) indicate the power and potential for data analytics in aiding decision-making in many aspects of SCM. This paper focuses on a multivariate statistical method, principal component analysis (PCA). PCA is designed for extracting uncorrelated components from correlated data, as described by Wold et al. (1987). It is known as a latent variable method (LVM), and is also considered as a type of machine learning method (Ge, 2017; Qin and Chiang, 2019). PCA has been successfully applied to industrial process modeling, monitoring and diagnosis (Kresta et al., 1991; Kourti and MacGregor, 1995; MacGregor and Kourti, 1995; Nomikos and MacGregor, 1995; Qin, 2012; Ge et al., 2017). Extensions of PCA have been developed for industrial process analysis, including dynamic PCA (Ku et al., 1995; Russell et al., 2000; Chiang et al., 2001; Li and Qin, 2001; Li et al., 2014a; Li et al., 2016), kernel/nonlinear PCA (Lee et al., 2004), recursive PCA, multi-block, multi-way PCA (Qin, 2012), dynamic inner PCA (Dong and Qin, 2018), and mixtures of probabilistic PCA (Kodamana et al., 2019).

In the current literature, PCA has been applied in some aspects of SCA. In Pozo et al.

(2012), PCA is employed to reduce the computational complexity of the multi-objective optimization problem formulated for supply chain design. The redundant environment metrics are identified and omitted while retaining the main features of the problem. In Li et al. (2014b), PCA is used to reduce the feature dimensions and improve the efficiency of model development. Lei and Moon (2015) use PCA to help determine market segments for new products, and develop a decision support system for market-driven product positioning and design. How and Lam (2018) use PCA to reduce the redundancies of performance indicators, thus aiding the multi-objective optimization of supply chain. In data-driven supply chain optimization, PCA is applied to help characterize uncertain parameters of supply chain, by reducing the dimensionality of the correlated uncertainty data (Ning and You, 2018b; Gao et al., 2019). Mele et al. (2005) demonstrate the application of PCA techniques for detection of manufacturing and transportation delays in a simulated supply chain system. Their study includes a wavelet based multi-scale PCA technique and a Genetic Algorithm based search scheme to account for time delays.

The above applications demonstrate some of the benefits of using PCA in SCA. However, research on applying PCA to monitoring, fault detection and diagnosis of a supply chain as a system is still lacking. In SCA applications of PCA, the monitoring statistics of the scores and residual are largely ignored and the ability of PCA in statistical monitoring has not been fully taken advantage of yet, in contrast to applications of statistical process monitoring using PCA in plant-wide processes such as manufacturing.

The objective of this paper is to investigate the use of PCA in supply chain monitoring, which means keeping track of the operating status of a supply chain as a system. Since there are usually many measurements in a supply chain, monitoring all of them individually could be difficult to implement and inefficient. If the operation of the supply chain is viewed as a process, there exists correlation in the supply chain data. Hence analogous to process monitoring, PCA can potentially be employed to model and monitor the operation status of a supply chain. This means helping to detect variation from the normal operating conditions (NOC) of the supply chain. Within the context of SCA, this research can be classified as descriptive analytics. In order to validate the effectiveness of PCA in supply chain monitoring, simulation is carried out to generate supply chain data, with PCA and dynamic PCA (DPCA)

performed on the simulated data.

The PCA and DPCA are used in this paper, since this is an initial investigation of applying data-driven techniques to supply chain monitoring. It is worth noting that other data-driven techniques, such as the canonical variate analysis (CVA), can also potentially be applied to supply chain monitoring. Since it was introduced to process monitoring over two decades ago (Wang et al., 1997), CVA has been standard textbook material on process monitoring (Chiang et al., 2001). Negiz and Çinar (1997) investigate a statistic based on CVA states to monitor the variation in the state space, and Russell et al. (2000) propose a residual space CVA statistic for fault detection. Jiang et al. (2015a) present CVA-based contributions to identify the fault-related variables. Jiang et al. (2015b) combine CVA and Fisher discriminant analysis to determine the root cause of faults. CVA is shown to perform better than PCA and DPCA (Negiz and Çinar, 1997; Russell et al., 2000), which indicates its potential in supply chain monitoring.

The remainder of this paper is organized as follows: Section 2.2 provides a summary of fault detection and diagnosis using PCA and DPCA. Section 2.3 describes the supply chain simulation model developed in this study, which is employed to generate simulated data for analysis. In Section 2.4, two case studies are introduced, and used to validate the proposed supply chain monitoring method. Conclusions are presented in Section 2.5.

2.2 Principal Component Analysis

A brief introduction of PCA and the fault detection and analysis (FDD) using PCA is given in Sections 2.2.1–2.2.3, respectively. Dynamic PCA is described in Section 2.2.4. The description of supply chain monitoring using PCA/DPCA is presented in Section 2.2.5. The formulas are extracted from Kresta et al. (1991), Kourti and MacGregor (1996), Qin (2012), Li et al. (2016), as well as other references where stated.

2.2.1 Principal component analysis

PCA is designed for extracting uncorrelated components from correlated data. Denote the data collected at time i as a K-dimensional vector $\mathbf{x}_i$, and the data collected over N time periods as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, which contains N samples. Assume that $\mathbf{X}$ has been scaled to zero-mean and unit-variance, then performing PCA on $\mathbf{X}$ corresponds to construction of the following relationships:

$$\mathbf{T} = \mathbf{X}\mathbf{P} \tag{2.1a}$$

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a^T + \mathbf{E} \tag{2.1b}$$

where $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_A]$ is the loading matrix; $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_A]$ is the score matrix, and $\mathbf{t}_a = \mathbf{X}\mathbf{p}_a$, for $a = 1, \ldots, A$, is the sample data of the a-th principal component (PC); and $\mathbf{E}$ is the residual. The loadings are determined by maximizing the variance of the scores. Usually $A < K$ to achieve dimensionality reduction.

PCA can be implemented through an eigen-decomposition on the sample covariance matrix $\mathbf{S} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$, or a singular value decomposition (SVD) on $\mathbf{X}$. Alternatively, the PCs can be efficiently extracted by the nonlinear iterative partial least squares (NIPALS) algorithm, which is actually a variant of the Power method (Wold et al., 1987).

2.2.2 Fault detection

For fault detection, first, a PCA model is built to characterize the normal operating conditions (NOC) from normal operating data. A sample $\mathbf{x}_i$ can be projected to the principal component subspace (PCS) and the residual subspace (RS) respectively:

$$\mathbf{t}_i^T = \mathbf{x}_i^T \mathbf{P} \tag{2.2a}$$

$$\tilde{\mathbf{x}}_i^T = \mathbf{x}_i^T - \mathbf{t}_i^T \mathbf{P}^T = \mathbf{x}_i^T (\mathbf{I} - \mathbf{P}\mathbf{P}^T) \tag{2.2b}$$

Qin (2003) gives a comprehensive review of fault detection indices in statistical process monitoring. In PCA, the Hotelling's T^2 and the squared prediction error (SPE) are widely employed to detect variations from the NOC. The Hotelling's T^2, also known as the D-statistic, of $\mathbf{x}_i$ is defined as:

$$T^2(\mathbf{x}_i) = \mathbf{t}_i^T \Lambda^{-1} \mathbf{t}_i = \mathbf{x}_i^T \mathbf{P} \Lambda^{-1} \mathbf{P}^T \mathbf{x}_i = \sum_{a=1}^{A} \frac{t_{ia}^2}{\lambda_a} \tag{2.3}$$

where $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_A\}$ contains the eigenvalues of $\mathbf{S}$ in descending order, and t_{ia} is the a-th component of $\mathbf{t}_i$.

Even when the data is not multivariate normally distributed, the principal components are approximately independently normally distributed when the number of variables is large, according to the central limit theorem (Sprang et al., 2002; Qin, 2003). When N is large, the distribution of T^2 can be approximated by a χ^2-distribution, and the confidence limit at a significance level α can be calculated as:

$$T_\alpha^2 = \chi_\alpha^2(A) \tag{2.4}$$

Alternatively, the confidence limit can also be estimated by the F-distribution (Kourti and MacGregor, 1995) or the Beta-distribution (MacGregor and Cinar, 2012). The significance level α can be determined based on the users' needs (Wang et al., 2016b).

The SPE, also referred to as the Q-statistic, of $\mathbf{x}_i$ is defined as:

$$\text{SPE}(\mathbf{x}_i) = \|\tilde{\mathbf{x}}_i\|_2^2 = \|(\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{x}_i\|_2^2 = \mathbf{x}_i^T \tilde{\mathbf{P}} \tilde{\mathbf{P}}^T \mathbf{x}_i \tag{2.5}$$

where $\tilde{\mathbf{P}} = [\mathbf{p}_{A+1}, \ldots, \mathbf{p}_K]$ comprises the $(K - A)$ eigenvectors that are not retained. The confidence limit of SPE at a significance level α can be estimated by a weighted χ^2-distribution (Nomikos and MacGregor, 1995):

$$\text{SPE}_\alpha = \frac{v}{2m} \chi_\alpha^2 \left(\frac{2m^2}{v}\right) \tag{2.6}$$

where $m = mean(\text{SPE}), v = var(\text{SPE})$, denote the mean and variance of the SPE, respectively. The approximating distribution using eq 2.6 works well in practice even when the errors are not normal (Sprang et al., 2002; Qin, 2012).

The T^2 and SPE model the variation in the PCS and RS, respectively. They can be used as a pair to indicate significant deviation from the NOC (MacGregor and Cinar, 2012). Their roles in process monitoring are not symmetric. Exceeding the T^2 limit does not necessarily indicate a fault, while probably a shift in the operation region; thus SPE is considered preferable over T^2 for fault detection (Qin, 2003). In this paper, the Hotelling' T^2 and SPE are adopted for supply chain monitoring, with greater emphasis on the SPE, since it is more reliable for non-normal data (Qin, 2012).

2.2.3 Fault diagnosis

Fault detection tells whether there is an abnormality or not. Fault diagnosis is to find the source cause of the deviation (MacGregor and Cinar, 2012) or to determine which fault has occurred (Chiang et al., 2000). A widely used approach for fault diagnosis is the contribution plot (Kourti and MacGregor, 1995; Westerhuis et al., 2000). If a fault is detected, the next step is to identify potential fault-related variables. The variables with large contributions to the T^2 and SPE are considered most likely to be fault related. For a sample $\mathbf{x}_i$, the contribution of the j-th variable x_{ij} to score T_a can be defined as follows (Miller et al., 1998):

$$contribution_{a,j} = x_{ij} p_{aj} \tag{2.7}$$

where p_{aj} is the j-th component of the a-th loading $\mathbf{p}_a$, while the contribution of x_{ij} to the SPE can be defined as:

$$contribution_j = \tilde{x}_{ij}^2 \tag{2.8}$$

where $\tilde{x}_{ij}$ is the j-th component of the residual $\tilde{\mathbf{x}}_i$.

The contribution plot is categorized as a fault diagnosis approach in some works (He et al., 2005; MacGregor and Cinar, 2012; Qin, 2012). It should be noted that although contribution

plots can narrow down the search for fault-related variables, it could not explicitly diagnose the root cause of faults (Chiang et al., 2000). Variables that have inflated the monitoring statistics are first determined by contribution plots, and then related to specific process behaviour or disturbances. Thus it is referred to as an indirect approach to fault diagnosis that replies on the interpretation of the plot (MacGregor and Cinar, 2012). Due to the limitation, it is categorized as a fault isolation technique rather than a fault diagnosis technique by Chiang et al. (2000).

In terms of fault diagnosis, Fisher discriminant analysis (FDA) is shown to have better performance than PCA (Chiang et al., 2000). PCA looks for directions that can efficiently represent the data, while FDA focuses on the discrimination of data (He et al., 2005). In FDA, the process data during faults are collected and categorized into classes, with each class representing a specific fault. Dimensionality reduction is implemented with the separability of the classes maximized. Observations can then be classified by a discriminant function, in a lower-dimensional space. By augmenting the observations with previous observations, FDA is enabled to taken into account the series correlations of data, which is referred to as the dynamic FDA (Chiang et al., 2001; Jiang et al., 2015b).

Other methods developed for fault diagnosis in process monitoring include the fault signature (Yoon and MacGregor, 2001), hierarchical contribution plots (Qin, 2003), causal map combined with data-driven approach (Chiang and Braatz, 2003), trajectory loading and score contribution plots (García-Muñoz et al., 2003), reconstruction based contribution (RBC) (Alcala and Qin, 2009), and the framework integrating dynamic PCA, RBC and Granger causality analysis (Li et al., 2016).

The contribution plots have been demonstrated to be very useful in many applications (Yoon and MacGregor, 2001). They greatly help narrow the scope of potential fault-related variables. Therefore, in this paper, the contribution plots for the score (eq 2.7) and SPE (eq 2.8) are adopted for fault diagnosis. The trajectory contribution plot is also employed. Contribution plots are used in this paper as an initial investigation of applying data-driven techniques to supply chain monitoring. Other approaches, such as FDA, can also potentially be applied for the fault diagnosis of supply chain systems.

2.2.4 Dynamic PCA

The standard (static) PCA deals with correlated data that are time independent. When the observations are time series, the auto-correlation and cross-correlation can be taken into account by dynamic PCA (DPCA). DPCA uses the time lag shift technique to enable the PCA model to capture dynamic behavior of a system (Ku et al., 1995). In DPCA, the observation at time k is augmented by the l previous observations:

$$\mathbf{z}_k = [\mathbf{x}_k^T, \mathbf{x}_{k-1}^T, \ldots, \mathbf{x}_{k-l}^T]^T \tag{2.9}$$

where l is the number of time lags.

DPCA then corresponds to PCA implemented on the following augmented matrix $\mathbf{Z}$ (Li et al., 2016):

$$\mathbf{Z} = [\mathbf{z}_{l+1}, \mathbf{z}_{l+2}, \ldots, \mathbf{z}_N]^T = \begin{bmatrix} \mathbf{x}_{l+1} & \mathbf{x}_{l+2} & \ldots & \mathbf{x}_N \\ \mathbf{x}_l & \mathbf{x}_{l+1} & \ldots & \mathbf{x}_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_{N-l} \end{bmatrix}^T \tag{2.10}$$

The augmented matrix can be equivalently expressed as $\mathbf{Z} = [\mathbf{X}_{l+1}, \mathbf{X}_l, \ldots, \mathbf{X}_1]$ (Ku et al., 1995), where $\mathbf{X}_k = [\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N-l-1}]^T$.

In DPCA models, the time lag needs to be taken into account when calculating the variable contributions. The contribution of one variable to T_a and SPE is summed over the lags (Huang and Yan, 2015), as shown in eq 2.11.

$$Contribution_j = contribution_{j,t} + contribution_{j,(t-1)} + \cdots + contribution_{j,(t-l)} \tag{2.11}$$

2.2.5 Supply chain monitoring using PCA/DPCA

The structure of a supply chain is shown in Figure 2.1. Generally, a supply chain consists of suppliers, manufacturers, warehouses (distribution centers), retailers, and consumers

(customers). At each agent, different inventory management policies can be adopted (Axsäter, 2015). There are various types of flows in a supply chain: material flow, product flow, and service flow which are from upstream to downstream; with information flow and cash flow from downstream to upstream (Lee et al., 2002; Pundoor and Herrmann, 2006). In this paper, the material flow from upstream to downstream, and the order information flow in the opposite direction are considered.

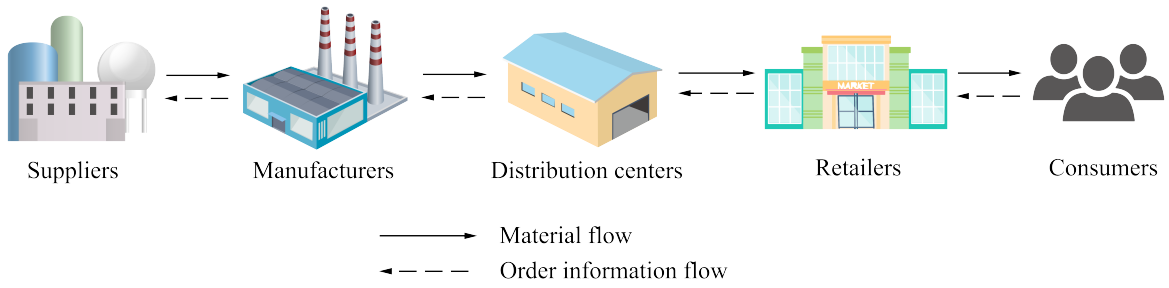


Figure 2.1. The structure of a supply chain.

In process system monitoring, a linear dynamic process can be represented by the following state-space equation (Russell et al., 2000; Jiang et al., 2015a):

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \tag{2.12a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{E}\mathbf{w}(t) + \mathbf{v}(t) \tag{2.12b}$$

where $\mathbf{x}(t), \mathbf{u}(t), \mathbf{y}(t)$ are the state variables, inputs and outputs, respectively; $\mathbf{w}(t), \mathbf{v}(t)$ are independent white noise processes in the state equation eq 2.12a and output equation eq 2.12b, respectively; $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ are coefficient matrices.

The dynamics of a supply chain system can also be modelled by the state-space model, which is typically used in literature on the model predictive control (MPC) of supply chains. For example, in Li and Marlin (2009), Subramanian et al. (2012), and Mastragostino et al. (2014), the state-space model of supply chain is given as:

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{C}\mathbf{d}(t) \tag{2.13}$$

with some constraints on the variables. Depending on the definitions in different works, the

states variables $\mathbf{x}(t)$ could contain inventory levels and backorders; the inputs $\mathbf{u}(t)$ could contain decision variables associated with material procurement, production, transportation and sale quantities; and the disturbances $\mathbf{d}(t)$ could be the demands, yields and delays. In Li and Marlin (2009), a feedback term is added to eq 2.13 to represent the difference between the measured and predicted inventories.

The state-space models eq 2.12a and eq 2.13 imply the similarities between the process systems and supply chain systems. And this relation is further strengthened by the work of Subramanian et al. (2013), which models a supply chain node analogously to a two-tank system, and use distributed MPC for supply chain optimization. Based on these works, the operations of a supply chain could potentially be treated as a dynamic process, which responses to the disturbances in the system. Due to the existence of disturbances, feedback information, and uncertainty in the parameters, statistical methods can potentially be used to monitor supply chain dynamics, just as in process systems. There is no obvious barrier in extending statistical process monitoring to supply chain systems. From this perspective, the 'process variables' of a supply chain, such as the time-varying market demands and inventory levels, are not independent, but correlated. Hence the idea is to apply PCA to analyze supply chain data. Assuming the information of all echelons and agents can be obtained, the correlation of data indicates the potential for PCA in dimensionality reduction and subsequent monitoring.

The procedure adopted for using PCA and DPCA for supply chain monitoring is presented as follows:

1. Training step:

(1) Collect NOC data of the supply chain, such as inventory levels, market demands, and the amount of products in transit;

(2) Preprocessing: augment data for DPCA; normalize the data to have zero mean and unit variation for each variable;

(3) Perform PCA/DPCA, obtain the loadings and scores;

(4) Calculate the monitoring statistics (T^2 and SPE) of the NOC data;

(5) Determine the confidence limits of T^2 and SPE (Qin, 2003; Qin, 2012; Wang et al., 2016b).

2. Monitoring step:

(1) For new data, normalize it with the mean and standard deviation of each variable from the training step;

(2) Project the new data into the PCS and RS to get scores and residuals;

(3) Calculate the T^2 and SPE of the new data;

(4) Check whether T^2 and SPE are both within the confidence limits; if so, then the data can be seen as normal; otherwise, there is an abnormal event, check the contribution plots to identify the fault-related variables.

2.3 Supply Chain Simulation

In order to validate the effectiveness of PCA and DPCA in supply chain monitoring, the simulation model of a supply chain is developed and the simulated data is collected for analysis. This section gives a detailed description of the simulation model developed in this paper.

2.3.1 The role and basics of supply chain simulation

Usually the optimal design of a supply chain network under uncertainty can be formulated as a two-stage stochastic program (Tsiakis et al., 2001), or a robust/adaptive optimization problem with more stages (Ning and You, 2019). Solving the optimization problem will produce an optimal supply chain configuration. However, it cannot tell what will happen after the supply chain design is implemented under arbitrary scenarios (Lee et al., 2002). In

comparison, simulation can help understand and evaluate the supply chain under different scenarios.

For example, inventory policies, such as (R, Q) and (s, S) policies with periodic or continuous review, might be used for the inventory management in a multi-echelon supply chain in reality (Axsäter, 2015). The (R, Q) policy requires that an order with lotsize Q be placed to the upstream when the inventory position falls below the re-order point R. The (s, S) policy is to maintain a target inventory level S when the inventory position falls below the re-order point s. Simulation can help analyze the performance of a supply chain over a time period when using different inventory policies, even with time-varying re-order point. Wilson (2007) uses simulation to study how the transportation disruptions affect the performance of supply chain. Carvalho et al. (2012) use simulation to investigate different mitigation strategies when disruptions occur in the supply chain.

In order to validate the proposed supply chain monitoring method, this paper uses simulation to generate supply chain data for analysis.

2.3.2 Supply chain simulation using Python

There are some softwares designed for supply chain simulation, such as Arena combined with Microsoft Excel (Pundoor and Herrmann, 2006), AnyLogic (Ivanov, 2017), and Supply Chain Analyzer (Lee et al., 2002). In this paper, the supply chain simulation is implemented using the open-source programming language Python 3.7.

The simulation model is developed using the discrete-continuous combined modelling as presented by Lee et al. (2002), where the supply chain elements can be classified into two groups, continuous and discrete. Inventory levels, order information and customer demands are considered as continuous elements, while transportation between agents is considered as discrete element. Object-oriented programming is implemented, which makes the simulation model flexible to be customized for different supply chains. The Python simulation model can be categorized into two parts: (1) The Python modules where the classes for the supply chain participants are defined. Each type of participant is defined as a class, while some

participants share some common attributes and methods. These classes can be debugged individually first before the systematic simulation. (2) The 'main' Python module where the classes are imported, the supply chain structure is defined, and the simulation is run.

For example, the attributes and methods defined for the 'Warehouse' class is shown in Table 2.1.

Table 2.1. Attributes and methods defined for the 'Warehouse' class.

Attributes	name	string, the name of this Warehouse
	total_periods	NumPy array, the time periods for which the simulation is run
	products	list, contains the names of all products stored at this Warehouse
	product	dictionary, created for each of the product in the 'products' list, containing information (key-value pairs) including inventory level, demand, inventory policy, re-order point, target inventory level, lot-size, order size, the amount of product in transit, arrival time of order, etc.
Methods	__init__()	initialize the attributes of the class
	update()	update the inventory profiles after satisfying downstream orders and receiving upstream shipments
	review()	review the inventory level, if the inventory level is below the re-order point, place an order to upstream according to the inventory policy
	collect_data()	collect data and save it as a 2-dimensional numpy array, write it into a .txt file, or comma-separate values (.csv) file
	plot()	plot the profiles

Figure 2.2 shows the causal loop diagram of the supply chain simulation model when all the participants adopt the (s, S) inventory policy. The 'inv' is used as the attribute name for inventory level for short. The 'Retailer' class satisfies the orders from the 'Customer', and places orders to the 'Warehouse'. Similarly, the 'Warehouse' orders from the 'Factory' and satisfies the orders from 'Retailer'. The 'Factory' orders raw materials from the 'Supplier', manufactures the product and delivers it to the 'Warehouse'. The 'Supplier' has raw material in stock while no product. A waiting line (with an attribute name 'waitline') of a participant is a list of its customers. It uses the first in, first out (FIFO) method, which means the first order is first served.

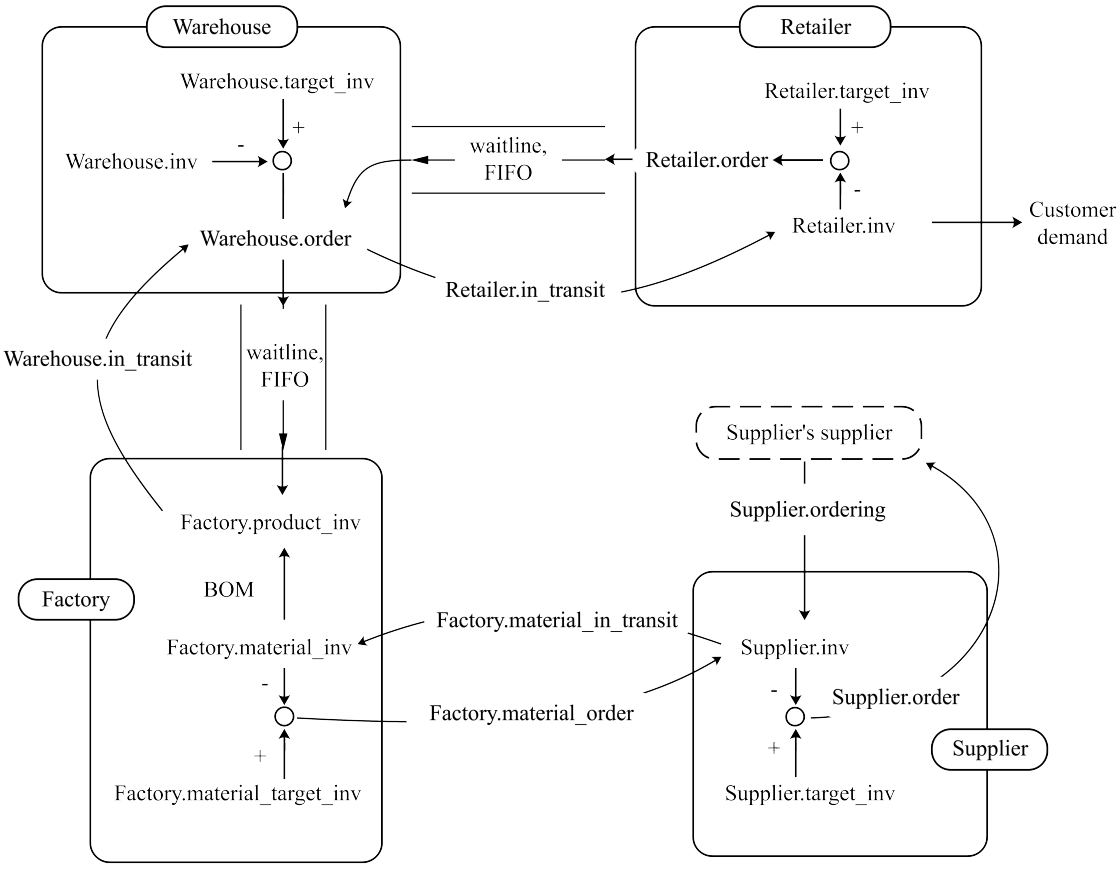


Figure 2.2. The causal loop diagram of the supply chain simulation model.

It is assumed that an agent does not place another order if there is an outstanding order which has not arrived yet. Further demands are satisfied using available stocks. The methods update() and review() for the participants are given as follows, which contain the material balance equations and show how the class attributes are updated at time period i.

Algorithm 2.1: Retailer.update(i):

```
Retailer.inv[i] = Retailer.inv[i-1] - Customer.demand[i]
if Retailer.order_ready == True then
   Retailer.in_transit = Retailer.ordersize
   Retailer.arrival_time = i + transportation_delay
   Retailer.order_ready = False
end if
if i == Retailer.arrival_time then
   Retailer.inv[i] = Retailer.inv[i] + Retailer.in_transit
   Retailer.in_transit = 0
   Retailer.outstand_order = False
end if
```

Algorithm 2.2: Retailer.review(i):

```
if Retailer.inv[i]≤ Retailer.reorder_point and Retailer.outstand_order == False then
   (s, S) policy: Retailer.ordersize = Retailer.target_inv - Retailer.inv[i]
   (r, Q) policy: Retailer.ordersize = Retailer.lotsize
   Retailer.outstand_order = True
end if
```

The attribute 'inv' indicates the inventory level, which is defined by eq 2.14, in the same way as defined by Axsäter (2015):

$$\text{inventory level} = \text{stock on hand} - \text{backorders} \tag{2.14}$$

The backorders are the orders that have been placed but cannot be fulfilled yet due to a shortage. For the Retailer, negative 'inv' means backorders. The Warehouse has a waiting line for its customers, and the orders are satisfied in a FIFO method. If the inventory is not sufficient, then the rest of the orders are recorded as backorders. The backorders will be satisfied once sufficient inventory is available. The method Warehouse.update() is given below. The review and replenishment of inventory for the Warehouse are similar to those of

the Retailer, and hence they are not described here. The attribute 'stock' indicates the stock on hand.

Algorithm 2.3: Warehouse.update(i):

while len(Warehouse.waitline) $>$ 0 **do**
  if Warehouse.stock[i] $\geq$ Warehouse.waitline[0].ordersize **then**
    Warehouse.stock[i] = Warehouse.stock[i-1] - Warehouse.waitline[0].ordersize
    Warehouse.waitline[0].order_ready = True
    Warehouse.waitline.pop(0)
  else
    break
  end if
end while
Warehouse.backorder[i] = $\sum_{\text{Retailer in waitline}}$ Retailer.ordersize
Warehouse.inv[i] = Warehouse.stock[i] - Warehouse.backorder[i]

The Factory uses a 'make-to-order' production system. It starts to manufacture only when an order is received. It has raw materials in stock, while no excess product in stock. The waiting line for the Factory is similar to that of the Warehouse, and the review and replenishment of raw materials for the Factory are also similar to those of the Retailer. The Supplier is similar to the Warehouse, while it is seen as the most upstream and provides raw materials. The material balance equations for manufacturing at the Factory are given in the following Factory.update() method, where Factory.material_BOM is the mass balance coefficient of material.

Algorithm 2.4: Factory.update(i):

for each material **do**
  Factory.material_inv[i] = Factory.material_inv[i-1] - Factory.production[i] $\times$ Factory.material_BOM
end for
Factory.product_inv[i] = Factory.product_inv[i-1] + Factory.production[i]

The Python libraries used for developing the simulation model and analyzing data are listed in Table 2.2. No other third-party library is used in the simulation.

For a supply chain shown in Figure 2.1, the pseudo-code of the simulation is presented in the following procedure. The simulation starts from the most downstream and proceeds to the

Table 2.2. Python libraries used in simulation and analysis.

Python library	Purpose
numpy	some profiles such as inventory levels are stored as numpy arrays; multivariate normal distribution
scipy	the χ^2- distribution (scipy.stats.chi2) is used to calculate the confidence limits of T^2 and SPE
matplotlib.plot	to plot the profiles
pandas	to store data as dataframes, which can then be saved as txt or csv files
sci-kit learn (sklearn)	sklearn.preprocessing.StandScaler for preprocessing data, and sklearn.decomposition.PCA for implementing PCA

most upstream echelon by echelon. First, the demands of the Customers in each time period are generated, for example, as constant values or randomly from statistical distributions, like multivariate Gaussian. Then the Retailers satisfy the demands of Customers, review their inventories and place orders to the Warehouse according to the inventory policy. The Retailers are appended to the waiting lines of the Warehouses. The Warehouses satisfy the orders of Retailers in the waiting lines in a FIFO way, and place orders to the Factories. Once a Retailer's orders are satisfied, it is moved out from the waiting line. The Factories update the attributes in a similar way and manufacture products according to orders from Warehouses. After that the Suppliers update.

2.3.3 PCA and DPCA using Python

The data analysis can be conducted in Python. In this paper, PCA is implemented using the scikit-learn (Pedregosa et al., 2011). The scikit-learn, also known as sklearn, is a machine learning library for Python, which provides various machine learning algorithms. The application program interface (API) sklearn.preprocessing.StandScaler is used to preprocess the collected supply chain data, and then the sklearn.decomposition.PCA is used to implement PCA on the preprocessed data. As for DPCA, there is no off-the-shelf API in scikit-learn, therefore it is coded in Python. The supply chain data is augmented with previous time lags, and then PCA is implemented on the augmented data by sklearn.decomposition.PCA.

Algorithm 2.5: Supply chain simulation

```
import Customer, Retailer, Warehouse, Factory, Supplier
initialize the supply chain participants
generate Customer.demand
for i = 1 to total_periods do
  for each Retailer do
    Retailer.demand[i] ← ∑_downstream Customer.demand[i]
    Retailer.update(i)
    Retailer.review(i)
  end for
  for each Warehouse do
    for each downstream Retailer of Warehouse do
      Warehouse.waitline.append(Retailer)
    end for
    Warehouse.update(i)
    Warehouse.review(i)
  end for
  for each Factory do
    for each downstream Warehouse of Factory do
      Factory.waitline.append(Warehouse)
    end for
    Factory.update(i)
    Factory.review(i)
  end for
  for each Supplier do
    for each downstream Factory of Supplier do
      Supplier.waitline.append(Factory)
    end for
    Supplier.update(i)
    Supplier.review(i)
  end for
end for
```

Also, sklearn.decomposition.PCA does not provide methods for calculating the Hotelling's T^2 and SPE, nor the variable contributions. Therefore, the methods for calculating the two statistics and plotting the variable contributions are coded in Python. The PCA model containing all these methods is coded as a class.

2.4 Case Studies

2.4.1 Case study 1: A multi-echelon supply chain with manufacturing process

In the first case study, a multi-echelon supply chain example with a manufacturing process for a single product is investigated. The structure of this supply chain is shown in Figure 2.3. It consists of 2 suppliers, 1 factory, 1 warehouse, 3 retailers and their customers. At the most upstream are the suppliers, and the most downstream are the customers. The Supplier1 and Supplier2 provide raw materials M_1 and M_2, respectively. The final product A is made from M_1 and M_2 at the Factory, and the production scheme is $0.5M_1 + M_2 \rightarrow A$.

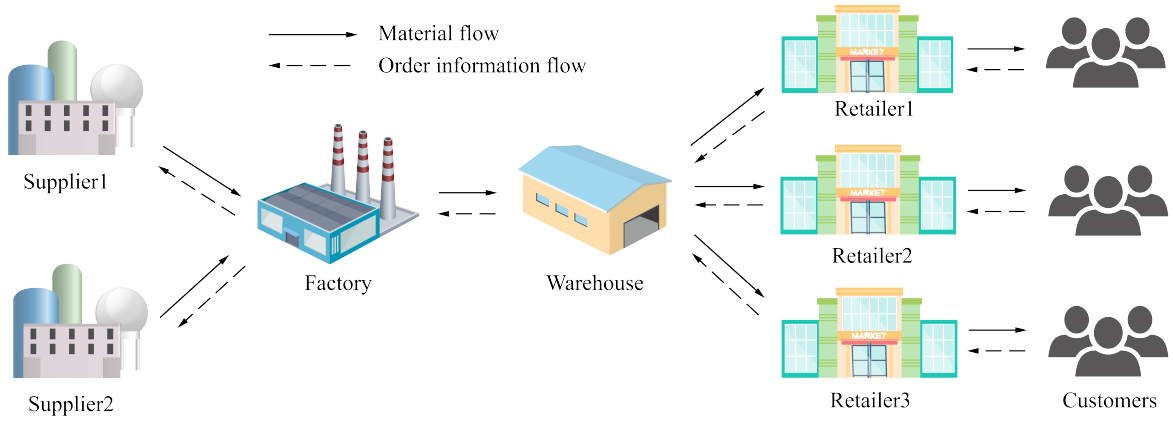


Figure 2.3. Case 1: a multi-echelon supply chain with manufacturing process.

The supply chain network in this case is similar to that of the famous "Beer Distribution Game" (Sterman, 1989), but more complicated in some aspects. The beer game is designed for role-playing simulation of a production and distribution system. They both have 4 echelons.

The supply chain in the beer game is a serial system (Axsäter, 2015), having one agent for each echelon. In Case 1, the wholesaler is not included, while multiple retailers and suppliers are included, which makes the supply chain network more complex.

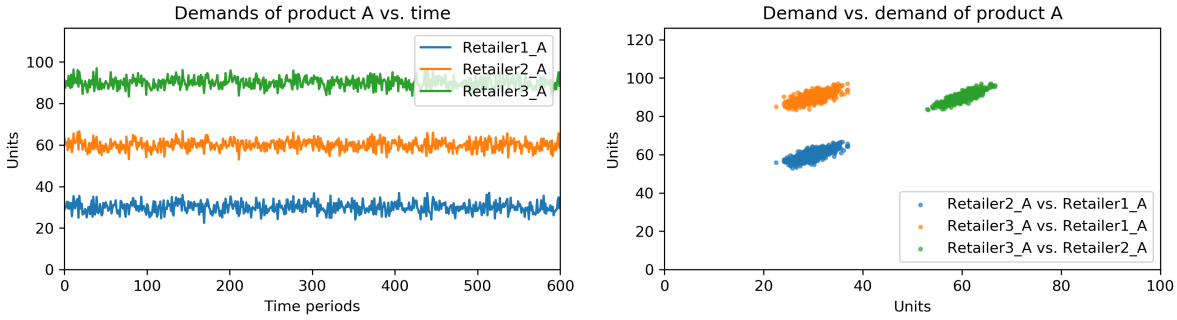
It is assumed that all the participants adopt the (s, S) inventory policy with continuous review, which means the inventory is reviewed in every period. The demands of Customers are generated from a multivariate Gaussian distribution (You and Grossmann, 2008; Ning and You, 2017). The mean vector of the Gaussian distribution is $[30, 60, 90]^T$, as given in Table 2.3, and the covariance matrix is a randomly generated positive semi-definite matrix: $[6.23, 4.37, 4.47; 4.37, 5.62, 4.95; 4.47, 4.95, 6.18]$. For each time period, the Retailers satisfy the demand of Customers if sufficient inventory is available, otherwise backorders are recorded; then they review the inventory and place an order to the Warehouse for replenishment. The Warehouse orders products from the Factory. The Factory follows the 'make-to-order' policy, which means only when there is an incoming order from the Warehouse, it starts to manufacture the product A. It is assumed that the Factory manufactures at a fixed production rate, 400 units per time period, until the order is fulfilled. It has raw materials in stock, but no excess product in stock. The Factory orders raw material M_1 from Supplier1 and M_2 from Supplier2. The parameters of the supply chain, such as the re-order points and target inventory levels, are listed in Table 2.3. The time interval of simulation is one time period, and 600 time periods are simulated. The normal transportation time between two agents is set as one time period.

Table 2.3. Parameters of supply chain.

Participant	demand of A (mean)	initial stocks (units)			target stocks (units)			re-order points (units)		
		A	M_1	M_2	A	M_1	M_2	A	M_1	M_2
Retailer1	30	300	–	–	300	–	–	100	–	–
Retailer2	60	400	–	–	500	–	–	180	–	–
Retailer3	90	400	–	–	800	–	–	200	–	–
Warehouse	–	4000	–	–	5000	–	–	2500	–	–
Factory	–	0	1000	2000	–	1500	3000	–	600	1200
Supplier1	–	–	4000	–	–	4000	–	–	2000	–
Supplier2	–	–	–	6000	–	–	8000	–	–	2500

The demands of Customers at the 3 Retailers over the simulated time periods are shown in

Figure 2.4a. It can be seen the demands fluctuate with time. Figure 2.4b shows the scatter plot of each pair of the demands, which indicates the assumed positive correlation between the demands of Customers. When the demand at one Retailer is high or low, the demands at the other two Retailers also tend to be high or low. The simulated supply chain data are collected, and the inventory profiles of the agents are shown in Figure 2.5a. A total of 21 variables are collected, including the orders received at the Retailers and the Warehouse (4 variables), and the inventory levels of A, M_1, and M_2 of the agents (9 variables), and the amount of products in transit or in processing (8 variables).



(a) Demands over the simulated time periods. (b) Demand versus demand for Retailer pairs.

Figure 2.4. Case 1: demands at the 3 Retailers.

The supply chain operation over 600 time periods is simulated and analyzed. Each time period can be seen as 1 day. The NOC data from time period 0 to 400 is taken as the training set, which is a matrix of 400 rows and 21 columns, denoted as $\mathbf{X}_{train}^{raw}$. For PCA, $\mathbf{X}_{train}^{raw}$ is mean centred and scaled to unit variance to get $\mathbf{X}_{train}$. Then PCA is performed on $\mathbf{X}_{train}$. With PCA, 10 PCs are retained to achieve a R^2 value (the ratio of variance explained) of 85%. For DPCA, the matrix $\mathbf{X}_{train}^{raw}$ is augmented with 2 time lags to get $\mathbf{X}_{train,aug}^{raw}$, which has $21 \times 3 = 63$ columns. It is then preprocessed to get $\mathbf{X}_{train,aug}$, and 18 PCs are extracted to achieve a R^2 value of 82%. A testing set $\mathbf{X}_{test}^{raw}$ from time period 401 to 600 is compiled, and also comprises NOC data. $\mathbf{X}_{test}^{raw}$ is preprocessed to get $\mathbf{X}_{test}$, using the means and standard deviations obtained from the training set. Then $\mathbf{X}_{test}$ is projected into the latent space using the same transformation as that implemented on $\mathbf{X}_{train}$.

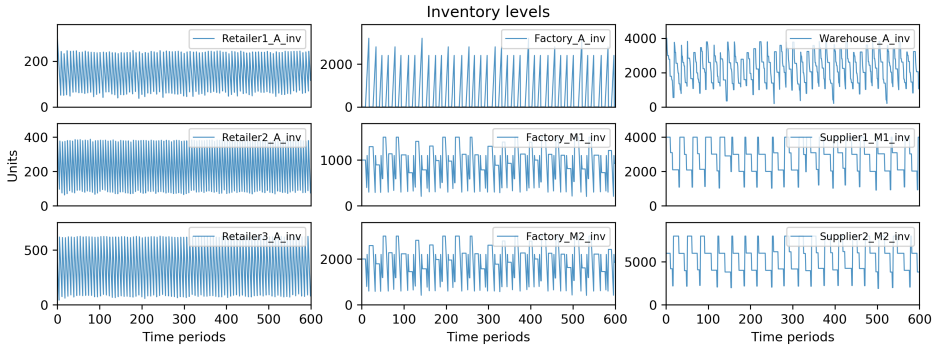
The monitoring charts for the Hotelling's T^2 and SPE using PCA and DPCA are shown in Figure 2.5b and Figure 2.5c, respectively. The confidence limits are estimated from the

training set and plotted as dashed lines. Since the supply chain data are not all normally distributed, in order to reduce the false alarm rates, the significance levels in eq 2.4 and eq 2.6 are tuned to 99% and 99.9%. Exceeding the 99% limit raises a warning that the supply chain could be behaving abnormally, while exceeding the 99.9% limit raises an alarm that it is very likely that some fault has occurred in the supply chain. It is observed that a small portion of NOC samples slightly violate the confidence limits, which are the false alarms. False alarms are not uncommon in fault detection, as in the research of Russell et al. (2000) and Dong and Qin (2018), for example. DPCA shows lower false alarm rate than PCA, which could be potentially due to the significant dynamics in the supply chain data, as can be seen in Figure 2.5a. As shown by Ku et al. (1995), DPCA performs better than static PCA for dynamic systems. In general, the statistics of testing set by DPCA are below the limits, which means no unexpected event is detected and the testing data can be seen as normal.

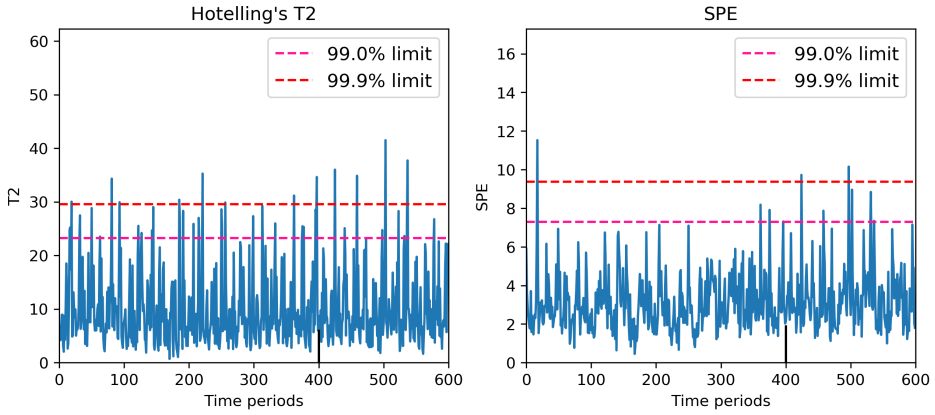
Three fault scenarios are simulated and analyzed using PCA and DPCA.

Scenario 1: The first fault scenario is the transportation delay between two echelons, which could lead to variations in the inventory levels of the affected participants. Suppose the order placed by the Retailer2 to the Warehouse at time period 454 is delayed, and the transportation time increases from 1 to 5 time periods. The simulated inventory profiles of the agents are shown in Figure 2.6a. The inventory level of the Retailer2 becomes low and backorder situation occurs due to the delay.

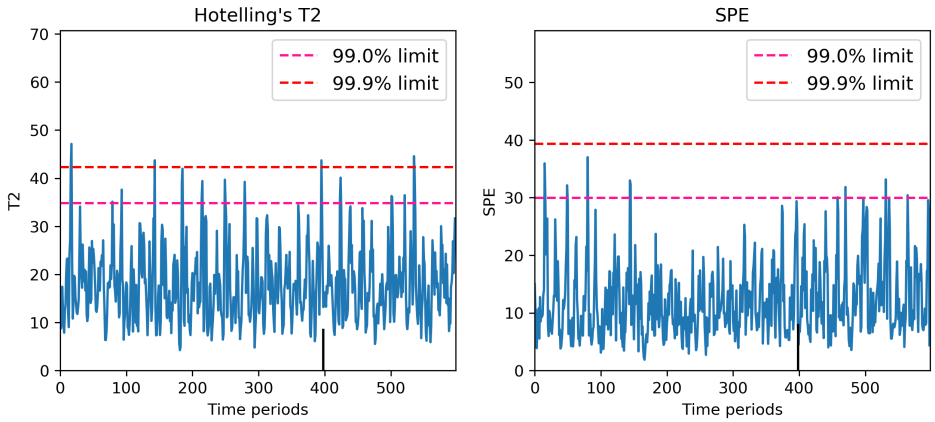
The monitoring charts using PCA and DPCA are shown in Figure 2.6b and Figure 2.6c, respectively. It can be seen that when the delay occurs, the SPE by DPCA exceeds its limits, which means a large variation from the NOC is detected. In comparison, the SPE by PCA is not able to detect the fault. In order to identify the fault-related variables, the SPE contribution plot of a fault sample detected by DPCA is shown in Figure 2.7a. It can be seen that the variables related to Retailer2's inventory contribute the most to SPE, which implies that they are most likely to be fault-related, and some unexpected event might have happened at the Retailer2. The trajectory SPE contribution plot of these two abnormal variables is given in Figure 2.7b. From the trajectory, the SPE contributions of the 2 variables increase abnormally during the period of the fault. This helps narrow the scope of the root cause



(a) Inventory levels.



(b) PCA.



(c) DPCA.

Figure 2.5. Case 1, NOC.

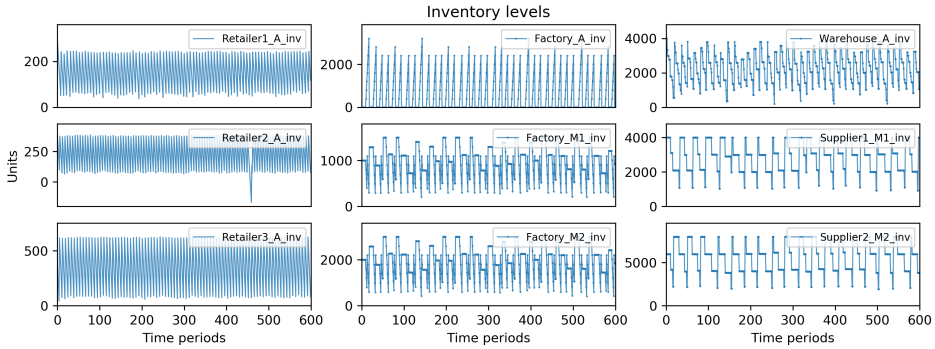
diagnosis. The SPE falls below the limits after the Retailer2 returns to NOC.

Scenario 2: The second simulated fault scenario is a low production rate (or poor yield) of the Factory. Suppose the production rate of the factory decreases from 400 to 150 units per time period suddenly within time period 451–463 due to some problem. The inventory profiles of the agents are shown in Figure 2.8a.

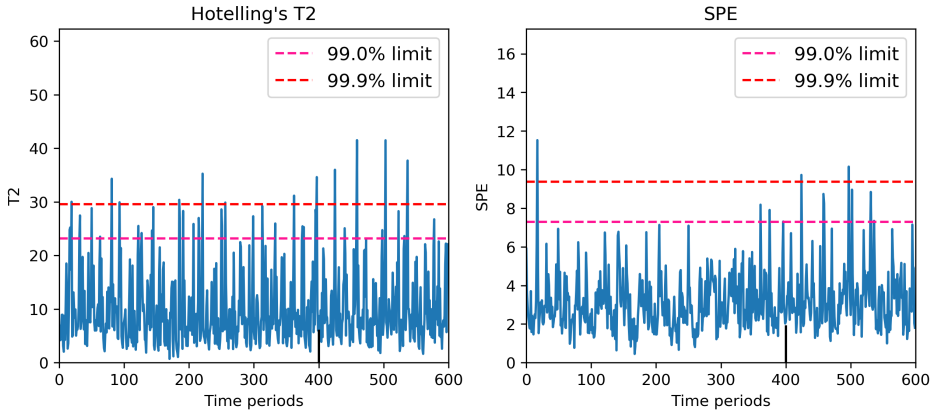
The monitoring charts using PCA and DPCA are shown in Figure 2.8b and Figure 2.8c, respectively. It can be seen that during the period with low production rate, the SPE by DPCA exceeds its limit when the inventory level of the Warehouse becomes low, and the orders from the Retailers cannot be satisfied. This means that the effect of the poor yield of the Factory on the supply chain is detected. In comparison, the SPE by PCA cannot detect the fault. For diagnosis, the SPE contribution plot of an abnormal sample and the trajectory SPE contribution plot using DPCA are shown in Figure 2.9a and Figure 2.9b, respectively. It can be seen that the contribution of the variable related to the inventory level of Warehouse is the largest. This implies that it is likely to have a large variation from the NOC, and hence indicates the low yield problem of the Factory and its effect on the operation of the Warehouse. The SPE falls below the limits after the supply chain returns to NOC. In terms of the Hotelling's T^2, PCA detects the fault while DPCA does not. This implies that PCA takes this fault as a large variation from NOC in the PCS, while DPCA takes it as a large variation in the RS.

Scenario 3: The third fault scenario is a raw material shortage of a supplier. It is simulated by a shortage occurring at the Supplier1 during time period 498–519. Figure 2.10a shows the inventory levels of the supply chain participants. Because of the shortage, the Factory cannot get raw material 'M1' from the Supplier1 and has to stop manufacturing. The Warehouse subsequently cannot get replenished, and a backorder situation occurs at the Retailer2.

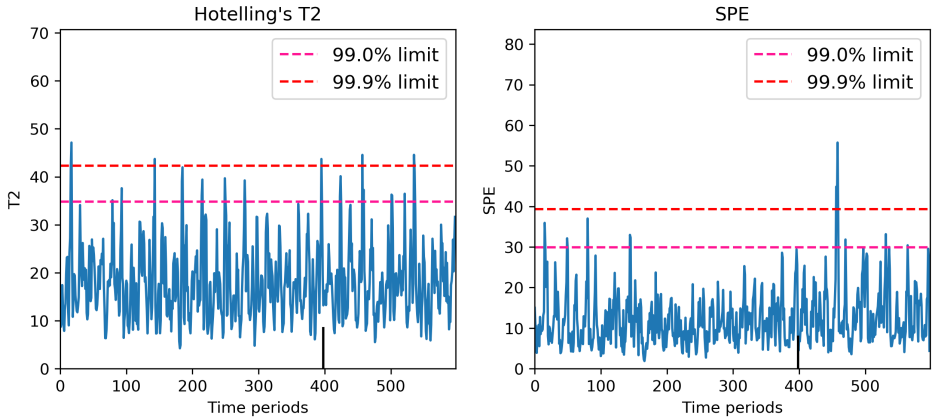
The monitoring charts by PCA and DPCA are shown in Figure 2.10b and Figure 2.10c, respectively. It can be seen that by DPCA, both the T^2 and SPE raise an alarm when the supplier's inventory level becomes low and abnormal, which is at the beginning of the stockout of the supplier. This means the abnormal behavior of the supply chain is detected soon after the stockout occurs at the supplier, before it affects downstream agents. The T^2



(a) Inventory levels.

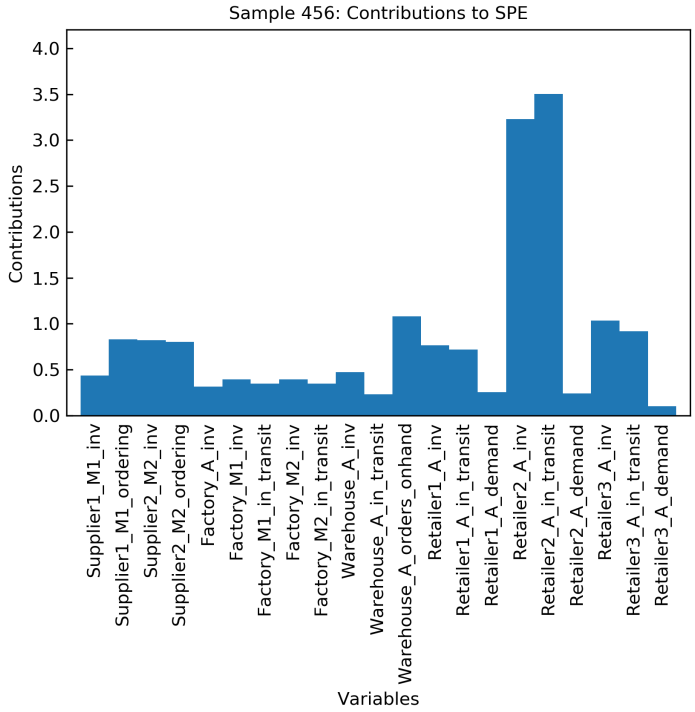


(b) PCA.

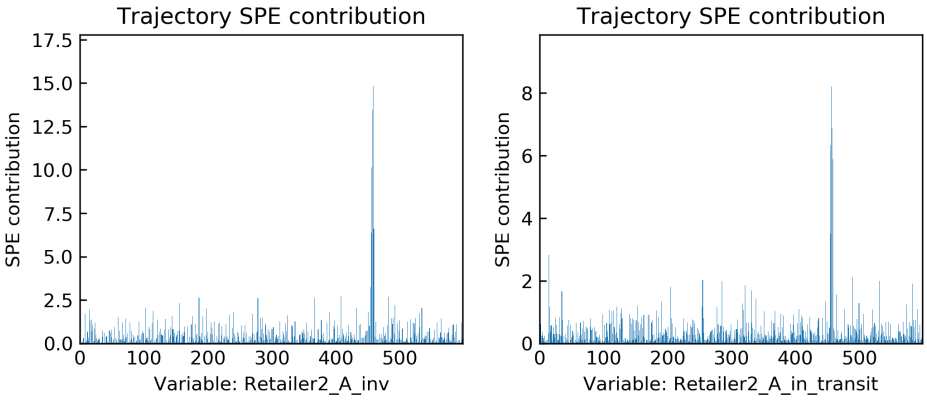


(c) DPCA.

Figure 2.6. Case 1, scenario 1: transportation delay.

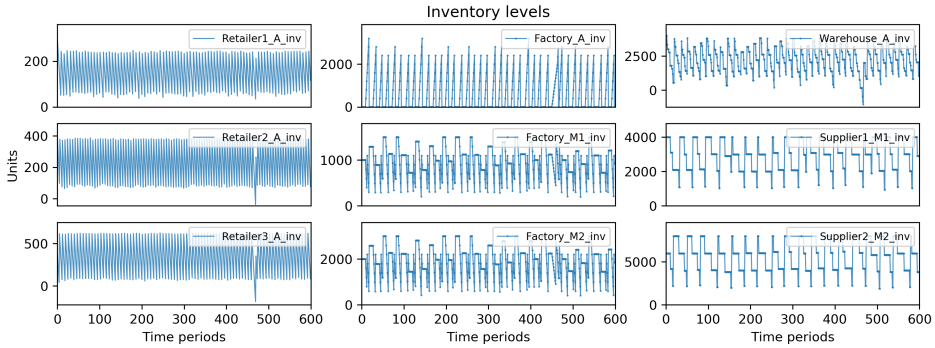


(a) Sample SPE contribution.

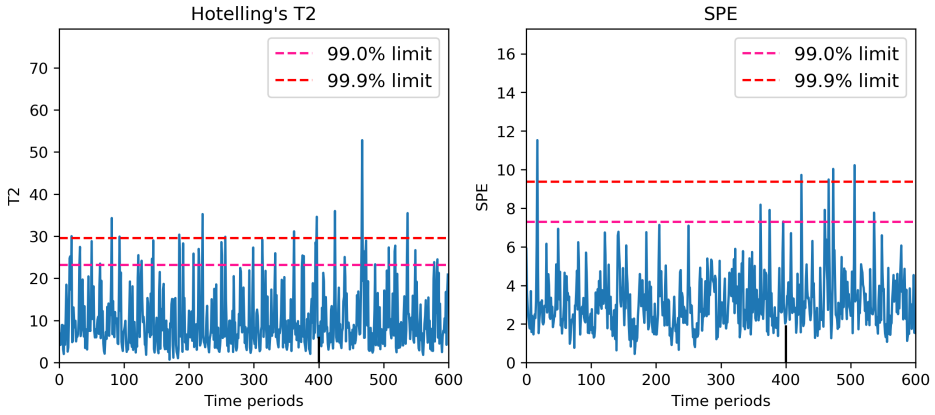


(b) Trajectory SPE contribution of fault-related variables.

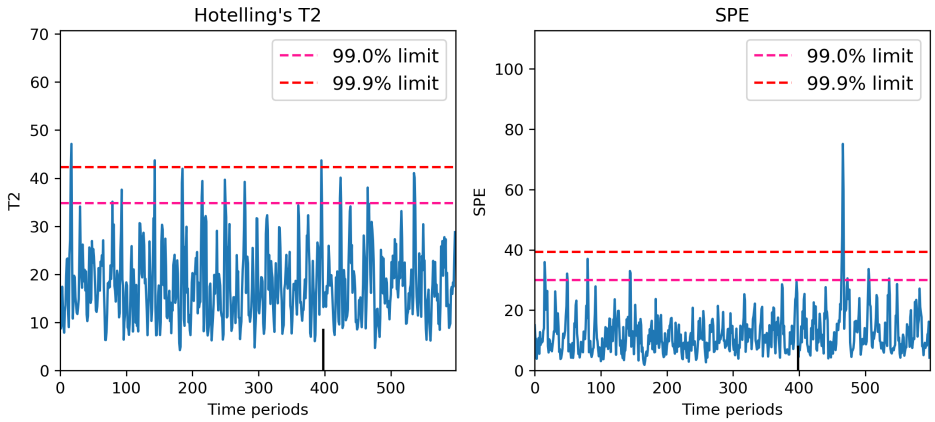
Figure 2.7. Case 1, scenario 1: DPCA, SPE contribution plots.



(a) Inventory levels.

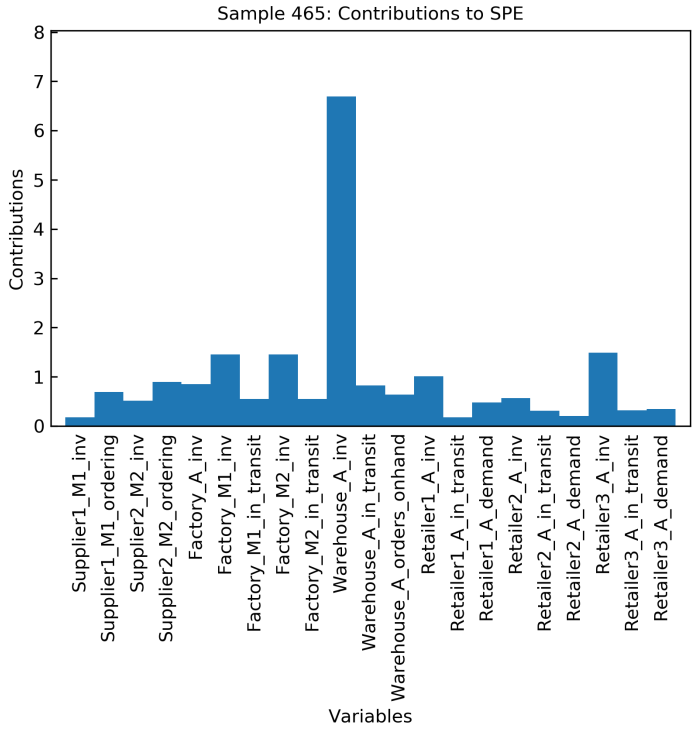


(b) PCA.

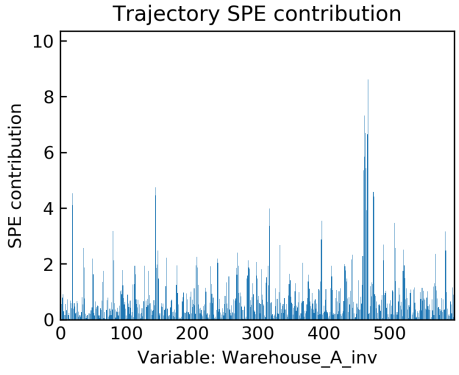


(c) DPCA.

Figure 2.8. Case 1, scenario 2: low production rate.



(a) Sample SPE contribution.



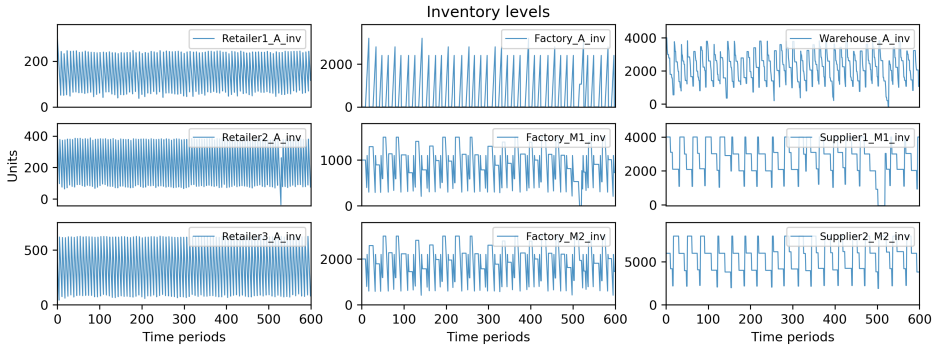
(b) Trajectory SPE contribution of fault-related variables.

Figure 2.9. Case 1, scenario 2: DPCA, SPE contribution plots.

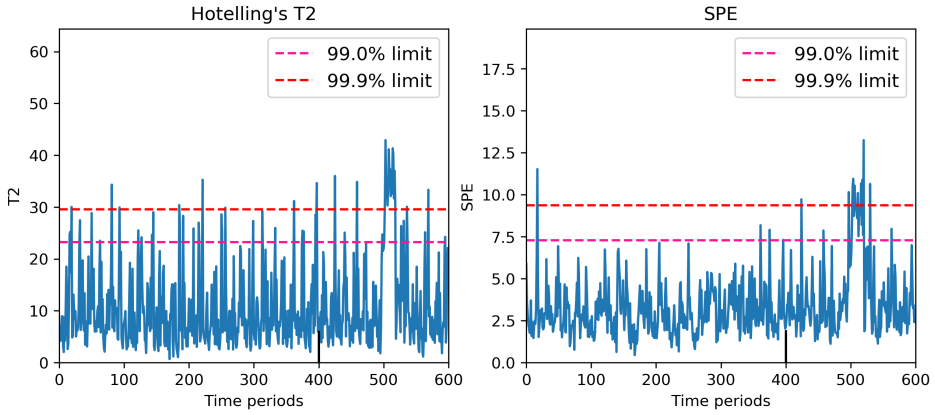
and SPE by PCA also pick up this fault. The SPE contribution plot of the abnormal sample 501 and the trajectory SPE contribution plot using DPCA are shown in Figure 2.11a and Figure 2.11b, respectively. It can be seen that the contribution of the variable related to the material in processing of Supplier1 is the largest, which implies it is likely to have a large variation from the NOC. Hence, the effect of the shortage of Supplier1 on the supply chain is detected. After the supply recovers, the supply chain returns to NOC, and the monitoring statistics fall below the limits.

It is worth mentioning that the training data does not have to be contiguous in time. The training set can be organized by removing fault data within some time periods and stacking the NOC data. A comparison of the DPCA results using different training sets while same number of PCs is given in Figure 2.12. Figure 2.12a shows the inventory levels over the simulated 1200 time periods, during which a shortage occurs twice at the Supplier1. The result when taking samples 0–300 as the training set, and samples 801–1200 as the testing set is shown in Figure 2.12b. The horizontal axes are labelled with a range of 0–700 continuously to represent these samples. It can be seen that although the abnormality is detected, the SPE of the NOC data in the testing set is generally higher than that of the training data. The reason for this is that when the supply chain recovers to NOC after the shortage, the relation between the variables and the operating status of the supply chain may change. Therefore, an increase in SPE is observed. In comparison, when combining samples 0–150 and 650–800 as the training set, the SPE of the NOC data in the testing set remain close to that of the training data, as shown in Figure 2.12c. This is because the NOC data after the shortage have been included in the training set. The comparison indicates that organizing the training set by removing fault data and stacking NOC data could help reduce overfitting to some extent. Moreover, it implies that new NOC data can be included into the training set to update the DPCA model.

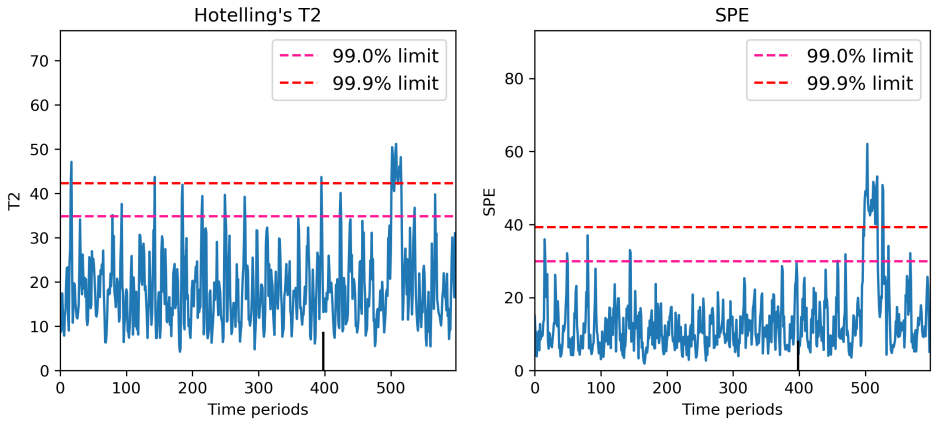
From the analysis and results shown above, the faults like transportation delay, low production rate, and supply shortage can be detected by the proposed supply chain monitoring method using DPCA. The monitoring charts raise an alarm when the abnormality occurs.



(a) Inventory levels.

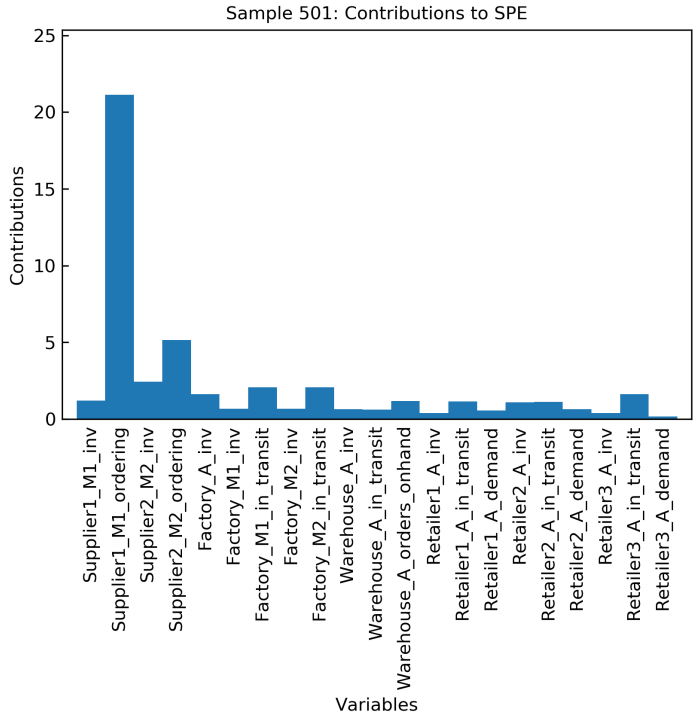


(b) PCA.

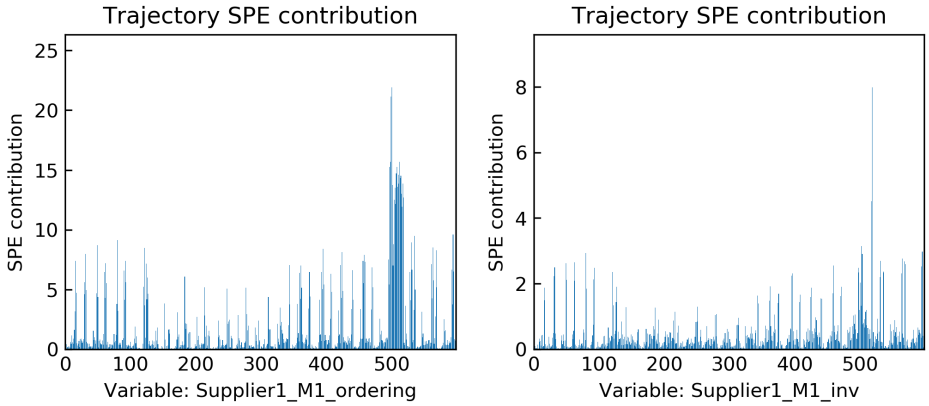


(c) DPCA.

Figure 2.10. Case 1, scenario 3: supply shortage of Supplier1.

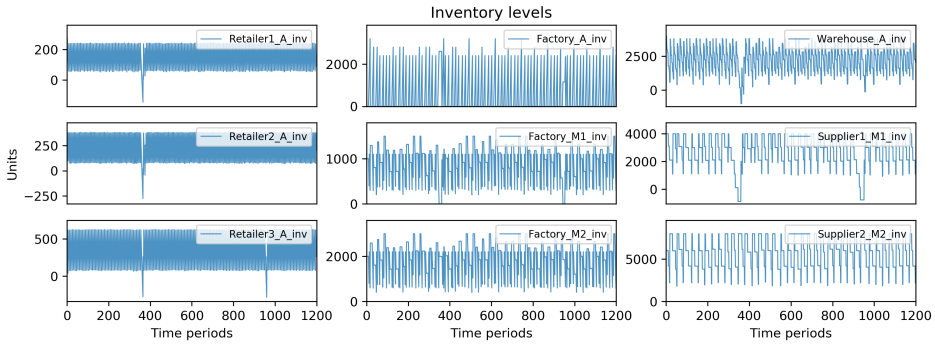


(a) Sample SPE contribution.

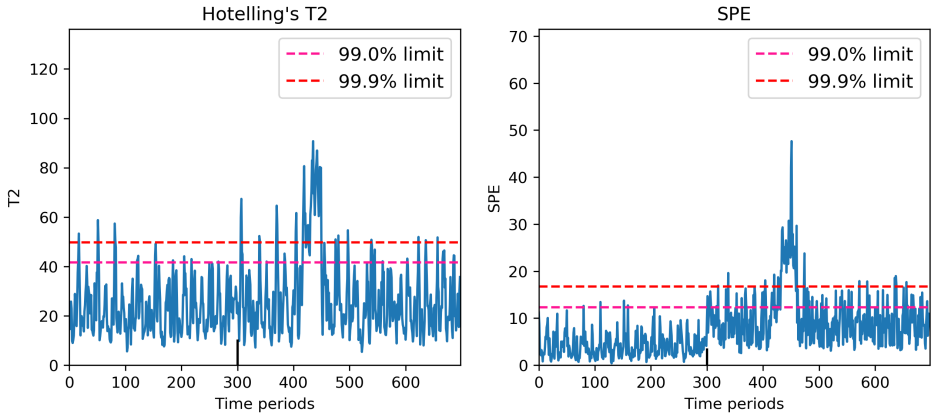


(b) Trajectory SPE contribution plot.

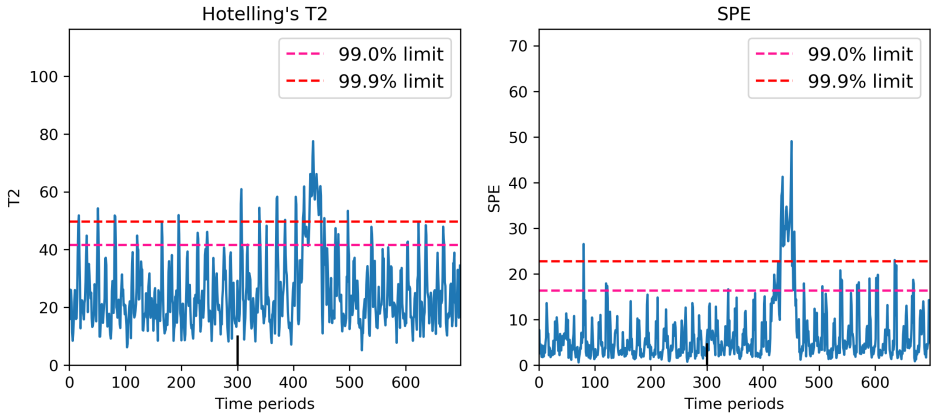
Figure 2.11. Case 1, scenario 3: DPCA, SPE contribution plots.



(a) Inventory levels.



(b) DPCA, using samples 0–300 as training set, 801–1200 as testing set.



(c) DPCA, using samples 0–150 and 650–800 as training set, 801–1200 as testing set.

Figure 2.12. Case 1: comparison of DPCA results using different training data.

2.4.2 Case study 2: A packaged liquefied gas supply chain

The second case study is based on the packaged liquefied gas supply chain investigated by Misra et al. (2019). The gas products are stored in containers (stock keeping units, SKUs). The structure of this supply chain is shown in Figure 2.13. It consists of 3 echelons: (1) the Customer locations, where the filled SKUs are consumed and then empty SKUs are generated. (2) the Warehouse, which stores both filled and empty SKUs. Filled SKUs are transported to the Customer locations to be replenished, and empty SKUs are collected from the Customer locations. The empty SKUs are transported to the Plants for refilling. (3) the Plants, where the empty SKUs from the Warehouse are refilled and then transported back.

The difference between this case study and the first case study is that the empty SKUs from downstream are transported back upstream for refilling. Therefore, there is both flow of filled SKUs from upstream to downstream, and flow of empty SKUs from downstream to upstream.

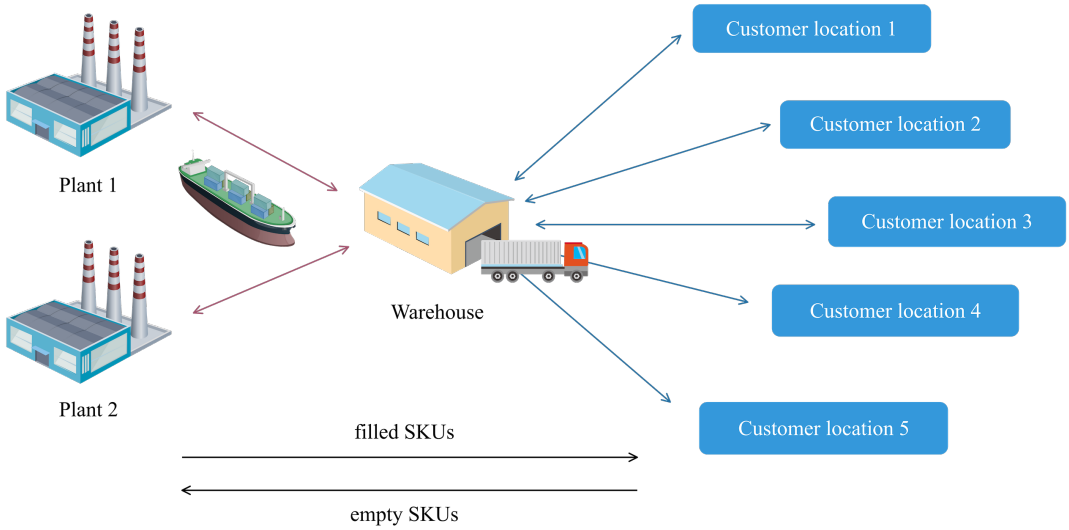


Figure 2.13. Case 2: the packaged liquefied gas supply chain from Misra et al. (2019).

In Misra et al. (2019), vendor managed inventory (VMI) is adopted to make sure the filled SKU inventory levels of the Customer locations are above the number of filled SKUs currently being used. VMI means the suppliers make the replenishment decision. In the simulation in this paper, the inventory policy is implemented in the following method: when the inventory

level of filled SKUs of a product at a Customer location is below its 're-order point', filled SKUs of this product are shipped from the Warehouse to the Customer location to replenish the stock. The number of delivered filled SKUs is exactly equal to the collected empty SKUs. Simultaneous delivery and pickup is adopted for the Customer locations, which means the trucks bring the filled SKUs to the Customer location, and collect the empty SKUs from it in the same time period of arrival, then bring them back to the Warehouse. All the delivered products originate from the Warehouse and all the collected empty containers are sent back to the Warehouse. When the inventory level of filled SKUs of the Warehouse is below its 're-order point', the empty SKUs are transported to the Plants for refilling. Ships are used for upstream transportation, while trucks are used for downstream distribution (Misra et al., 2019). Hence in this paper, the transportation times of upstream and downstream are set as 3 time periods and 1 time period, respectively.

In order to simplify the simulation model, some assumptions are made here. Two types of products A and B are transported across this supply chain, and refilled at the Plant 1 and 2, respectively; there is no product in transit before time period 0; in normal operating conditions, the transportation times are constant. The supply chain parameters used in this paper, such as the initial inventory levels and re-order points, are listed in Table 2.4. The time interval of simulation is one time period. The normal period is from 0 to 600 time periods. The numbers of the SKUs in the supply chain are assumed to be fixed, and there are no new or departing customers.

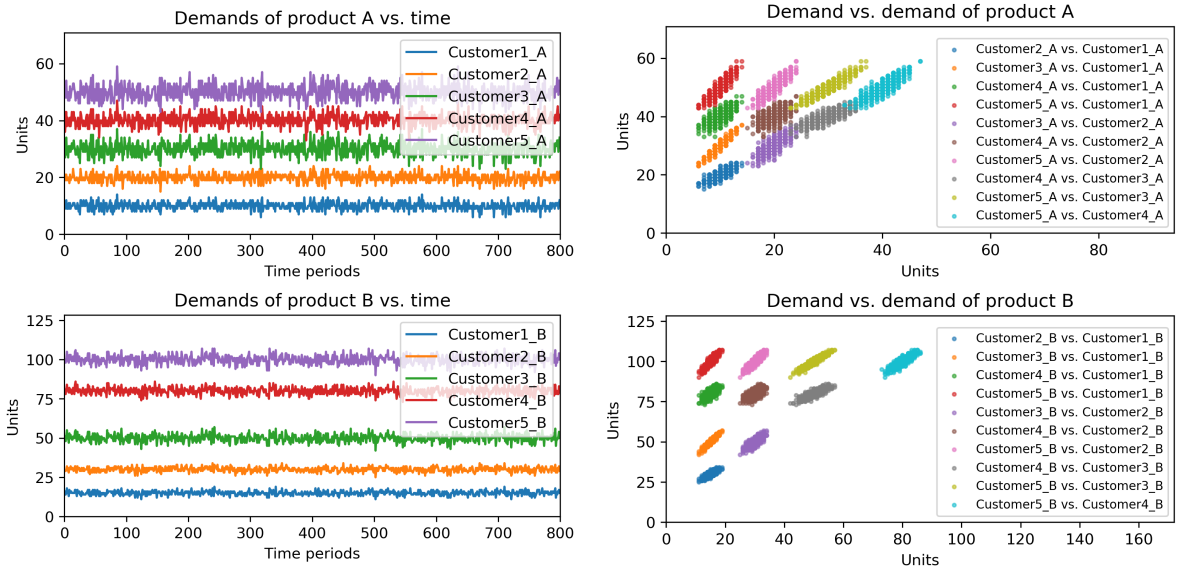
Table 2.4. Parameters of the supply chain.

Participant	demand (mean)		initial filled SKUs		initial empty SKUs		re-order points	
	A	B	A	B	A	B	A	B
Customer1	10	15	100	150	0	0	50	80
Customer2	20	30	200	350	0	0	100	200
Customer3	30	50	300	450	0	0	180	250
Customer4	40	80	400	800	0	0	240	500
Customer5	50	100	600	1500	0	0	400	800
Warehouse	–	–	4000	8000	2000	3500	2000	4000

The demands for A and B at the 5 Customer locations are shown in Figure 2.14. They are generated from multivariate Gaussian distributions, respectively, and then rounded into

integers. The demand means the amount of filled SKUs consumed at a Customer location in each time period, which is equal to the amount of empty SKUs generated. The simulated inventory profiles of the agents for the two products are shown in Figure 2.15a and Figure 2.15b, respectively. The data of the Warehouse and Customer locations are collected for analysis. There are a total of 60 variables collected, including the demands for A and B at each Customer location (10 variables), inventory levels of filled and empty SKUs for A and B at each agent (total of 24 variables), and the amount of filled and empty SKUs for A and B in transit and in refilling at the Plants (total of 26 variables).

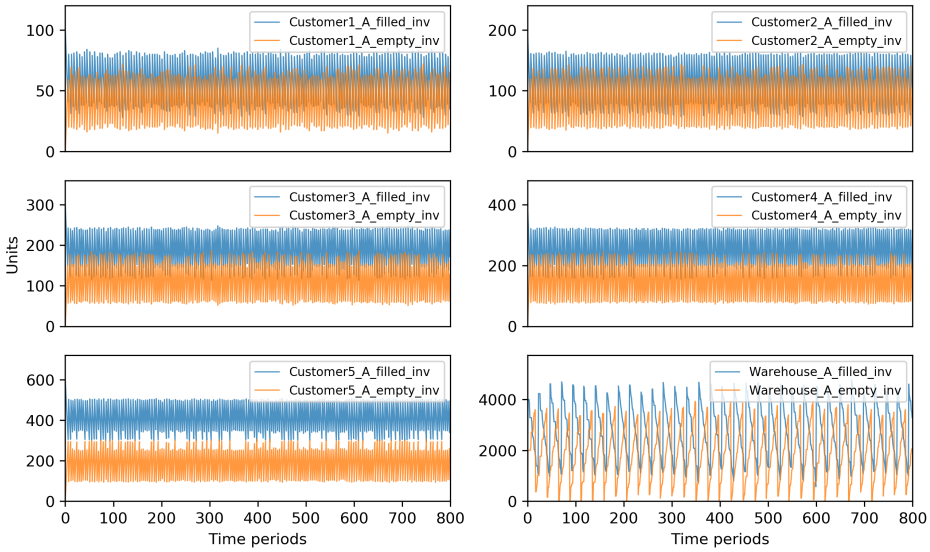
The supply chain operation over 800 time periods is simulated and analyzed. Each time period can be seen as 1 day. The NOC data during time period 10–600 is taken as the training set. Using DPCA with 2 time lags, the raw data is augmented to $60 \times 3 = 180$ variables, and 50 PCs are extracted to achieve a R^2 value of 95%. The data between period 601–800 is taken as the testing set, which are also normal data. The monitoring charts for the D- and Q- statistics using DPCA are shown in Figure 2.16. The 99% and 99.9% confidence limits are estimated from the training data and plotted as dashed lines. It can be seen that in NOC, the two statistics of testing set are within the confidence limits. This suggests no unexpected event occurs in the supply chain and the testing data can be seen as normal.



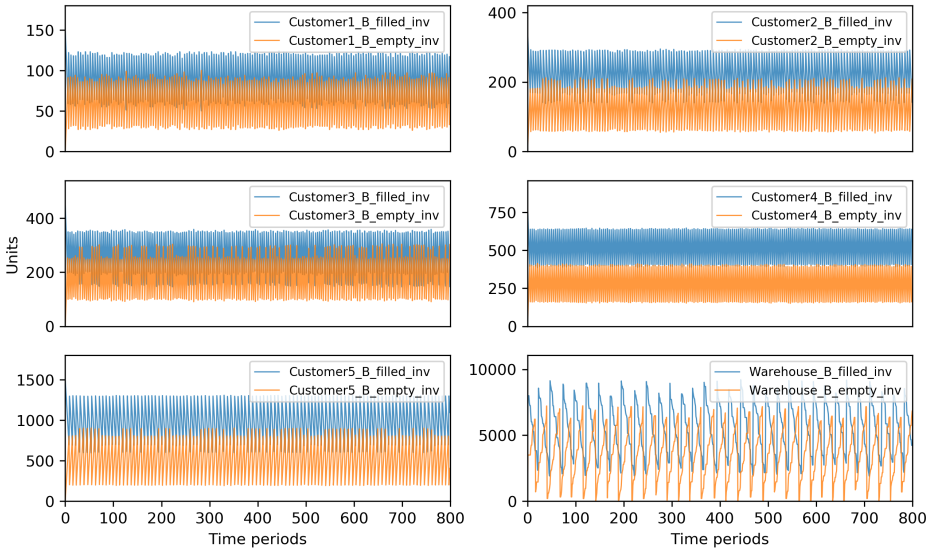
(a) Demands versus time.

(b) Demand versus demand for Customer pairs.

Figure 2.14. Case 2: demands.



(a) Product A.



(b) Product B.

Figure 2.15. Case 2, NOC: inventory levels.

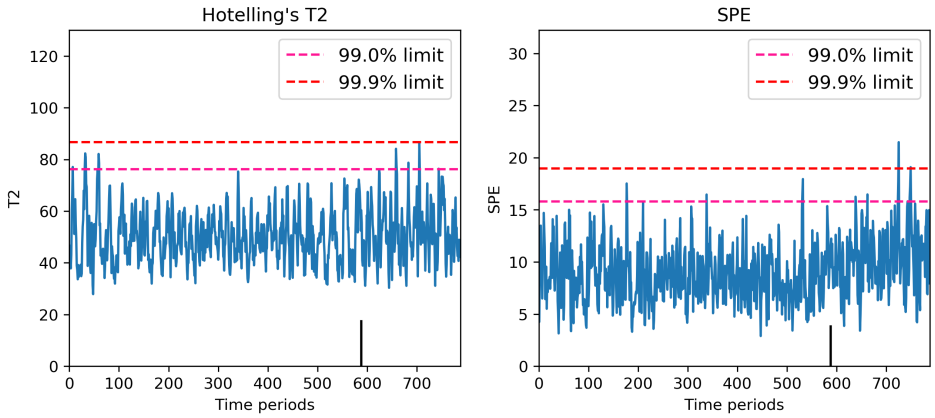
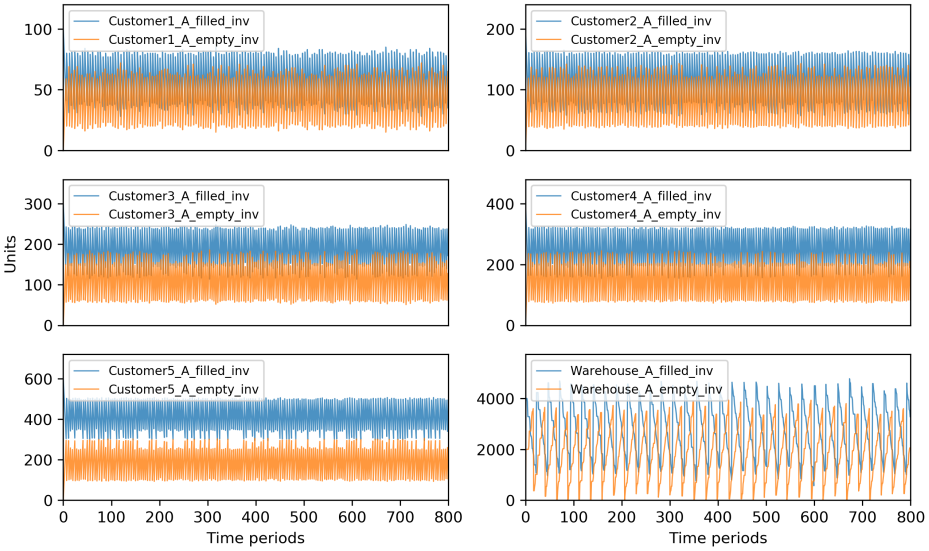


Figure 2.16. Case 2, NOC: DPCA

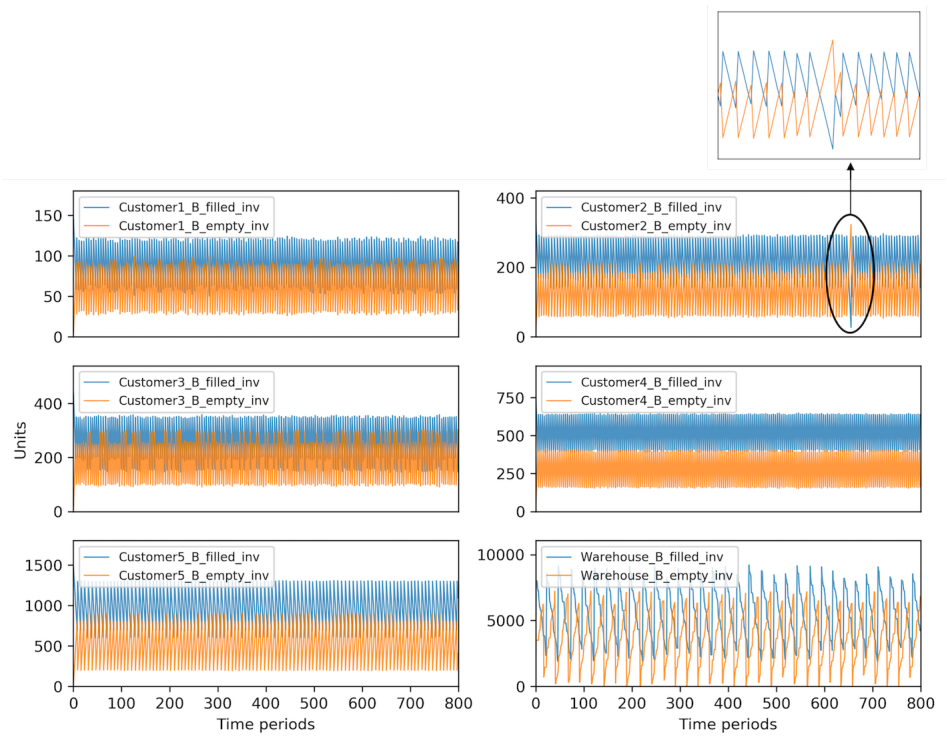
The transportation delay is stated as a supply chain risk by Chopra and Sodhi(Chopra and Sodhi, 2004). In this case study, two fault scenarios are designed in the simulation and analyzed: the transportation delay under Gaussian demands and seasonal demands, respectively. In the analysis, DPCA is focused on.

Scenario 1: The first scenario is the transportation delay under multivariate Gaussian demands. Suppose a problem occurs in the transportation link between the Warehouse and the Customer location 2, and the transportation time of the shipment of product B to the Customer location 2 at time period 650 increases from 1 day to 5 time periods. The inventory profiles of the agents are shown in Figure 2.17a and Figure 2.17b. It can be seen from Figure 2.17b that when the delay occurs, the Customer location 2's inventory level of filled SKUs of product B becomes low and the empty SKUs pile up.

The monitoring charts using DPCA are shown in Figure 2.18. It can be seen that when the delay occurs, the SPE by DPCA exceeds the 99.9% confidence limit by a wide margin, which means there is a large variation from the NOC. In order to identify the fault-related variables, the SPE contribution plots by DPCA of a sample above the limit is presented in Figure 2.19. From this plot, the variables related to the Customer location 2's inventory of product B contribute the largest to SPE, which implies they are most likely to be fault-related, and some abnormal event might happened at Customer location 2. This helps in the diagnosis of the root cause.



(a) Product A.



(b) Product B.

Figure 2.17. Case 2, scenario 1: inventory levels.

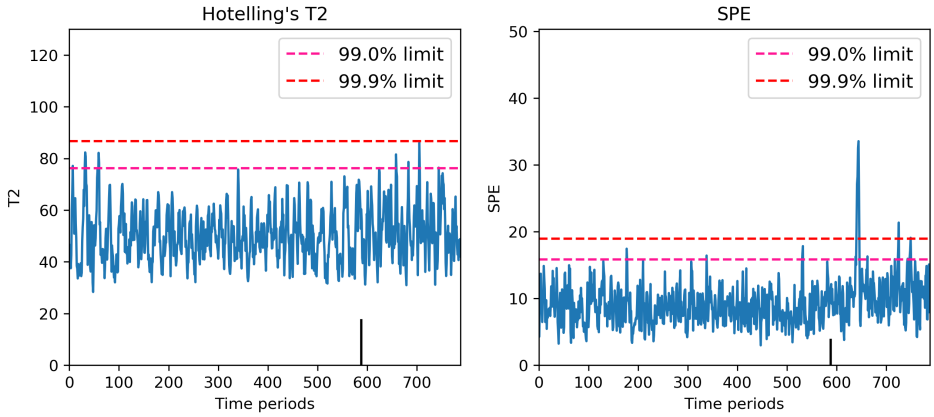


Figure 2.18. Case 2, scenario 1: DPCA.

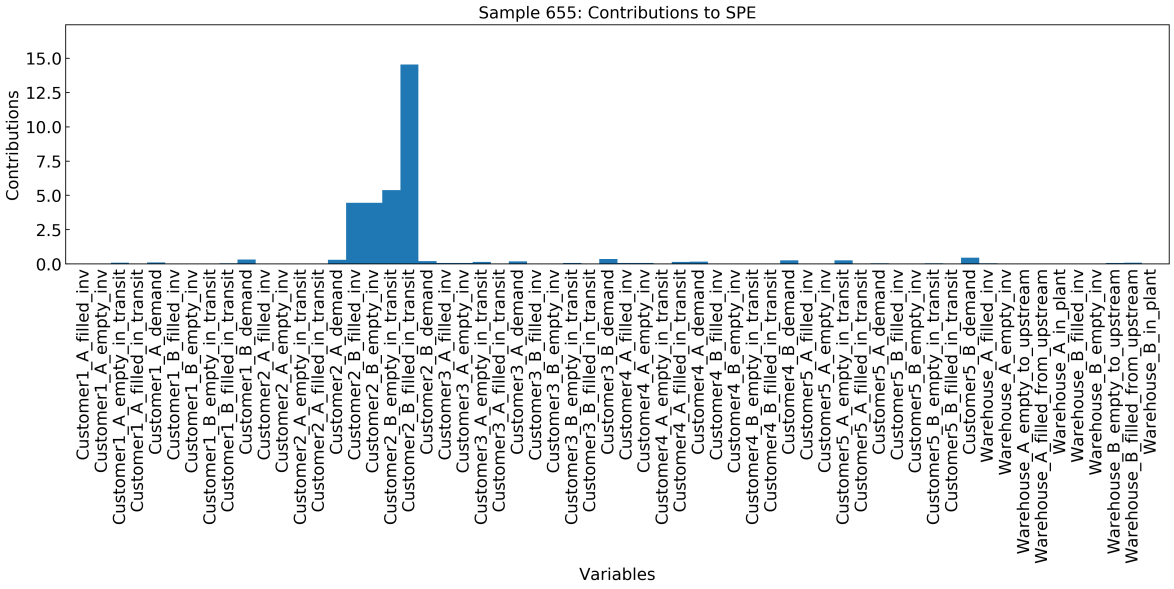


Figure 2.19. Case 2, scenario 1: DPCA, sample SPE contribution.

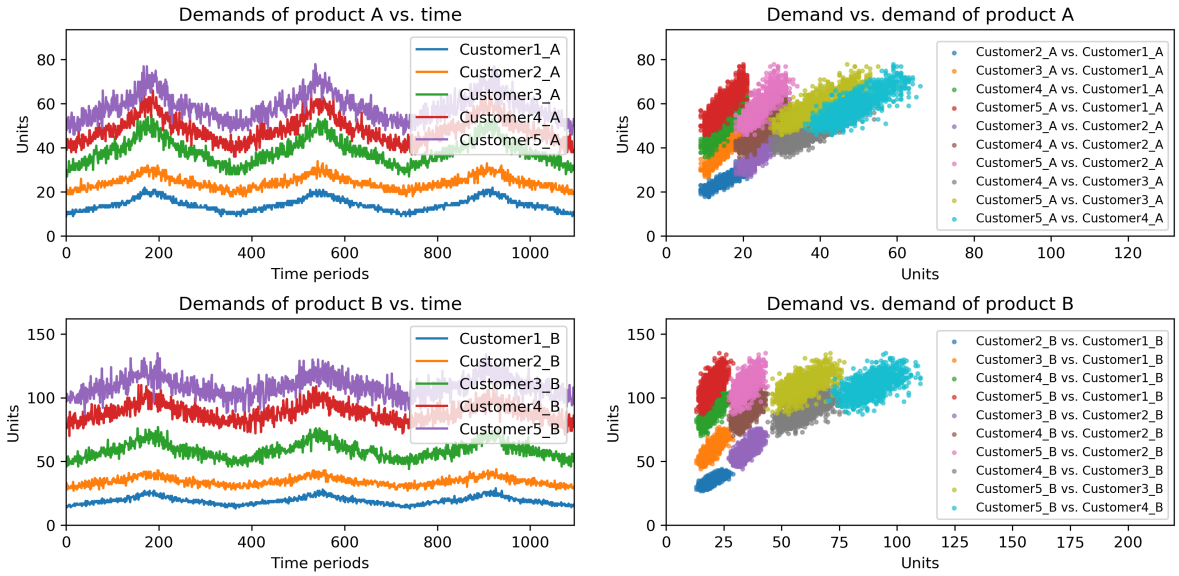
Scenario 2: The other fault scenario simulated is the transportation delay under seasonal demands, which could cause a variation in the inventory level of the affected Customer location. The seasonal demands at the 5 Customer locations are shown in Figure 2.20. For this scenario, a total of 1095 time periods are simulated. The Customer location 2 experiences a transportation delay for the shipment of product B from the Warehouse at time period 784. The shipment is delayed to 5 time periods. The inventory profiles of the agents are shown in Figure 2.21. It can be seen from Figure 2.21b that at Customer location 2, the inventory level of filled SKUs of product B is low than usual when the delay occurs, and the empty SKUs pile up.

The NOC data during time period 10–730 are taken as the training set, and the data during time period 731–1095 are taken as the testing set. 45 PCs are retained to achieve a R^2 value of 95%. The monitoring charts using DPCA are shown in Figure 2.22. It can be seen that during the period when the delay occurs, the SPE greatly exceeds its limit. This means the effect of the delay on the inventory levels of the Customer location 2 is successfully detected. For diagnosis, the SPE contribution plot of an abnormal sample is shown in Figure 2.23. It can be seen that the contributions of variables related to the Customer location 2's inventory of product B are the largest. This implies the inventory of the Customer location 2 have a large variation from the NOC, and hence the fault is identified.

From the analysis and results shown above, the effect of transportation delay on the supply chain, under Gaussian and seasonal demands, is detected by the proposed supply chain monitoring method using DPCA.

2.5 Conclusion

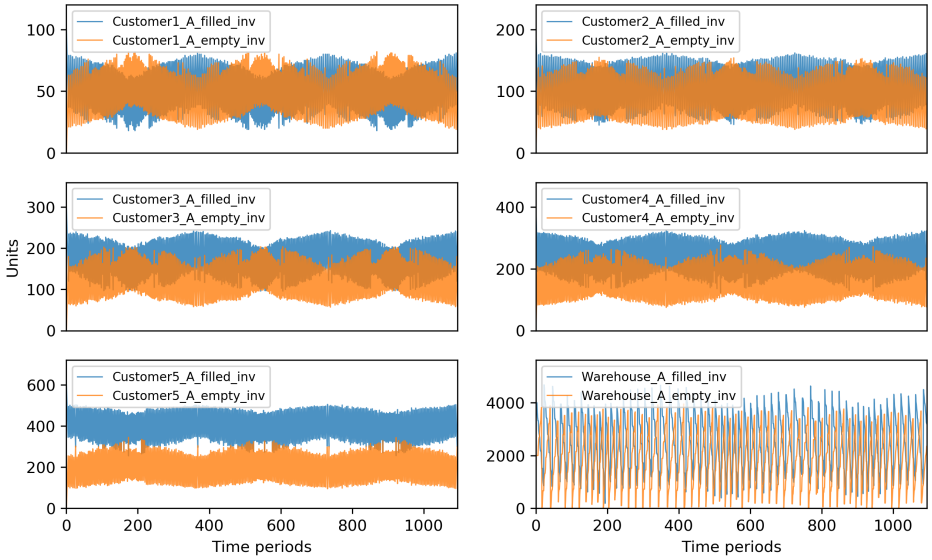
In this paper, PCA and dynamic PCA are applied for the fault detection and diagnosis of supply chain systems. In order to monitor the supply chain, data such as inventory levels, market demands, and material in transit are collected. PCA and DPCA are employed to model the NOC of the supply chain, and the Hotelling's T^2 and SPE are used to detect abnormal behavior of the supply chain. Contribution plots are adopted to identify the fault-related



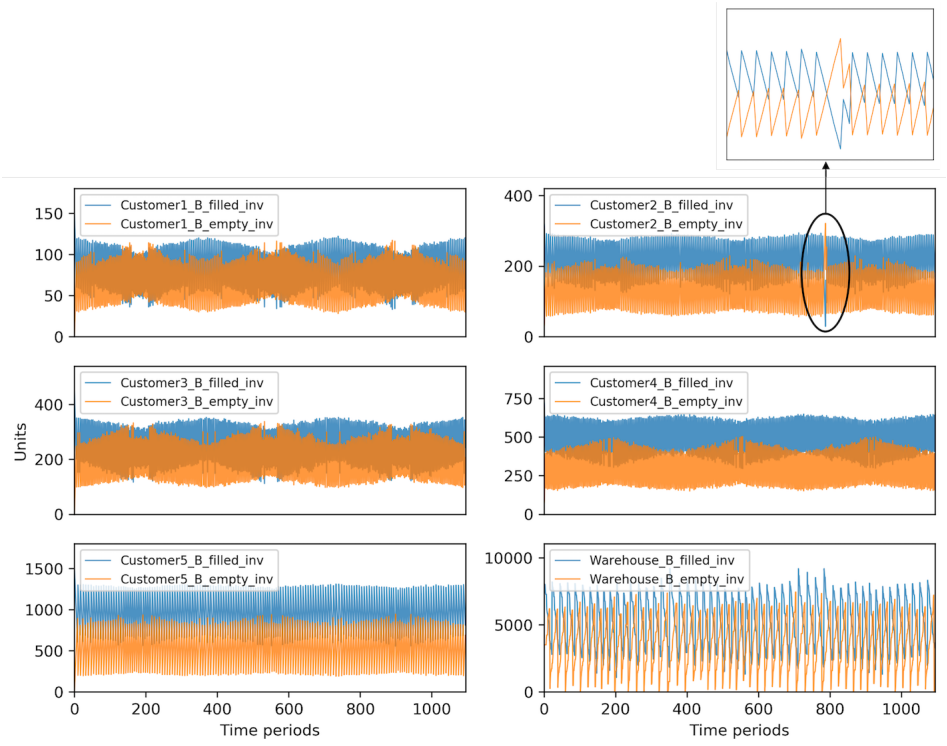
(a) Demands versus time. (b) Demand versus demand for Customer pairs.

Figure 2.20. Case 2, scenario 2: seasonal demands at the 5 Customer locations.

variables when at least one index exceeds the limits. The proposed supply chain monitoring method is validated on two case studies, one of which is a 4-echelon supply chain with single product, and the other is a two-product supply chain with materials transported not only from upstream to downstream but also in the opposite direction. A Python-based supply chain simulator is developed to generate supply chain data. Different scenarios are simulated and analyzed. The results show that the SPE by DPCA is more reliable than the other fault detection indices considered in this paper, while that by PCA is not sensitive enough. Abnormal behavior of the supply chain, such as transportation delay, low production rate and supply shortage, can be successfully detected by DPCA. The proposed method applies to non-contiguous data and seasonal market demands. Moreover, the contribution plots can help interpret the abnormality and identify the fault-related variables. Future work would consider investigating the use of other data-driven methods, such the CVA and FDA, in fault detection and diagnosis of supply chains.



(a) Product A.



(b) Product B.

Figure 2.21. Case 2, scenario 2: inventory levels.

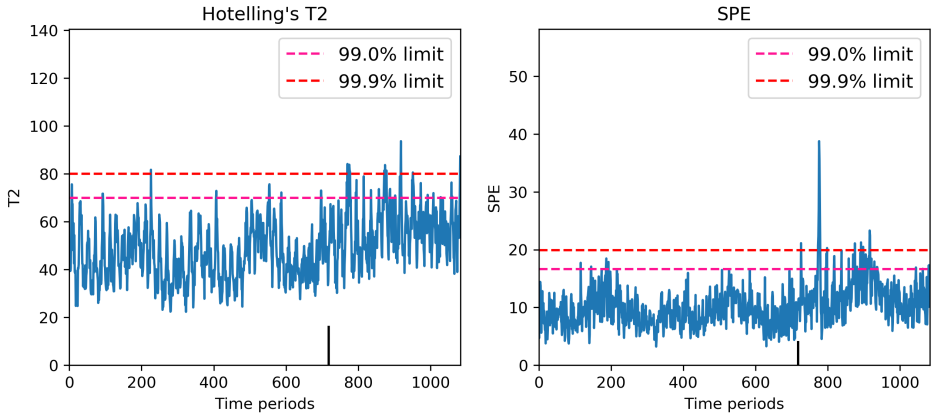


Figure 2.22. Case 2, scenario 2: DPCA.

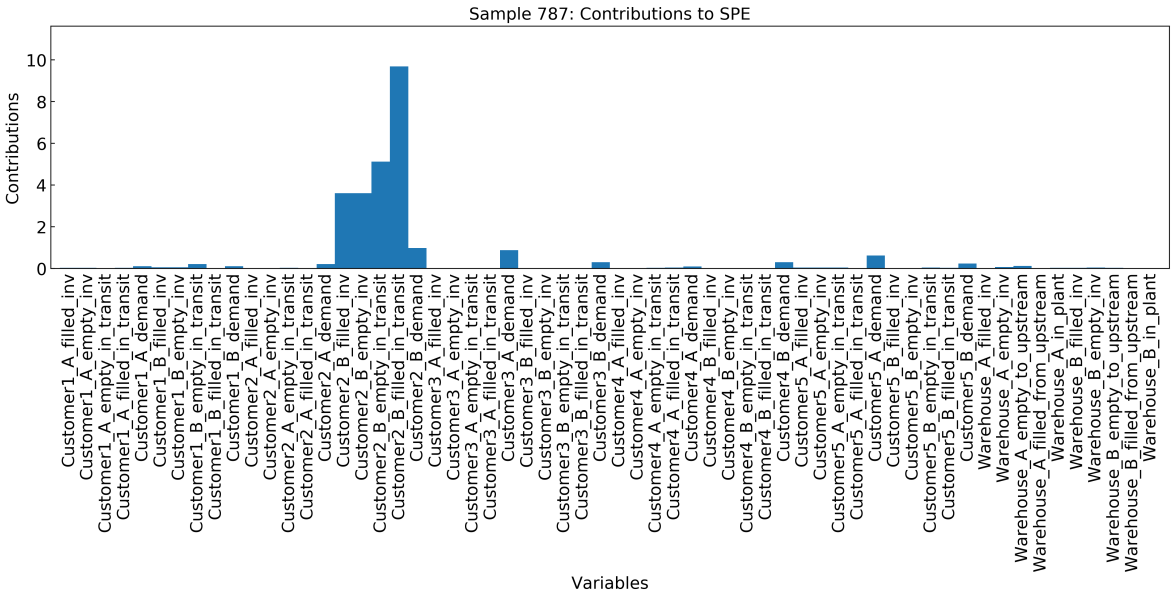


Figure 2.23. Case 2, scenario 2: DPCA, sample SPE contribution.

References

Alain Yee, L. C., B. Li, E. W. T. Ngai, E. Ch'ng, and L. Filbert (2016). "Predicting online product sales via online reviews, sentiments, and promotion strategies". In: *International Journal of Operations & Production Management* 36.4, pp. 358–383.

Alcala, C. F. and S. J. Qin (2009). "Reconstruction-based contribution for process monitoring". In: *Automatica* 45.7, pp. 1593–1600.

Axsäter, S. (2015). *Inventory Control*. 3rd. Springer, Cham.

Carvalho, H., A. P. Barroso, V. H. Machado, S. Azevedo, and V. Cruz-Machado (2012). "Supply chain redesign for resilience using simulation". In: *Computers & Industrial Engineering* 62.1, pp. 329–341.

Chiang, D. M.-H., C.-P. Lin, and M.-C. Chen (2011). "The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres". In: *Enterprise Information Systems* 5.2, pp. 219–234.

Chiang, L., E. Russell, and R. Braatz (2001). *Fault Detection and Diagnosis in Industrial Systems*. Springer-Verlag, London.

Chiang, L. H. and R. D. Braatz (2003). "Process monitoring using causal map and multivariate statistics: fault detection and identification". In: *Chemometrics and Intelligent Laboratory Systems* 65.2, pp. 159–178.

Chiang, L. H., E. L. Russell, and R. D. Braatz (2000). "Fault diagnosis in chemical processes using Fisher discriminant analysis, discriminant partial least squares, and principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 50.2, pp. 243–252.

Chopra, S. and M. S. Sodhi (2004). "Managing risk to avoid supply-chain breakdown." In: *MIT Sloan Management Review* 46.1, pp. 53–62.

— (2014). "Reducing the risk of supply chain disruptions". In: *MIT Sloan Management Review* 55.3, pp. 73–80.

Dong, Y. and S. J. Qin (2018). "A novel dynamic PCA algorithm for dynamic data modeling and process monitoring". In: *Journal of Process Control* 67, pp. 1–11.

Gao, J., C. Ning, and F. You (2019). "Data-driven distributionally robust optimization of shale gas supply chains under uncertainty". In: *AIChE Journal* 65.3, pp. 947–963.

García-Muñoz, S., T. Kourti, J. F. MacGregor, A. G. Mateos, and G. Murphy (2003). "Troubleshooting of an industrial batch process using multivariate methods". In: *Industrial & Engineering Chemistry Research* 42.15, pp. 3592–3601.

Ge, Z., Z. Song, S. X. Ding, and B. Huang (2017). "Data mining and analytics in the process industry: The role of machine learning". In: *IEEE Access* 5, pp. 20590–20616.

Ge, Z. (2017). "Review on data-driven modeling and monitoring for plant-wide industrial processes". In: *Chemometrics and Intelligent Laboratory Systems* 171, pp. 16–25.

Grossmann, I. E. (2005). "Enterprise-wide optimization: A new frontier in process systems engineering". In: *AIChE Journal* 51.7, pp. 1846–1857.

He, Q. P., S. J. Qin, and J. Wang (2005). "A new fault diagnosis method using fault directions in Fisher discriminant analysis". In: *AIChE Journal* 51.2, pp. 555–571.

How, B. S. and H. L. Lam (2018). "Sustainability evaluation for biomass supply chain synthesis: Novel principal component analysis (PCA) aided optimisation approach". In: *Journal of Cleaner Production* 189, pp. 941–961.

Huang, J. and X. Yan (2015). "Dynamic process fault detection and diagnosis based on dynamic principal component analysis, dynamic independent component analysis and Bayesian inference". In: *Chemometrics and Intelligent Laboratory Systems* 148, pp. 115–127.

Ivanov, D. (2017). *Operations and supply chain simulation with AnyLogic: Decision-oriented introductory notes for master students.* 2nd. E-Textbook, Berlin School of Economics and Law.

Jain, R., A. R. Singh, H. C. Yadav, and P. K. Mishra (Feb. 2014). "Using data mining synergies for evaluating criteria at pre-qualification stage of supplier selection". In: *Journal of Intelligent Manufacturing* 25.1, pp. 165–175.

Jiang, B., D. Huang, X. Zhu, F. Yang, and R. D. Braatz (2015a). "Canonical variate analysis-based contributions for fault identification". In: *Journal of Process Control* 26, pp. 17–25.

Jiang, B., X. Zhu, D. Huang, J. A. Paulson, and R. D. Braatz (2015b). "A combined canonical variate analysis and Fisher discriminant analysis (CVA-FDA) approach for fault diagnosis". In: *Computers & Chemical Engineering* 77, pp. 1–9.

Kodamana, H., R. Raveendran, and B. Huang (Mar. 2019). "Mixtures of probabilistic PCA with common structure latent bases for process monitoring". In: *IEEE Transactions on Control Systems Technology* 27.2, pp. 838–846.

Kourti, T. and J. F. MacGregor (1995). "Process analysis, monitoring and diagnosis, using multivariate projection methods". In: *Chemometrics and Intelligent Laboratory Systems* 28.1, pp. 3–21.

— (1996). "Multivariate SPC methods for process and product monitoring". In: *Journal of Quality Technology* 28.4, pp. 409–428.

Kresta, J. V., J. F. Macgregor, and T. E. Marlin (1991). "Multivariate statistical monitoring of process operating performance". In: *The Canadian Journal of Chemical Engineering* 69.1, pp. 35–47.

Ku, W., R. H. Storer, and C. Georgakis (1995). "Disturbance detection and isolation by dynamic principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 30.1, pp. 179–196.

Lee, J.-M., C. Yoo, S. W. Choi, P. A. Vanrolleghem, and I.-B. Lee (2004). "Nonlinear process monitoring using kernel principal component analysis". In: *Chemical Engineering Science* 59.1, pp. 223–234.

Lee, Y. H., M. K. Cho, S. J. Kim, and Y. B. Kim (2002). "Supply chain simulation with discrete–continuous combined modeling". In: *Computers and Industrial Engineering* 43.1, pp. 375–392.

Lei, N. and S. K. Moon (2015). "A Decision Support System for market-driven product positioning and design". In: *Decision Support Systems* 69, pp. 82–91.

Li, G., S. J. Qin, and T. Chai (June 2014a). "Multi-directional reconstruction based contributions for root-cause diagnosis of dynamic processes". In: *2014 American Control Conference*, pp. 3500–3505.

Li, G., S. J. Qin, and T. Yuan (2016). "Data-driven root cause diagnosis of faults in process industries". In: *Chemometrics and Intelligent Laboratory Systems* 159, pp. 1–11.

Li, H., D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur (2014b). "Improving rail network velocity: A machine learning approach to predictive maintenance". In: *Transportation Research Part C: Emerging Technologies* 45, pp. 17–26.

Li, L., X. Su, Y. Wang, Y. Lin, Z. Li, and Y. Li (2015). "Robust causal dependence mining in big data network and its application to traffic flow predictions". In: *Transportation Research Part C: Emerging Technologies* 58, pp. 292–307.

Li, W. and S. Qin (2001). "Consistent dynamic PCA based on errors-in-variables subspace identification". In: *Journal of Process Control* 11.6, pp. 661–678.

Li, X. and T. E. Marlin (2009). "Robust supply chain performance via Model Predictive Control". In: *Computers & Chemical Engineering* 33.12, pp. 2134–2143.

Ma, J., M. Kwak, and H. M. Kim (2014). "Demand trend mining for predictive life cycle design". In: *Journal of Cleaner Production* 68, pp. 189–199.

MacGregor, J. F. and A. Cinar (2012). "Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods". In: *Computers and Chemical Engineering* 47, pp. 111–120.

MacGregor, J. F. and T. Kourti (1995). "Statistical process control of multivariate processes". In: *Control Engineering Practice* 3.3, pp. 403–414.

Mastragostino, R., S. Patel, and C. L. Swartz (2014). "Robust decision making for hybrid process supply chain systems via model predictive control". In: *Computers & Chemical Engineering* 62, pp. 37–55.

Mele, F. D., E. Musulin, and L. Puigjaner (2005). "Supply chain monitoring: a statistical approach". In: *European Symposium on Computer Aided Process Engineering-15*. Ed. by L. Puigjaner and A. Espuna. Vol. 20B. Computer Aided Chemical Engineering. Elsevier, pp. 1375–1380.

Miller, P., R. E. Swanson, and C. E. Heckler (1998). "Contribution plots: a missing link in multivariate quality control". In: *Applied Mathematics and Computer Science* 8.4, pp. 775–792.

Mishra, D., A. Gunasekaran, T. Papadopoulos, and S. J. Childe (Nov. 2018). "Big Data and supply chain management: a review and bibliometric analysis". In: *Annals of Operations Research* 270.1, pp. 313–336.

Misra, S., M. Kapadi, R. D. Gudi, and D. Saxena (2019). "Resource optimization and inventory routing of the packaged liquefied gas supply chain". In: *Industrial & Engineering Chemistry Research* 58.18, pp. 7579–7592.

Mori, J., Y. Kajikawa, H. Kashima, and I. Sakata (2012). "Machine learning approach for finding business partners and building reciprocal relationships". In: *Expert Systems with Applications* 39.12, pp. 10402–10407.

Negiz, A. and A. Çinar (1997). "Statistical monitoring of multivariable dynamic processes with state-space models". In: *AIChE Journal* 43.8, pp. 2002–2020.

Nguyen, T., L. Zhou, V. Spiegler, P. Ieromonachou, and Y. Lin (2018). "Big data analytics in supply chain management: A state-of-the-art literature review". In: *Computers and Operations Research* 98, pp. 254–264.

Ning, C. and F. You (2017). "Data-driven adaptive nested robust optimization: General modeling framework and efficient computational algorithm for decision making under uncertainty". In: *AIChE Journal* 63.9, pp. 3790–3817.

— (2018a). "Data-driven decision making under uncertainty integrating robust optimization with principal component analysis and kernel smoothing methods". In: *Computers and Chemical Engineering* 112, pp. 190–210.

— (2018b). "Data-driven stochastic robust optimization: General computational framework and algorithm leveraging machine learning for optimization under uncertainty in the big data era". In: *Computers & Chemical Engineering* 111, pp. 115–133.

— (2019). "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming". In: *Computers & Chemical Engineering* 125, pp. 434–448.

Nomikos, P. and J. F. MacGregor (1995). "Multivariate SPC charts for monitoring batch processes". In: *Technometrics* 37.1, pp. 41–59.

Papageorgiou, L. G. (2009). "Supply chain optimisation for the process industries: Advances and opportunities". In: *Computers & Chemical Engineering* 33.12, pp. 1931–1938.

Patel, S. and C. L. Swartz (2019). "Supply chain design with time-limited transportation contracts". In: *Computers & Chemical Engineering*, p. 106579.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pozo, C., R. Ruíz-Femenia, J. Caballero, G. Guillén-Gosálbez, and L. Jiménez (2012). "On the use of Principal Component Analysis for reducing the number of environmental objectives in multi-objective optimization: Application to the design of chemical supply chains". In: *Chemical Engineering Science* 69.1, pp. 146–158.

Pundoor, G. and J. W. Herrmann (2006). "A hierarchical approach to supply chain simulation modelling using the Supply Chain Operations Reference model". In: *International Journal of Simulation and Process Modelling* 2.3-4, pp. 124–132.

Qin, S. J. (2003). "Statistical process monitoring: basics and beyond". In: *Journal of Chemometrics* 17.8-9, pp. 480–502.

— (2012). "Survey on data-driven industrial process monitoring and diagnosis". In: *Annual Reviews in Control* 36.2, pp. 220–234.

Qin, S. J. and L. H. Chiang (2019). "Advances and opportunities in machine learning for process data analytics". In: *Computers and Chemical Engineering* 126, pp. 465–473.

Russell, E. L., L. H. Chiang, and R. D. Braatz (2000). "Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 51.1, pp. 81–93.

Salehan, M. and D. J. Kim (2016). "Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics". In: *Decision Support Systems* 81, pp. 30–40.

Shah, N. (2005). "Process industry supply chains: Advances and challenges". In: *Computers & Chemical Engineering* 29.6, pp. 1225–1236.

Shang, C. and F. You (2018). "Distributionally robust optimization for planning and scheduling under uncertainty". In: *Computers & Chemical Engineering* 110, pp. 53–68.

Sheffi, Y. (2015). "Preparing for disruptions through early detection". In: *MIT Sloan Management Review* 57.1, pp. 31–42.

Souza, G. C. (2014). "Supply chain analytics". In: *Business Horizons* 57.5, pp. 595–605.

Sprang, E. N. van, H.-J. Ramaker, J. A. Westerhuis, S. P. Gurden, and A. K. Smilde (2002). "Critical evaluation of approaches for on-line batch process monitoring". In: *Chemical Engineering Science* 57.18, pp. 3979–3991.

Sterman, J. D. (1989). "Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment". In: *Management Science* 35.3, pp. 321–339.

Subramanian, K., C. T. Maravelias, and J. B. Rawlings (2012). "A state-space model for chemical production scheduling". In: *Computers & Chemical Engineering* 47, pp. 97–110.

Subramanian, K., J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan (2013). "Integration of control theory and scheduling methods for supply chain management". In: *Computers & Chemical Engineering* 51, pp. 4–20.

Tang, C. S. (2006a). "Perspectives in supply chain risk management". In: *International Journal of Production Economics* 103.2, pp. 451–488.

— (2006b). "Robust strategies for mitigating supply chain disruptions". In: *International Journal of Logistics Research and Applications* 9.1, pp. 33–45.

Tiwari, S., H. Wee, and Y. Daryanto (2018). "Big data analytics in supply chain management between 2010 and 2016: Insights to industries". In: *Computers & Industrial Engineering* 115, pp. 319–330.

Toole, J. L., S. Colak, B. Sturt, L. P. Alexander, A. Evsukoff, and M. C. González (2015). "The path most traveled: Travel demand estimation using big data resources". In: *Transportation Research Part C: Emerging Technologies* 58, pp. 162–177.

Tsai, C.-Y. and S.-H. Huang (2015). "A data mining approach to optimise shelf space allocation in consideration of customer purchase and moving behaviours". In: *International Journal of Production Research* 53.3, pp. 850–866.

Tsiakis, P., N. Shah, and C. C. Pantelides (2001). "Design of multi-echelon supply chain networks under demand uncertainty". In: *Industrial & Engineering Chemistry Research* 40.16, pp. 3585–3604.

Wang, G., A. Gunasekaran, E. W. Ngai, and T. Papadopoulos (2016a). "Big data analytics in logistics and supply chain management: Certain investigations for research and applications". In: *International Journal of Production Economics* 176, pp. 98–110.

Wang, T., H. Wu, M. Ni, M. Zhang, J. Dong, M. E. H. Benbouzid, and X. Hu (2016b). "An adaptive confidence limit for periodic non-steady conditions fault detection". In: *Mechanical Systems and Signal Processing* 72-73, pp. 328–345.

Wang, Y., D. E. Seborg, and W. E. Larimore (1997). "Process monitoring based on canonical variate analysis". In: *1997 European Control Conference (ECC)*, pp. 3089–3094.

Westerhuis, J. A., S. P. Gurden, and A. K. Smilde (2000). "Generalized contribution plots in multivariate statistical process monitoring". In: *Chemometrics and Intelligent Laboratory Systems* 51.1, pp. 95–114.

Wilson, M. C. (2007). "The impact of transportation disruptions on supply chain performance". In: *Transportation Research Part E: Logistics and Transportation Review* 43.4, pp. 295–320.

Wold, S., K. Esbensen, and P. Geladi (1987). "Principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 2.1, pp. 37–52.

Yoon, S. and J. F. MacGregor (2001). "Fault diagnosis with multivariate statistical models part I: using steady state fault signatures". In: *Journal of Process Control* 11.4, pp. 387–400.

You, F. and I. E. Grossmann (2008). "Design of responsive supply chains under demand uncertainty". In: *Computers & Chemical Engineering* 32.12, pp. 3090–3111.

Zhao, R., Y. Liu, N. Zhang, and T. Huang (2017). "An optimization model for green supply chain management by using a big data analytic approach". In: *Journal of Cleaner Production* 142, pp. 1085–1097.

Zhong, R. Y., G. Q. Huang, S. Lan, Q. Dai, T. Zhang, and C. Xu (2015). "A two-level advanced production planning and scheduling model for RFID-enabled ubiquitous manufacturing". In: *Advanced Engineering Informatics* 29.4, pp. 799–812.

Chapter 3

Data-driven Supply Chain Monitoring Using Canonical Variate Analysis

For the study in this chapter, the following paper will be submitted:

[1] Jing Wang, Christopher L.E. Swartz, and Kai Huang. "Data-driven supply chain monitoring using canonical variate analysis". *Computers and Chemical Engineering*. To be submitted.

3.1 Introduction

In Chapter 2, we developed a SCMo framework based on a data-driven method, PCA. The study considers the SCMo problem from a process engineering perspective and adopts the statistical process monitoring method for fault detection and diagnosis of supply chain systems. The results show that PCA-based SCMo is able to detect and diagnose abnormal supply chain operations. This technique of applying data-driven methods to SCMo is called *data-driven SCMo* in this thesis. Although the study in Chapter 2 focuses on the PCA method, it indicates the potential of applying other data-driven methods to SCMo. In this chapter, we follow this research direction and continue exploring data-driven SCMo with a different statistical process monitoring method, canonical variate analysis (CVA).

It is worth mentioning that in the context of global supply chains, supply chain systems are exposed to many risks, and SCMo is poised to gain more attention. An example that shows the importance of being aware of the status of a supply chain is the COVID-19 pandemic, which occurred with an extremely low probability while has impacted global supply chains adversely and significantly since it started. The impacts of the COVID-19 pandemic on supply chains include spike or decline in demand of products, shortage of supply and products, production disruption, delays in transportation and distribution, shift of logistics pattern, and ripple effect (Queiroz et al., 2020; Chowdhury et al., 2021). The consequences of supply chain disruptions have resulted in an increasing interest in supply chain risk management (Fahimnia et al., 2015). It is important and often challenging to assess, monitor, and raise awareness of supply chain risks (Heckmann et al., 2015; Tran et al., 2018). The ability to detect and provide early warning of abnormal operations is vital to SCM. Tracking the operating status of a supply chain and monitoring the uncertainties continuously reveals the gap between planning and execution, thus helping to identify potential problems and improvement strategies. For example, regular monitoring of the obsolete inventory allows supply chain participants to take more proactive measures and reduces the overall inventory cost (Chae, 2009). SCMo has become an integral part of SCM and received much attention of both academics and practitioners. The goal of SCMo is to

support decision-making through characterizing the normal operating conditions of a supply chain, raising warnings of abnormal situations, identifying potential reasons, and providing suggestions (Mele et al., 2005; Wang et al., 2020). Moreover, SCMo is considered to be a critical component of supply chain visualization (Goh et al., 2013). Therefore, in this chapter, we further investigate the use of data-driven methods to provide deeper insights to SCMo.

The methods we consider for SCMo are mainly statistical process monitoring (SPM) methods that have been established in the process system engineering community. SPM, also known as data-driven process monitoring, describes the area of applying data-driven and multivariate statistical methods to fault detection and diagnosis for industrial processes (Russell et al., 2000a; MacGregor and Cinar, 2012; Qin, 2012; Jiang et al., 2019). It has been well-established and recognized in academia and industry over the past three decades. In comparison to univariate methods, multivariate SPM methods have advantages such as providing more precise process information and raising earlier warnings (Kourti and MacGregor, 1995; Norvilas et al., 2000; MacGregor and Cinar, 2012). Classical data-driven methods include PCA, partial least squares (PLS), CVA, as well as their variants (Kresta et al., 1991; Ku et al., 1995; Russell et al., 2000b; Severson et al., 2016). PCA is a linear dimensionality reduction technique that extracts uncorrelated components from cross-correlated data by maximizing the variance explained. PCA-based SCMo has been investigated in Chapter 2. PLS is another linear dimensionality reduction technique, while it deals with two groups of variables and aims to maximizes the covariance between them in the reduced space. It is mainly used for monitoring quality variables. CVA utilizes the state-space model of a system and discovers the relationship between the past and future system data. The canonical states in a CVA model are determined by maximizing the correlation between a linear combination of past outputs and inputs and a linear combination of future outputs.

CVA is considered to provide a better representation of system dynamics, and to be a more suitable method for monitoring dynamic systems than dynamic PCA and PLS in some works (Odiowei and Cao, 2010; Jiang et al., 2015a; Jiang et al., 2015c; Ruiz-Cárcel et al., 2015; Lu et al., 2018a). Therefore, in this chapter, we investigate a CVA-based SCMo method. The main contributions of this chapter are:

1. The development of a state-space model of a supply chain;
2. The application of CVA to SCMo;
3. A modification to the Q-statistic of sparse CVA model based on the right Moore-Penrose inverse;
4. An approach to hyperparameter tuning for the sparse CVA method;
5. A comparative study on data-driven SCMo based on CVA and PCA, in terms of dimensionality reduction, false alarm rate, missed detection rate, and detection delay.

The remainder of this chapter is organized as follows. Section 3.2 presents a literature review on SCMo and the CVA method. Then, the data-driven SCMo method based on CVA is proposed in Section 3.3. In Section 3.4, two case studies are presented, and the performance of the proposed method is assessed and compared with dynamic PCA. The conclusion is given in Section 3.5.

3.2 Literature Review

The data-driven SCMo proposed in this chapter involves SCMo and a data-driven method, CVA. Relevant literature for the two elements are reviewed in Section 3.2.1 and Section 3.2.2, respectively.

3.2.1 Supply chain monitoring

The literature review in Chapter 2 focused on data analytics in supply chain, rather than SCMo. In this section, a comprehensive review of SCMo in the current literature is presented as follows. Lau et al. (2002) proposed a fuzzy logic scheme to monitor the performance of suppliers and adjust the next order quantity based on historical data. The criteria considered are the product quality and delivery time. Bansal et al. (2005) designed an agent-based framework for the supply chain disruption management system. This system monitors key

performance indicators (KPIs) of a supply chain, such as inventory and throughput, and identifies the potential causes for the alarms using causal models. Bae and Seo (2007) focused the development of an integrated architecture of supply chain process modelling, execution and monitoring based on business process management techniques. They proposed a role-based supply chain process monitoring model where each supply chain participant is defined an access level of monitoring information. Fei and Wang (2008) proposed a supply chain monitoring and early-warning method based on the system dynamics model of supply chain. A recurrent neural network that predicts the dynamic behaviour of indicators acts as the real-time monitoring engine. Chae (2009) proposed a list of primary and secondary KPIs for measuring supply chain performance, and suggested that the supply chain operation be overseen by a cross-functional group through monitoring the proposed KPIs. This group also takes the lead in identifying the root causes of events detected by low-performing KPIs, communicating with responsible units, and elaborating improvement strategies. Zhou and Rong (2010) proposed a multi-level SCMo model based on the Integration Definition for Function Modelling (IDEF0) method. A supply chain IDEF0* model is developed based on IDEF0. This structured analysis approach incorporates the characteristics of supply chain and models the system events and information flow. Fernández et al. (2012) presented a model-driven development method for generating the monitoring component within the supply chain event management (SCEM) system. The monitoring model predicts disruptive events by analyzing the impact of the resource attributes and environment parameters on the order attributes. Goh et al. (2013) developed a supply chain visualization platform with real-time risk monitoring based on multi-hierarchical modular design. The platform visualizes the spatio-temporal connectivity patterns of supply chain entities, and monitors the logistics, inventory, order and manufacturing information, as well as real-time risk-related information such as natural disasters. Irizarry et al. (2013) proposed a visualization system for monitoring resources in the construction SCM through integrating building information modelling and geographic information systems. The system keeps track of the status of materials within the supply chain and generates warning signals to ensure timely delivery. Fernández et al. (2015) designed a multi-agent SCMo system for detecting and predicting disruptive events. The system consists of a set of structured autonomous agents that are able

to concurrently monitor orders and resources in different schedules. McKinney et al. (2015) studied the development and application of container monitoring devices for the tracking of supply chain systems. They showed that supply chain monitoring and visibility bring not only short-term benefits such as awareness of actual real-time terminal activities, but also long-term benefits such as more reliable end-to-end service to businesses. Jiang et al. (2017) proposed a SCMo framework composed of interacting components. Data associated to real-time inventory, order and transportation information, machine status, supply chain plans and situations are collected. An alert generator propagates the detected deviation on one product to associated products to assess the consequence, and sends the alert for notification, analysis and solution. Blos et al. (2018) presented a framework for modelling and monitoring supply chain disruptions using Petri net and agent-based model techniques. Probabilistic disruption scenarios are generated based on historical database to foresee the occurrence and duration of potential supply chain disruptions. The agent-based model is adopted to derive mitigation actions and evaluate the impact of the disruptions.

Mele et al. (2005) and Wang et al. (2020) considered the SCMo problem from a process engineering perspective and adopted the SPM methods. Mele et al. (2005) investigated the performance of a multi-scale delay adjusted PCA method in monitoring a supply chain network simulated by an event-driven model, and obtained promising results. Wang et al. (2020) investigated the fault detection and diagnosis of supply chain systems using dynamic PCA and agent-based modelling. This study shows that PCA-based monitoring statistics and contributions can detect and diagnose abnormal supply chain operation caused by transportation delay, poor manufacturing yield and supply shortage. This technique of applying data-driven methods to SCMo is called *data-driven SCMo* in this chapter.

3.2.2 Data-driven process monitoring based on CVA

CVA, also known as canonical correlation analysis (CCA) in the statistics community, is a data-driven method for multivariate statistical analysis (Russell et al., 2000a; Hardoon et al., 2004). It was introduced to system and process identification by Larimore (1990) and Schaper et al. (1994), investigated for SPM later (Larimore, 1997; Wang et al., 1997; Negiz

and Çinar, 1997; Russell et al., 2000b), and has received wide attention (Odiowei and Cao, 2010; Ruiz-Cárcel et al., 2015; Ruiz-Cárcel et al., 2016; Jiang et al., 2015a; Jiang et al., 2015b; Jiang and Braatz, 2017; Severson et al., 2016).

Negiz and Çinar (1997) demonstrated the ability of CVA in dealing with the cross-correlation and autocorrelation in data. This study showed CVA can determine the minimal state order for describing a dynamic system, while classical PCA and PLS are unable to identify it by non-zero eigenvalues or singular values. Russell et al. (2000b) compared the performance of DPCA and CVA using the Tennessee Eastman process. The results show that CVA-based residual monitoring statistics usually achieve smaller detection delays than DPCA, with properly adjusted control limits. Simoglou et al. (2002) investigated the performance of CVA and PLS in the monitoring of a continuous stirred tank co-polymerisation reactor. It is shown that the two approaches generally achieve comparable results in fault detection, while CVA has fewer false alarms. Moreover, the statistics based on the output residuals are found to be the most reliable, where the two approaches perform equally well. Juricek et al. (2004) investigated process monitoring based on the multivariate state-space model identified by CVA. A fault detection method based on the local statistic for canonical variables is proposed, and the comparison with the Kalman filter residuals and canonical variable residuals showed its sensitivity for detecting process changes. Treasure et al. (2004) pointed out that a limitation of DPCA is that the model size can be considerable when the process is complex and a large number of process variables are involved. This makes it more difficult for fault diagnosis using contribution plots, given that time-lagged values are involved. By comparison, through integrating PCA into subspace model identification, significant fewer variables are needed. Odiowei and Cao (2010) proposed a kernel density estimation method to determine the upper control limits of the CVA monitoring statistics. This method is more appropriate for the monitoring of nonlinear dynamic process where the statistics violate the Gaussian assumption, and achieves better performance than dynamic PCA and PLS. Jiang et al. (2015a) proposed CVA-based variable contributions based on the monitoring statistics in the canonical state and residual spaces for fault identification. The fault-related variables are classified into state-space and residual-space faulty variables, which provides insights into the characteristics of the process and faults. Jiang et al. (2015b) used CVA to

quantify the dissimilarity among the causal dependencies, which are the feature measures generated from the original process data to facilitate the monitoring of process correlation structures. Ruiz-Cárcel et al. (2015) designed a benchmark case for assessing the performance of SPM techniques based on an industrial scale experimental multiphase flow facility and artificially seeded faults. It is demonstrated that CVA can be effectively applied for monitoring real complex systems, and the performance of CVA including Kernel Density Estimators is superior to PCA and PLS methods in terms of detection rate, false alarm rate and detection time. The case study is further investigated by Ruiz-Cárcel et al. (2016) to examine the system identification capability of CVA. CVA is adopted to identify the system not only under normal operating conditions, but also under faulty conditions to estimate performance degradation and predict the behaviour of a faulty system. Jiang and Braatz (2017) proposed two monitoring statistics to measure the variation of dynamic correlations in the state and residual spaces of CVA model. The proficiency of monitoring is improved by utilizing effective feature representation of the canonical correlation with reduced redundancy in the feature space. Lu et al. (2018b) proposed a sparse CVA method that is applicable when the sample covariance matrices are highly ill-conditioned or even singular. This method has the advantages of sparse canonical vectors in interpreting the process and promoting the identification of faulty variables.

In general, PCA and PLS are limited in their ability to generate accurate models for dynamic processes, even though the observation vector has been augmented with lagged data of process variables (Odiowei and Cao, 2010; Jiang et al., 2015a; Jiang et al., 2015c; Lu et al., 2018a). Thus, the contribution plots based on them have limited ability in fault diagnosis. To this end, fault detection and diagnosis based on state-space models have been investigated to better deal with the process dynamics. CVA can outperform dynamic PCA and alternative identification methods in some case studies. CVA finds the direction that yields maximum correlation information, thus it can achieve a similar prediction accuracy to PLS using fewer latent variables (Yao and Gao, 2008).

Based on the literature review, we found that although many approaches have been proposed, data-driven SCMo by considering the supply chain as a system from a process system point of view has not been investigated to a satisfactory extent. In this chapter, we focus on the

investigation of data-driven SCMo. Considering the advantages of CVA, this work aims to extend the current research on SCMo through investigating the use of CVA for data-driven SCMo. As each node of the supply chain network generates data, the data-driven SCMo method extracts useful information from supply chain data. The proposed CVA-based SCMo method falls within the range of descriptive analytics. The goal is to support decision-making through raising alarm when the supply chain is abnormal when the fault has affected the operation of supply chain (Mele et al., 2005).

3.3 Data-driven SCMo Based on CVA

This section introduces the proposed data-driven SCMo method using CVA. Section 3.3.1 describes the methodology of CVA for SPM, and the monitoring statistics and their confidence limits. Section 3.3.2 presents the method of using CVA for SCMo and the state-space model of a supply chain. Section 3.3.3 describes the hyperparameter tuning method used in this work.

3.3.1 Statistical process monitoring using CVA

As a data-driven method, CVA discovers the relationship between two group of variables and constructs a model from data. Consider two sets of process variables $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. The historical data for x and y over N time periods are denoted as $X = [x(1), x(2), \ldots, x(N)]^T \in \mathbb{R}^{N\times m}$ and $Y = [y(1), y(2), \ldots, y(N)]^T \in \mathbb{R}^{N\times n}$, respectively. Assume X and Y have been preprocessed to have zero means. The sample covariance matrices of X and Y, Σ_{xx} and Σ_{yy}, and the sample cross-covariance matrix between X and Y, Σ_{xy}, are calculated as:

$$\Sigma_{xx} = \frac{1}{N-1}X^T X, \quad \Sigma_{yy} = \frac{1}{N-1}Y^T Y, \quad \Sigma_{xy} = \frac{1}{N-1}X^T Y \tag{3.1}$$

CVA aims to find a pair of canonical vectors $\alpha \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^n$ to construct projections of

x and y, which are $\alpha^T x$ and $\beta^T y$, such that the correlation between them is maximized. This results in the following optimization problem (Hardoon et al., 2004; Odiowei and Cao, 2010; Yang et al., 2021):

$$\max_{\alpha,\beta} corr(\alpha^T x, \beta^T y) = \frac{\alpha^T \Sigma_{xy} \beta}{\sqrt{\alpha^T \Sigma_{xx} \alpha}\sqrt{\beta^T \Sigma_{yy} \beta}} = \frac{\alpha^T X^T Y \beta}{\sqrt{\alpha^T X^T X \alpha}\sqrt{\beta^T Y^T Y \beta}} \tag{3.2}$$

In the above optimization problem (3.2), rescaling α or β does not affect the value of the objective function (Hardoon et al., 2004). Therefore, problem (3.2) can be written as (Hardoon et al., 2004; Yang et al., 2021):

$$\begin{aligned} &\max_{\alpha,\beta} \alpha^T \Sigma_{xy} \beta \\ \text{s. t.} \quad &\alpha^T \Sigma_{xx} \alpha = 1 \\ &\beta^T \Sigma_{yy} \beta = 1 \end{aligned} \tag{3.3}$$

or equivalently (Witten et al., 2009; Lu et al., 2018b):

$$\begin{aligned} &\max_{\alpha,\beta} \alpha^T X^T Y \beta \\ \text{s. t.} \quad &\alpha^T X^T X \alpha \leq 1, \quad \beta^T Y^T Y \beta \leq 1 \end{aligned} \tag{3.4}$$

Letting $u = \Sigma_{xx}^{1/2} \alpha, v = \Sigma_{yy}^{1/2} \beta$, the above optimization problem (3.3) can be written as (Odiowei and Cao, 2010; Yang et al., 2021):

$$\begin{aligned} &\max_{u,v} u^T \Sigma_{xx}^{-1/2} \Sigma_{xy} \Sigma_{yy}^{-1/2} v \\ \text{s. t.} \quad &u^T u = 1 \\ &v^T v = 1 \end{aligned} \tag{3.5}$$

Denote $\overline{\Sigma}_{xy} = \Sigma_{xx}^{-1/2} \Sigma_{xy} \Sigma_{yy}^{-1/2}$. The Lagrangian associated with the constrained optimization problem (3.5) can be formulated as (Yang et al., 2021):

$$L(u, v, \lambda_1, \lambda_2) = u^T \overline{\Sigma}_{xy} v - \frac{\lambda_1}{2}(u^T u - 1) - \frac{\lambda_2}{2}(v^T v - 1) \tag{3.6}$$

where λ_1 and λ_2 are the Lagrange multipliers associated with the constraints.

Taking the partial derivatives and setting them equal to 0, we have:

$$\begin{aligned} \frac{\partial L}{\partial u} &= \overline{\Sigma}_{xy} v - \lambda_1 u = 0 \\ \frac{\partial L}{\partial v} &= \overline{\Sigma}_{xy}^T u - \lambda_2 v = 0 \end{aligned} \tag{3.7}$$

Multiplying the above two equations by u^T and v^T on the left, respectively, we have:

$$\begin{aligned} u^T \overline{\Sigma}_{xy} v - \lambda_1 u^T u = 0 \\ v^T \overline{\Sigma}_{xy}^T u - \lambda_2 v^T v = 0 \end{aligned} \tag{3.8}$$

Since $u^T u = v^T v = 1$, from (3.8), we have $u^T \overline{\Sigma}_{xy} v = \lambda_1 = \lambda_2$. Denote $\lambda_1 = \lambda_2 = \lambda$; then

$$\begin{aligned} \overline{\Sigma}_{xy} v = \lambda u \\ \overline{\Sigma}_{xy}^T u = \lambda v \end{aligned} \tag{3.9}$$

From Equation (3.9), we have:

$$\begin{aligned} \overline{\Sigma}_{xy}^T \overline{\Sigma}_{xy} v = \lambda \overline{\Sigma}_{xy}^T u = \lambda^2 v \\ \overline{\Sigma}_{xy} \overline{\Sigma}_{xy}^T u = \lambda \overline{\Sigma}_{xy} v = \lambda^2 u \end{aligned} \tag{3.10}$$

Equation (3.10) implies that u is the eigenvector of $\overline{\Sigma}_{xy} \overline{\Sigma}_{xy}^T$, i.e., $\Sigma_{xx}^{-1/2} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1/2}$, corresponding to λ^2; and v is the eigenvector of $\overline{\Sigma}_{xy}^T \overline{\Sigma}_{xy}$, i.e., $\Sigma_{yy}^{-1/2} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1/2}$, corresponding to λ^2. This indicates that u and v are the left and right singular vectors of $\overline{\Sigma}_{xy}$ corresponding to the singular value λ, respectively (Klema and Laub, 1980; Odiowei and Cao, 2010; Yang et al., 2021).

Furthermore, notice that $u^T \overline{\Sigma}_{xy} v = \lambda$ is the objective function to be maximized in problem (3.5). This means the solution u and v are the singular vector corresponding to the largest singular value. Then, α and β are computed by $\alpha = \Sigma_{xx}^{-1/2} u$ and $\beta = \Sigma_{yy}^{-1/2} v$, respectively.

In this way, the problem (3.3) is solved.

The linear combinations $\alpha^T x$ and $\beta^T y$ are referred to as the *canonical variables* (Jiang et al., 2015a; Jiang et al., 2015b). If more than one pair of canonical variables need to be extracted, then the vectors u and v are computed according to a descending order of the singular values. Thus, the CVA problem can be solved by the following singular value decomposition (SVD):

$$\overline{\Sigma}_{xy} = \Sigma_{xx}^{-1/2}\Sigma_{xy}\Sigma_{yy}^{-1/2} = U\Sigma V^T \tag{3.11}$$

where Σ is a $m \times n$ matrix with the singular values of $\overline{\Sigma}_{xy}$ on its diagonal; $U = [u_1, u_2, \ldots, u_m]$ and $V = [v_1, v_2, \ldots, v_n]$ are orthogonal matrices with their first $\min(m, n)$ columns being the corresponding left and right singular vectors.

Assume that we retain d pairs of canonical variables, i.e., we retain the first d ($d \leq \min(m, n)$) columns of U and V. Then, for $i = 1, 2, \ldots, d$, we have

$$\begin{aligned} x_d^i &= \alpha_i^T x = u_i^T \Sigma_{xx}^{-1/2} x \\ y_d^i &= \beta_i^T y = v_i^T \Sigma_{yy}^{-1/2} y \end{aligned} \tag{3.12}$$

where u_i and v_i are the singular vectors corresponding to λ_i.

The d pairs of canonical variables are expressed as $x_d = [x_d^1, x_d^2, \ldots, x_d^d]^T$ and $y_d = [y_d^1, y_d^2, \ldots, y_d^d]^T$. Denote $U_d = [u_1, u_2, \ldots, u_d]$ and $V_d = [v_1, v_2, \ldots, v_d]$. Let

$$\begin{aligned} J_d &= [\alpha_1, \alpha_2, \ldots, \alpha_d]^T = U_d^T \Sigma_{xx}^{-1/2} \\ L_d &= [\beta_1, \beta_2, \ldots, \beta_d]^T = V_d^T \Sigma_{yy}^{-1/2} \end{aligned} \tag{3.13}$$

Then, the canonical variables can be computed by $x_d = J_d x$ and $y_d = L_d y$.

For a sample $x(t)$ in X, $x_d(t) = J_d x(t)$, and $X_d = [x_d(1), x_d(2), \ldots, x_d(N)]^T = XJ_d^T$. The sample covariance matrix of x_d can be calculated as

$$\Sigma_{xx}^d = \frac{1}{N-1} X_d^T X_d = J_d X^T X J_d^T = U_d^T \Sigma_{xx}^{-1/2} \Sigma_{xx} \Sigma_{xx}^{-1/2} U_d = I_d \tag{3.14}$$

where $I_d \in \mathbb{R}^{d\times d}$ is an identity matrix.

Similarly, $\Sigma_{yy}^{d} = I_d$. This means J_d transforms the variables in x into mutually uncorrelated canonical variables, and L_d transforms the variables in y into mutually uncorrelated canonical variables. Meanwhile, the canonical variables in x_d and y_d are pairwise correlated.

The use of CVA for data-driven process monitoring is based on the state-space model of a system, which can be expressed as follows:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + v(t) \\ y(t) &= Cx(t) + Du(t) + Ev(t) + w(t) \end{aligned} \tag{3.15}$$

where t is the time period, $x(t)$ is a k-order state vector; $u(t) \in \mathbb{R}^{m_u}$ and $y(t) \in \mathbb{R}^{m_y}$ are the input and output of the system, respectively; $v(t)$ and $w(t)$ are independent white noise processes.

Constructing a CVA model means finding the best linear combinations of the past and future observations such that the correlation between them is maximized (Odiowei and Cao, 2010; Jiang et al., 2015a). For period t, the past vector $p(t)$ comprises of the system outputs and inputs during the past l periods, and the future vector $f(t)$ comprises of the system outputs during the h periods starting from period t:

$$\begin{aligned} p(t) &= [y^T(t-1), y^T(t-2), \ldots, y^T(t-l), u^T(t-1), u^T(t-2), \ldots, u^T(t-l)]^T \in \mathbb{R}^{n_p} \\ f(t) &= [y^T(t), y^T(t+1), \ldots, y^T(t+h-1)]^T \in \mathbb{R}^{n_f} \end{aligned} \tag{3.16}$$

Here, $n_p = l(m_u + m_y)$, and $n_f = hm_y$. The matrices for past and future information, P and F, are constructed by stacking $p(t)$ and $f(t)$ as follows:

$$\begin{aligned} P &= [p(l+1), p(l+2), \ldots, p(N-h)]^T \\ F &= [f(l+2), f(l+3), \ldots, f(N+1-h)]^T \end{aligned} \tag{3.17}$$

The dimensions of P and F are $\mathbb{R}^{(N-l-h)\times(m_u l + m_y l)}$ and $\mathbb{R}^{(N-l-h)\times(m_y h)}$, respectively. Then, the CVA problem for system (3.15) is obtained by substituting matrices X and Y in problem

(3.3) with P and F, respectively. And the resulting SVD problem is obtained by substituting matrices Σ_{xx}, Σ_{yy}, and Σ_{xy} in problem (3.11) with Σ_{pp}, Σ_{ff}, and Σ_{pf}, respectively. Dimensionality reduction is achieved by retaining the first d *canonical states* (CSs), i.e., $x_d(t) = J_d p(t) = U_d^T \Sigma_{pp}^{-1/2} p(t)$, where U_d contains the first d columns of U. The process monitoring statistics can be defined based on $x_d(t)$.

It should be noted that according to Equation (3.11), building a CVA model requires computing the inverse of $\Sigma_{pp}^{1/2}$ and $\Sigma_{ff}^{1/2}$. In other words, CVA implicitly assumes that Σ_{pp} and Σ_{ff} are invertible. If at least one of these two matrices is singular and cannot be inverted, CVA fails immediately. Even if they are ill-conditioned, computational errors exist and the monitoring statistics can be overly sensitive (Lu et al., 2018b). This problem is found to occur when we apply the CVA method to supply chain data, where the two matrices are not invertible. Therefore, a model that can work with singular Σ_{pp} or Σ_{ff} is needed.

In this chapter, we consider the use of the sparse CVA (SCVA) method (Lu et al., 2018b). SCVA is a process monitoring method proposed based on a penalized matrix decomposition method for sparse CCA (Witten et al., 2009; Lu et al., 2018b). The SCVA problem is derived from the classical CVA problem. In problem (3.4), substituting X and Y with P and F yields:

$$\begin{aligned} &\max_{\alpha,\beta} \alpha^T P^T F \beta \\ &\text{s. t.} \quad \alpha^T P^T P \alpha \leq 1, \quad \beta^T F^T F \beta \leq 1 \end{aligned} \tag{3.18}$$

Then, through approximating the sample covariance matrices $P^T P$ and $F^T F$ with the identity matrices, the following optimization problem is obtained:

$$\begin{aligned} &\max_{\alpha,\beta} \alpha^T P^T F \beta \\ &\text{s. t.} \quad \alpha^T \alpha \leq 1, \quad \beta^T \beta \leq 1 \end{aligned} \tag{3.19}$$

Next, two extra constraints are imposed on the l_1-norms of α and β to enforce their sparsity:

$$\begin{aligned} &\max_{\alpha,\beta} \alpha^T P^T F \beta \\ &\text{s. t.} \quad \|\alpha\|_2 \leq 1, \quad \|\beta\|_2 \leq 1, \quad \|\alpha\|_1 \leq c_1, \quad \|\beta\|_1 \leq c_2 \end{aligned} \tag{3.20}$$

Problem (3.20) is the optimization problem associated with a SCVA model (Witten et al., 2009; Lu et al., 2018b). Problem (3.19) gets rid of the singular covariance matrices through an approximation. It has been shown that in other high-dimensional problems, treating the covariance matrix as diagonal can yield good results (Witten et al., 2009). The two extra constraints in problem (3.20) restrict α and β to contain only fewer nonzero components by setting upper bounds on the summation of absolute values of the components of them. The sparse canonical vectors obtained in this way can help reveal important relationships between process variables, and strengthen the contributions of major fault-related variables, thus improving fault identification. SCVA is shown to be able to achieve better fault detection performance than CVA for the Tennessee Eastman process, and to have advantage in singling out faulty variables (Lu et al., 2018b).

The solutions of problem (3.20) can be efficiently computed by a penalized matrix decomposition, Algorithm 3.1 (Witten et al., 2009; Lu et al., 2018b). The algorithm takes the matrices P and F as inputs and returns the matrices J_d and L_d. There are 3 hyperparameters that need to be specified for the algorithm to work, which are the number of canonical states to extract, d, and the sparsity parameter c_1 and c_2. The algorithm begins with computing the first pair of canonical vectors α_1 and β_1 by an iterative method. After α_1 and β_1 are obtained, the matrix $Z^1 = P^T F$ is deflated to get Z^2, and the second pair of canonical vectors α_2 and β_2 are computed based on the deflated matrix. This process repeats until d pairs of canonical vectors are obtained. In Algorithm 3.1, $S(a, c) = sign(a)(|a| - c)_+$ is a soft-thresholding function, with $sign(a)$ returns the sign of the elements of vector a, and x_+ returns x if $x > 0$ and 0 otherwise.

For classical CVA, the Hotelling's T^2 statistic can be used to measure the variation within the canonical state space, which is defined as T_d^2 (Russell et al., 2000a; Russell et al., 2000b;

Algorithm 3.1: SCVA algorithm (Witten et al., 2009; Lu et al., 2018b)

Data: P, F
Result: J_d, L_d
Parameters: d, c_1, c_2

$Z^1 \leftarrow P^T F$
for $k = 1, 2, \ldots, d$ **do**
  Initialize v with $\|v\|_2 = 1$
  while *not converge* **do**
    $u \leftarrow \frac{S(Z^k v, \Delta_1)}{\|S(Z^k v, \Delta_1)\|_2}$ where $\Delta_1 = 0$ if it results in $\|u\|_1 \leq c_1$; otherwise, Δ_1 is chosen by a binary search such that $\|u\|_1 = c_1$
    $v \leftarrow \frac{S((Z^k)^T u, \Delta_2)}{\|S((Z^k)^T u, \Delta_2)\|_2}$ where $\Delta_2 = 0$ if it results in $\|v\|_1 \leq c_2$; otherwise, Δ_2 is chosen by a binary search such that $\|v\|_1 = c_2$
  end
  $\gamma_k \leftarrow u^T Z v, \alpha^k \leftarrow u, \beta^k \leftarrow v$
  $Z^{k+1} \leftarrow Z^k - \gamma_k u v^T$
end
$J_d \leftarrow [\alpha_1, \ldots, \alpha_d]^T, L_d \leftarrow [\beta_1, \ldots, \beta_d]^T, D \leftarrow diag(\gamma_1, \ldots, \gamma_d)$

Jiang et al., 2015a; Jiang et al., 2015b):

$$T_d^2(t) = x_d^T(t) x_d(t) \tag{3.21}$$

For quantifying the variation in the residual space, the following T_r^2 statistic can be defined (Russell et al., 2000b; Jiang et al., 2015a; Jiang et al., 2015c):

$$T_r^2(t) = x_e^T(t) x_e(t)$$

where $x_e(t) = J_e p(t) = U_e^T \Sigma_{pp}^{-1/2} p(t)$, with U_e containing the last $e = n_p - d$ columns of U.

Alternatively, a Q-statistic, $Q = r^T(t) r(t)$, can be defined to measure the variation in the residual space, where $r(t)$ is the residual vector. $r(t)$ can be defined as $r(t) = (I - J_d^T J_d) p(t)$ (Russell et al., 2000a; Russell et al., 2000b), or $r(t) = (I - U_d U_d^T) \Sigma_{pp}^{-1/2} p(t)$ (Odiowei and Cao, 2010; Samuel and Cao, 2015; Ruiz-Cárcel et al., 2015; Ruiz-Cárcel et al., 2016).

Given a significance level α, the confidence limits for T_d^2 and T_r^2 can be estimated using the F-distribution:

$$\begin{aligned} T_{d,\alpha}^2 &= \frac{d(N^2-1)}{N(N-d)} F_\alpha(d, N-d) \\ T_{r,\alpha}^2 &= \frac{e(N^2-1)}{N(N-e)} F_\alpha(e, N-e) \end{aligned} \tag{3.22}$$

where d is the number of canonical states, and N is the number of samples. Usually $\alpha =$ 95% or 99% is used. It should be noted that T_r^2 can be overly sensitive due to the matrix inversion. Thus its confidence limit should be adjusted before applying the statistics for process monitoring (Russell et al., 2000b; Lu et al., 2018b).

The canonical states obtained by SCVA may be not mutually uncorrelated due to the l_1 constraints. Therefore, the T_d^2 statistic is defined with the covariance matrix taken into account (Lu et al., 2018b):

$$T_d^2(t) = x_d^T(t) \Lambda^{-1} x_d(t) \tag{3.23}$$

where Λ is the covariance matrix of canonical state $x_d(t)$ computed from training data. The confidence limit of T_d^2 can be estimated by using a F-distribution similar to T_d^2 in Equation (3.22).

For SCVA, the Q-statistic that measures the variation in the residual space is calculated based on a reconstructed vector $\hat{p}(t) = J_d^+ x_d(t)$. According to Lu et al. (2018b), $J_d^+ = (J_d^T J_d)^{-1} J_d^T$ is the Moore-Penrose pseudoinverse of J_d. The residual is then calculated as:

$$r(t) = p(t) - \hat{p}(t) = [I - J_d^+ J_d] p(t) \tag{3.24}$$

It is worth mentioning that $J_d \in \mathbb{R}^{d \times n_p}$. Therefore, if $d < n_p$, then $rank(J_d^T J_d) \leq d < n_p$. Hence, $J_d^T J_d \in \mathbb{R}^{n_p \times n_p}$ is not invertible, and the above J_d^+ is unavailable. This implies that the left Moore-Penrose pseudoinverse may be not applicable under this circumstance. Note that $J_d J_d^T \in \mathbb{R}^{d \times d}$ is invertible, thus in this chapter, we use the right Moore-Penrose pseudoinverse of J_d, $J_d^+ = J_d^T (J_d J_d^T)^{-1}$ (Barata and Hussein, 2012), to calculate the residual:

$$r(t) = p(t) - \hat{p}(t) = [I - J_d^T (J_d J_d^T)^{-1} J_d] p(t) \tag{3.25}$$

The confidence limit of Q corresponding to a significance level α, denoted as Q_α, is determined in such way that a $(1-\alpha)$ portion of the training samples are below it. T_d^2 and Q are used as a pair to indicate significant deviation from the normal operating conditions (NOC). An alarm is raised when at least one of them exceeds the confidence limit. This is defined by a overall measure $S_{overall}$ (Lu et al., 2018b):

$$S_{overall} = \begin{cases} 1, & \text{if} \quad T_d^2 > T_{d,\alpha}^2 \quad \text{or} \quad Q > Q_\alpha \\ 0, & \text{otherwise} \end{cases} \tag{3.26}$$

If $S_{overall} = 1$, then the SCVA model raises an alarm. When a fault is detected, the next step is to identify potential fault-related variables. A widely used approach for fault diagnosis is the contribution plot (Kourti and MacGregor, 1995; MacGregor and Cinar, 2012). The variables with large contributions to the deviation of monitoring statistics are considered most likely to be fault related. This greatly helps to narrow the scope of potential fault-related variables.

As defined in Equation (3.16), a sample $p(t)$ comprises of the input and output variables over l time lags. Therefore, the contribution of an input variable u_k in the state space is calculated as a sum of the contribution of corresponding elements over the l lags (Lu et al., 2018b):

$$C_{u_k}^d(t) = \sum_{j=1}^{l} x_d^T(t)\Lambda^{-1}u_k(t-j)J_{d,k_j} \tag{3.27}$$

where k_j is the index of the column of J_d that corresponds to the lagged variable $u_k(t-j)$.

The contribution in the residual space is calculated as (Lu et al., 2018b):

$$C_{u_k}^r(t) = \sum_{j=1}^{l} r^T(t)u_k(t-j)J_{e,k_j} \tag{3.28}$$

where $J_e = I - J_d^+ J_d$, and k_j is the index of the column of J_e that corresponds to the lagged variable $u_k(t-j)$.

The contributions of a output variable y_k are calculated analogously. In this chapter, the combined contribution is calculated as $\frac{1}{2}\left(|C_{u_k}^d(t)| + |C_{u_k}^r(t)|\right)$ to facilitate visualization by

the contribution plot.

3.3.2 State-space model of supply chain

As the CVA-based process monitoring method is based on the assumption of a state-space model of the system, representing the supply chain system in a state-space form would lay the foundation for CVA-based SCMo. Based on the state-space model, the state, input, output, and disturbance variables can be specified to facilitate the use of CVA. In this section, the state-space model of a supply chain system is developed. A classical supply chain example, the beer distribution game, is used to illustrate the model. Then, a more general supply chain model is presented.

The beer game is a roleplay simulation of an industrial production-distribution system (Sterman, 1989; Chen and Samroengraja, 2000). It involves a four-echelon serial supply chain that deals with the production and distribution of a beer product. As shown in Figure 3.1, this supply chain consists of a factory, a distributor, a warehouse, and a retailer. This benchmark has been widely used in operation management as an example to illustrate the dynamics of supply chain system or investigate supply chain control (Chaharsooghi et al., 2008; Oroojlooyjadid et al., 2022). Each position holds an inventory of the beer product, receives orders from and ships beer to its downstream entity, orders beer from its upstream entity and receives beer after a shipping delay (except for the factory, which places production requests and receives beer after a production delay).

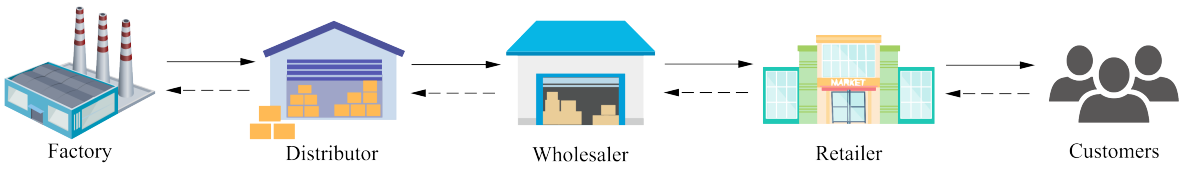


Figure 3.1. The beer game supply chain.

In each time period, the Customer agent purchases products from the Retailer agent. The Retailer places an order to the Factory according to its ordering policy. The Factory determines the production quantity and fills Retailer's orders. The agents hold inventory of the product

and record backlog. The material balances are given as follows:

$$
\begin{aligned}
I^{p}_{t+1} &= I^{p}_{t} - Q^{p,dc}_{t} + Q^{p,prod}_{t-\delta_{p,prod}}, && \forall t \in T \\
B^{p}_{t+1} &= B^{p}_{t} - Q^{p,dc}_{t} + O^{dc}_{t}, && \forall t \in T \\
I^{dc}_{t+1} &= I^{dc}_{t} - Q^{dc,w}_{t} + Q^{p,dc}_{t-\delta_{p,dc}}, && \forall t \in T \\
B^{dc}_{t+1} &= B^{dc}_{t} - Q^{dc,w}_{t} + O^{w}_{t}, && \forall t \in T \\
I^{w}_{t+1} &= I^{w}_{t} - Q^{w,r}_{t} + Q^{dc,w}_{t-\delta_{dc,w}}, && \forall t \in T \\
B^{w}_{t+1} &= B^{w}_{t} - Q^{w,r}_{t} + O^{r}_{t}, && \forall t \in T \\
I^{r}_{t+1} &= I^{r}_{t} - SA^{r}_{t} + Q^{w,r}_{t-\delta_{w,r}}, && \forall t \in T \\
B^{r}_{t+1} &= B^{r}_{t} - SA^{r}_{t} + D^{r}_{t}, && \forall t \in T
\end{aligned}
\tag{3.29}
$$

where the superscripts p, dc, w, and r represent the factory, distributor, wholesaler, and retailer, respectively; I^{p}_{t}, I^{dc}_{t}, I^{w}_{t}, I^{r}_{t} represent the stock on hand of corresponding agent at the beginning of time period t; B^{p}_{t}, B^{dc}_{t}, B^{w}_{t}, B^{r}_{t} represent the backorder of corresponding agent at the beginning of time period t; $Q^{p,prod}_{t}$ is the production quantity of the factory; $Q^{p,dc}_{t}$, $Q^{dc,w}_{t}$, and $Q^{w,r}_{t}$ are the shipment quantities from the factory to the distributor, from the distributor to the wholesaler, and from the wholesaler to the retailer, respectively; $\delta_{p,prod}$, $\delta_{p,dc}$, $\delta_{dc,w}$, and $\delta_{w,r}$ are the corresponding production and shipment delays; O^{dc}_{t}, O^{w}_{t}, and O^{r}_{t} are the quantities of order placed by corresponding agent; SA^{r}_{t} is the sales quantity of the retailer; D^{r}_{t} is the customer demand.

In order to transform Equation (3.29) into the state-space form Equation (3.15), one has to deal with the lagged terms. A variable lifting technique is used to address this problem (Li and Marlin, 2009; Mastragostino et al., 2014). Additional state variables (called auxiliary variables) are introduced to model the delays in the system. Consider modelling a lagged term $Q_{t-\delta}$, which has a delay of δ periods. In order to model the delay, δ auxiliary variables

are introduced for each time period t, and the relationship is expressed as Equation (3.30).

$$\begin{bmatrix} Q_{t+1,1} \\ Q_{t+1,2} \\ \vdots \\ Q_{t+1,\delta-1} \\ Q_{t+1,\delta} \end{bmatrix} = \begin{bmatrix} Q_{t,2} \\ Q_{t,3} \\ \vdots \\ Q_{t,\delta} \\ Q_{t} \end{bmatrix}, \quad \forall t \tag{3.30}$$

By replacing t with $t-\delta$ in the last row of Equation (3.30), we have

$$Q_{t-\delta+1,\delta} = Q_{t-\delta} \tag{3.31}$$

Similarly, in the last but one row, by replacing t with $t-\delta+1$, we have $Q_{t-\delta+2,\delta-1} = Q_{t-\delta+1,\delta}$. More generally, in $Q_{t+1,k} = Q_{t,k+1}$, by replacing t with $t-k$, we have:

$$Q_{t-k+1,k} = Q_{t-k,k+1}, \quad \text{for} \quad k = 1, 2, \ldots, \delta-1 \tag{3.32}$$

Equation (3.32) is equivalent to:

$$Q_{t,1} = Q_{t-1,2} = Q_{t-2,3} = \cdots = Q_{t-\delta+2,\delta-1} = Q_{t-\delta+1,\delta} \tag{3.33}$$

Hence, based on Equation (3.31) and Equation (3.33), we have:

$$Q_{t-\delta} = Q_{t,1} \tag{3.34}$$

Equation (3.34) transforms a lagged term $Q_{t-\delta}$ to a state variable at time period t, $Q_{t,1}$. For example, by replacing the lagged term $Q^{p,prod}_{t-\delta_{p,prod}}$ with the auxiliary state $Q^{p,prod}_{t,1}$ and incorporating additional equations similar to Equation (3.30), the equation $I^{p}_{t+1} = I^{p}_{t} -$

$Q_t^{p,dc} + Q_{t-\delta_{p,prod}}^{p,prod}$ can be transformed into the state-space form as follows:

$$\begin{aligned} I_{t+1}^{p} &= I_t^p - Q_t^{p,dc} + Q_{t,1}^{p,prod}, && \forall t \in T \\ Q_{t+1,k}^{p,prod} &= Q_{t,k+1}^{p,prod}, && \forall t \in T, k = 1,2,\ldots,\delta_p - 1 \\ Q_{t+1,\delta_{p,prod}}^{p,prod} &= Q_t^{p,prod}, && \forall t \in T \end{aligned} \tag{3.35}$$

The other lagged terms can be modelled in a similar way. Then, the supply chain system (3.29) can be transformed into the following state-space form:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + v(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.36}$$

with the state variables $x(t)$, control inputs $u(t)$, disturbance $d(t)$, and outputs $y(t)$ defined as follows:

$$\begin{aligned} x(t) =& [I_t^p, B_t^p, I_t^{dc}, B_t^{dc}, I_t^w, B_t^w, I_t^r, B_t^r, \\ & Q_{t,1}^{p,prod}, \ldots, Q_{t,\delta_{p,prod}}^{p,prod}, Q_{t,1}^{p,dc}, \ldots, Q_{t,\delta_{p,dc}}^{p,dc}, Q_{t,1}^{dc,w}, \ldots, Q_{t,\delta_{dc,w}}^{dc,w}, Q_{t,1}^{w,r}, \ldots, Q_{t,\delta_{w,r}}^{w,r}]^T \\ & \in \mathbb{R}^{8+\delta_{p,prod}+\delta_{p,dc}+\delta_{dc,w}+\delta_{w,r}} \\ u(t) =& [Q_t^{p,prod}, Q_t^{p,dc}, Q_t^{dc,w}, Q_t^{w,r}, O_t^{dc}, O_t^w, O_t^r, SA_t^r]^T \in \mathbb{R}^8 \\ v(t) =& [0,0,\ldots,0,D_t^r,0,\ldots,0]^T \in \mathbb{R}^{8+\delta_{p,prod}+\delta_{p,dc}+\delta_{dc,w}+\delta_{w,r}} \\ y(t) =& [I_t^p, B_t^p, I_t^{dc}, B_t^{dc}, I_t^w, B_t^w, I_t^r, B_t^r]^T \in \mathbb{R}^8 \end{aligned} \tag{3.37}$$

The state variable $x(t)$ is defined to comprise of the stock on hand and backorders of the supply chain agents, and also the auxiliary variables. The control input $u(t)$ is defined to comprise of the quantities of production, shipment, sales, and order quantities. The customer demand is considered as the disturbance $v(t)$ to the system. The output $y(t)$ is defined to comprise of the stock on hand and backorders. Since CVA assumes the disturbance $v(t)$ to be white noise, the demand needs to be included in the input vector $u(t)$ if it is autocorrelated.

Consider a general supply chain, in which the material balances can be modelled as follows:

$$\begin{aligned} I_{t+1}^{a,m} &= I_t^{a,m} - \sum_{i \in ADJ(a)+a} Q_t^{a,i,m} + \sum_{j \in ADJ(a)+a} Q_{t-\delta_{j,a}}^{j,a,m}, \qquad \forall a \in A, m \in M, t \in T \\ B_{t+1}^{a,m} &= B_t^{a,m} - \sum_{i \in ADJ(a)} Q_t^{a,i,m} + \sum_{i \in ADJ(a)} O_t^{i,a,m} + D_t^{a,m}, \qquad \forall a \in A, m \in M, t \in T \end{aligned} \tag{3.38}$$

where $I_t^{a,m}$ and $B_t^{a,m}$ are the stock on hand and backorder of material m of agent a at the beginning of time period t, respectively; $ADJ(a)$ is the set of adjacent nodes of a; $Q_t^{a,i,m}$ represents the transportation quantity of m from a to i at period t; similarly, $Q_t^{j,a,m}$ represents the transportation quantity of m from j to a at period t; especially, $Q_t^{a,a,m}$ represents the quantity of m produced or consumed at a at period t; $O_t^{i,a,m}$ is the order quantity of m placed to a by i; $\delta_{j,a}$ is the production or shipment delays; A and M are the sets of supply chain agents and materials, respectively; $D_t^{a,m}$ is the demand of m at a at period t.

Consider modelling a lagged term $Q_{t-\delta_{j,a}}^{j,a,m}$, which has a delay of $\delta_{j,a}$ periods. In order to model the delay, $\delta_{j,a}$ auxiliary variables, $Q_{t,i}^{j,a,m}$, $i = 1, 2, \ldots, \delta_{j,a}$, are introduced for each time period t in a similar manner as described earlier. By replacing the lagged term $Q_{t-\delta_{j,a}}^{j,a,m}$ and adding the extra equations, the system (3.38) can be rewritten as:

$$\begin{aligned} I_{t+1}^{a,m} &= I_t^{a,m} - \sum_{i \in ADJ(a)+a} Q_t^{a,i,m} + \sum_{j \in ADJ(a)+a} Q_{t,1}^{j,a,m}, && \forall a, m, t \\ B_{t+1}^{a,m} &= B_t^{a,m} - \sum_{i \in ADJ(a)} Q_t^{a,i,m} + \sum_{i \in ADJ(a)} O_t^{i,a,m} + D_t^{a,m}, && \forall a, m, t \\ Q_{t+1,k}^{j,a,m} &= Q_{t,k+1}^{j,a,m}, && \forall a, m, t, k = 1, 2, \ldots, \delta_{j,a} - 1 \\ Q_{t+1,\delta_{j,a}}^{j,a,m} &= Q_t^{j,a,m}, && \forall a, m, t \end{aligned} \tag{3.39}$$

Therefore, the system (3.38) can be transformed into the state-space form (3.36), with the state

variables $x(t)$, control inputs $u(t)$, disturbance $v(t)$, and outputs $y(t)$ defined as follows:

$$\begin{aligned} x(t) &= \left[\left(I_t^{a,m} \forall a,m\right), \left(B_t^{a,m} \forall a,m\right), \left(Q_{t,k}^{j,a,m} \forall a,j,m,k\right) \right]^T \\ u(t) &= \left[\left(Q_t^{a,i,m} \forall a,i,m\right), \left(O_t^{i,a,m} \forall a,i,m\right) \right]^T \\ v(t) &= \left[\left(D_t^{a,m} \forall a,m\right) \right]^T \\ y(t) &= \left[\left(I_t^{a,m} \forall a,m\right), \left(B_t^{a,m} \forall a,m\right) \right]^T \end{aligned} \tag{3.40}$$

For a general supply chain, the state variable $x(t)$ is defined to comprise of the stock on hand and backorder of the supply chain agents, and also the auxiliary variables. The control input $u(t)$ is defined to comprise of the quantities of purchase, production, shipment, sale, and order. The output $y(t)$ is defined to comprise of the stock on hand and backorders. The customer demand can be considered as the disturbance $v(t)$ to the system or included in the input vector.

The above analysis indicates that a general supply chain system can be represented by a state-space model. This is the foundation for using CVA to reduce the dimensionality and model the supply chain. Based on the definition in Equation (3.40), the supply chain data can be collected and organized according to Equation (3.16) and Equation (3.17). Then, a CVA-based SCMo model can be developed.

3.3.3 SCVA hyperparameter tuning

It is worth mentioning that based on the state-space model of supply chain, the covariance matrices of supply chain data may be singular. For example, if in Equation (3.40), a backorder variable $B_t^{a,m} = 0, \forall t$ under the NOC, then the covariance matrices Σ_{xx} and Σ_{yy} would be singular. In this case, the SVD (3.11) cannot be computed, and the classical CVA method cannot be applied directly. Therefore, in this chapter, the SCVA method is adopted.

Before implementing Algorithm 3.1 to develop a SCVA model, there are five hyperparameters that need to be specified, which are the lag orders for the past and future vectors, l and

h, respectively; the state order d; and the sparsity parameters c_1 and c_2. Hyperparameter tuning is important to the SCVA algorithm, as the choice of hyperparameters can have an impact on the performance of the SCVA model.

In the current literature, there are a few ways used for hyperparameter tuning. The lag orders l and h can be determined by using an ARX model and the Akaike information criterion (AIC) (Russell et al., 2000a; Russell et al., 2000b; Jiang et al., 2015a; Jiang et al., 2015c; Jiang et al., 2015b; Lu et al., 2018b; Lu et al., 2018a), or by checking the autocorrelation in the sum of squares of all measurements (Ruiz-Cárcel et al., 2015; Ruiz-Cárcel et al., 2016). AIC is a metric that rewards the goodness of fit of a model, but penalizes an increase in the number of model parameters. This study employs the autocorrelation plot to determine the lag orders. It is common to consider $l = h$ (Larimore, 1990; Larimore, 1997; Odiowei and Cao, 2010; Ruiz-Cárcel et al., 2015; Ruiz-Cárcel et al., 2016; Lu et al., 2018b; Lu et al., 2018a), which is what we adopt in this chapter. For classical CVA, the state order d can be determined by minimizing the AIC (Larimore, 1997; Russell et al., 2000a; Burnham and Anderson, 2004), or according to the dominant singular values (Odiowei and Cao, 2010). However, since the SCVA algorithm is not a SVD problem, the dimension of canonical states cannot be determined according to dominant singular values. In this chapter, the mean squared error (MSE) is chosen as the selection criterion (Witten et al., 2009).

The sparsity parameters c_1 and c_2 should be selected within $[1, \sqrt{n_p}]$ and $[1, \sqrt{n_f}]$, respectively, in order to maintain the feature of CVA and meanwhile let the l_1-norm constraints be active to enforce sparsity (Witten et al., 2009; Lu et al., 2018b). In the current literature, c_1 and c_2 are determined independently of the state order d (Witten et al., 2009), or after d is selected (Lu et al., 2018b). In this chapter, we consider the combined tuning of the 3 hyperparameters. The procedure for tuning parameters is based on a grid search, and is presented in Algorithm 3.2. This procedure is based on the method presented by Witten et al. (2009) and is similar to cross-validation. First, for simplicity, the two sparsity parameters are reduced to one. This can be done by setting $c_1 = c\sqrt{n_p}$ and $c_2 = c\sqrt{n_f}$ (Witten et al., 2009), or by setting $c_1 = c_2 = c\sqrt{n_p}$ (Lu et al., 2018b). Here, the first approach is used. This allows the sparsity to be represented using single parameter c, and the canonical vectors α and β to be equally sparse. Next, the subset of hyperparameter space, i.e., the set of candidate

(d, c) pairs, are specified. Then the optimal (d, c) pair is determined by minimizing the MSE through a grid search, as this is a relatively common practice in machine learning. By tuning d and c as a pair, the procedure helps understand the impact of them on the SCVA model. In Algorithm 3.2, K is a parameter that is specified by the user, and can be selected to be, e.g., 10 (Witten et al., 2009).

Algorithm 3.2: SCVA hyperparameter tuning (d, c)

Data: P, F
Result: $(d, c)^\star$
Parameters: K

$Z \leftarrow P^T F$
Partition Z into K non-overlapping matrices Z_k, $k = 1, 2, \ldots, K$, with Z_k containing $1/K$ of the rows of Z
Specify the candidate sets $\mathcal{D}$ for state order d, and $\mathcal{C}$ for sparsity parameter c
for *each* $d \in \mathcal{D}$ **do**
  for *each* $c \in \mathcal{C}$ **do**
    for $k = 1, 2, \ldots, K$ **do**
      Fit the penalized decomposition with (d, c) to $\bar{Z}_k$, i.e., Z with Z_k removed
      Calculate the residual $\bar{Z}_k^{res} = \bar{Z}_k - \sum_{n=1}^{d} \gamma_n \alpha_n \beta_n^T$
      Calculate the mean squared error: $MSE(d, c, k) = mean(\sum_{z \in \bar{Z}_k^{res}} z^2)$
    end
    Calculate the average MSE: $AMSE(d, c) = \sum_{k=1}^{K} MSE(d, c, k)/K$
  end
end
Select $(d, c)^\star$ that minimizes the AMSE: $(d, c)^\star = \arg\min_{d,c} AMSE(d, c)$

3.4 Case Studies

Two case studies are carried out to validate the CVA-based SCMo method. The first case is an illustrative example based on the beer game supply chain, and the second case involves a packaged liquified gas supply chain that is more complicated. Studies on the two cases are presented in Section 3.4.1 and 3.4.2, respectively. Results of PCA-based SCMo are also presented for comparative study.

3.4.1 Case study I: The beer distribution game

The beer game supply chain, which has been shown in Figure 3.1 and analyzed in Section 3.3.2, is used in the first case study to demonstrate the CVA-based SCMo method. First, a simulation model for the beer game supply chain is developed to generate simulated data for analysis. The simulation model is developed using the programming language `Python 3.8` and the agent-based modelling (ABM) technique. ABM is a well-suited paradigm for the modelling and simulation of distributed systems such as supply chain system. Within the ABM framework, the components of a supply chain, such as the retailer, wholesaler, distributor and factory, are modelled as autonomous agents that are able to communicate with other agents and make decisions. For example, an agent Retailer is defined to mimic the behaviour of a retailer, and the sequence of actions of the Retailer for each time period is given as follows:

1. Places an order to the upstream distributor agent. The order quantity is determined by an inventory policy.
2. Receives the products that were shipped from the distributor and arrives in this period after a transportation delay.
3. Satisfies the backorders and demand of the customer agent. If the current stock on hand is sufficient, then products are shipped to fill the orders immediately; otherwise the unfilled part is recorded as backorders and will be processed prior to the demand once sufficient stock is available.
4. Updates the inventory and backorders profiles.

The wholesaler, distributor and factory agents are developed in a similar way. The simulation model developed in this work is similar to that has been used by Wang et al. (2020), where the details of the model are given and a causal diagram is presented to show the interaction between the agents.

Two inventory policies are investigated for the beer game supply chain. The first one is

a continuous review $(S-1, S)$ policy, which is also known as the "one for one" policy (Chaharsooghi et al., 2008; Axsäter, 2015). With this policy, a supply chain agent reviews the inventory continuously, and orders to its upstream whatever is ordered by the downstream. The second one is a periodic review order-up-to (OUT) (R, S) policy, where R is the review period, and S is the OUT level (Chatfield et al., 2004). With this policy, a supply chain agent reviews the inventory periodically, and orders to its upstream to maintain a target inventory position S. It is assumed that all the supply chain agents manage their inventory according to the same policy.

The SCMo model is able to monitor the supply chain under uncertainty. Therefore, the simulation model mimics the supply chain operation under uncertain customer demand. Two demand models are considered in order to investigate the impact of the autocorrelation in demand on supply chain dynamics. The first demand model is the normal distribution (Gupta and Maranas, 2003; Chatfield et al., 2004). It is assumed that the demand follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. In the case study, the demand is assumed to follow $\mathcal{N}(50, 10^2)$, and is sampled from this distribution by the application programming interface (API) `numpy.random.normal()` provided by the `numpy` library for `Python`. The samples are rounded to integer, and negative samples are set to zero to avoid negative demand. It is observed that customer demand generated from the normal distribution does not have significant autocorrelation with previous observations.

The second model used for demand modelling is a time series model, the AutoRegressive Integrated Moving Average (ARIMA) model (Box et al., 2015). An ARIMA(p, d, q) model for modelling a time series $x_1, x_2, \ldots, x_t, \ldots$ can be expressed as the following difference equation form:

$$\nabla^d x_t = \sum_{i=1}^{p} \phi_i \nabla^d x_{t-i} - \sum_{j=1}^{q} \theta_j a_{t-j} + a_t \tag{3.41}$$

alternatively:

$$x_t = \sum_{i=1}^{p+d} \varphi_i x_{t-i} - \sum_{j=1}^{q} \theta_j a_{t-j} + a_t \tag{3.42}$$

where ∇^d is the dth order (backward) difference, p is the order of the autoregressive part, q

is the order of the moving average part, and a_t is the forecasting error in period t, which is assumed to follow a normal distribution with zero mean; ϕ_i, θ_j and φ_i are the weight parameters.

Equations (3.41) and (3.42) show that in an ARIMA model, the current value (or differenced value) can be expressed as a linear combination of it lagged observations and forecasting errors. A normal distribution is essentially corresponding to an ARIMA$(0,0,0)$ model, where x_t is irrelevant to the previous data. It is observed that demand generated from the ARIMA model is usually autocorrelated with lagged observations, except for ARIMA$(0,0,0)$. To capture the seasonality in demand, a seasonal ARIMA (SARIMA) model can be used, which is written as ARIMA$(p,d,q) \times (P,D,Q)_s$. The (p,d,q) models the non-seasonal part, (P,D,Q) models the seasonal part, and parameter s models the seasonal pattern. ARIMA and SARIMA models have been adopted in demand modelling in the literature (Aburto and Weber, 2007; Jaipuria and Mahapatra, 2014; Mastragostino et al., 2014). Gilbert (2005) shows that if the customer demand and lead times at each echelon follow ARIMA models, then the orders and inventories at each echelon are also ARIMA when an OUT policy is adopted. This implies that a supply chain system operated under autocorrelated demand may also have an autocorrelated structure. This is investigated in this section.

In order to generate a ARIMA model that simulates real-world demand pattern, the Iowa Liquor Sales dataset[1] is used to train the model. This dataset is published as open data by the Iowa Department of Commerce, Alcoholic Beverages Division. It contains every wholesale purchase of liquor in the State of Iowa by retailers for sale to individuals since January 1, 2012, and is updated monthly. It is a good resource for analyzing real-world customer demand of individual products at the store level, and thus is used in this case study to train the ARIMA demand model. The daily sales data of Canadian Whiskies are extracted from the dataset to fit a ARIMA$(p,d,q) \times (P,D,Q)_s$ model. The model is built and trained by using `pmdarima`, which is a statistical library designed for time series analysis in `Python`. The seasonality parameter s is determined as 5 according to the autocorrelation plot of data. The other parameters are tuned automatically using the `pmdarima.auto_arima()` API. An ARIMA$(4,0,0) \times (0,0,2)_5$ model is obtained for demand simulation. Demand scenarios are

[1] https://data.iowa.gov/Sales-Distribution/Iowa-Liquor-Sales/m3tr-qhgy

simulated by using the model and adding a disturbance at each time step. The disturbance is generated from a normal distribution with zero mean. Results show that the simulated time series basically maintains the autocorrelation of the original data.

The supply chain simulator and the SCMo models are developed in the programming language `Python`. The supply chain operation is simulated with the demand models and inventory policies. The normal transportation delays between each two adjacent agents are set to 2 time periods. According to the state-space model developed in Section 3.3, a total of 17 variables are collected to build the data-driven SCMo model, which are 9 input variables, including the quantities of production, shipment, order, sale, and demand; and 8 output variables, including the stock on hand and backlog of the 4 agents. For the normal distribution demand model, a total of 1000 demand samples are generated from $\mathcal{N}(50, 10^2)$, and the supply chain operation is simulated over 1000 time periods. Thus, 1000 samples of supply chain data are collected to train the SCVA model. For the ARIMA demand, 1090 demand samples are extracted from the dataset, and therefore 1090 samples of supply chain operation data are obtained.

For NOC under the normal distributed demand and the 1-1 policy, the time lag in supply chain data are determined as $l = h = 2$ according to the autocorrelation plot, which is shown in Figure 3.2. Therefore, the observation at each time period is augmented into $17 \times 2 = 34$ variables. Then the NOC is characterized by SCVA to construct the SCMo model. In order to determine the hyperparameter of the SCVA model, the MSEs using different hyperparameter combinations are calculated and visualized in Figure 3.3. In Figure 3.3, the horizontal and vertical axes represent the state order d and the sparsity parameter value c, respectively. The MSE values are plotted as bubbles. The size and color of a bubble indicate the average MSE of the model under a pair of (d, c), and a smaller size indicates a smaller MSE value. It is observed that with 1 canonical state ($d = 1$), the MSE is relatively large when $c < 0.6$, and is small when $c \geq 0.67$. With 2 or more canonical states $d \geq 2$, the MSE is very small. This suggests that the supply chain data can be explained using 1 canonical state. Therefore, a state order $d = 1$ and a sparsity parameter $c = 0.8$ is determined as a good hyperparameter combination, because it results in a low MSE and a small number of states. Thus, the 34 supply chain variables are reduced to 1 canonical state through dimensionality reduction by

SCVA.

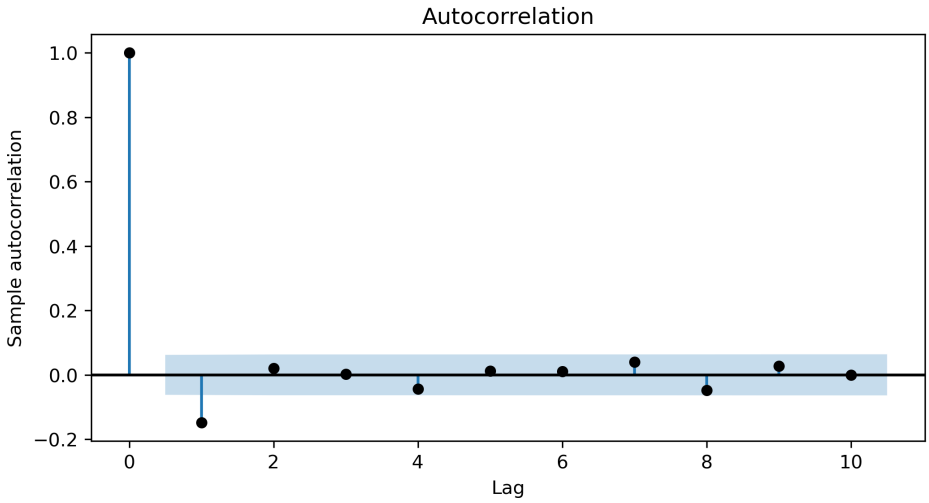


Figure 3.2. Case study I: normal demand, autocorrelation plot.

A SCVA model is then trained with the selected hyperparameters. In order to examine the performance of the monitoring model under NOC, 200 testing scenarios are simulated and the average false alarm rate (FAR) is calculated. FAR, also know as the Type I error rate or false positive rate, is calculated as the ratio of false alarms relative to the total samples. The average FAR of 200 testing scenarios is 2%, which is a satisfactory level that indicates good performance under NOC. Similarly, the CVA-based SCMo models for ARIMA demand and (R, S) policy are also built to investigate their performance under different conditions. For the (R, S) policy, the review period R is set to 1, and the OUT level S is set to 300 for all agents.

For comparison, a DPCA model that has been shown to be effective for SCMo (Wang et al., 2020) is also developed in this work. For the DPCA model, the hyperparameter that needs to be specified before training is the number of principal components (PCs), N_{PC}. N_{PC} is determined through a 10-fold cross-validation based on two metrics R^2 and Q^2, which are the explained variance ratio of the training set and testing set, respectively. The R^2 and Q^2 are shown in the bar chart Figure 3.4. The first PC has a R^2 and a Q^2 of approximately 55%. The first 2 PCs together have a R^2 and a Q^2 of approximately 93%. And the first 3 PCs together have a R^2 and a Q^2 over 99.9%. Therefore, $N_{PC} = 2$ is a good option, and the 17 supply chain variables are reduced to 2 PCs by DPCA.

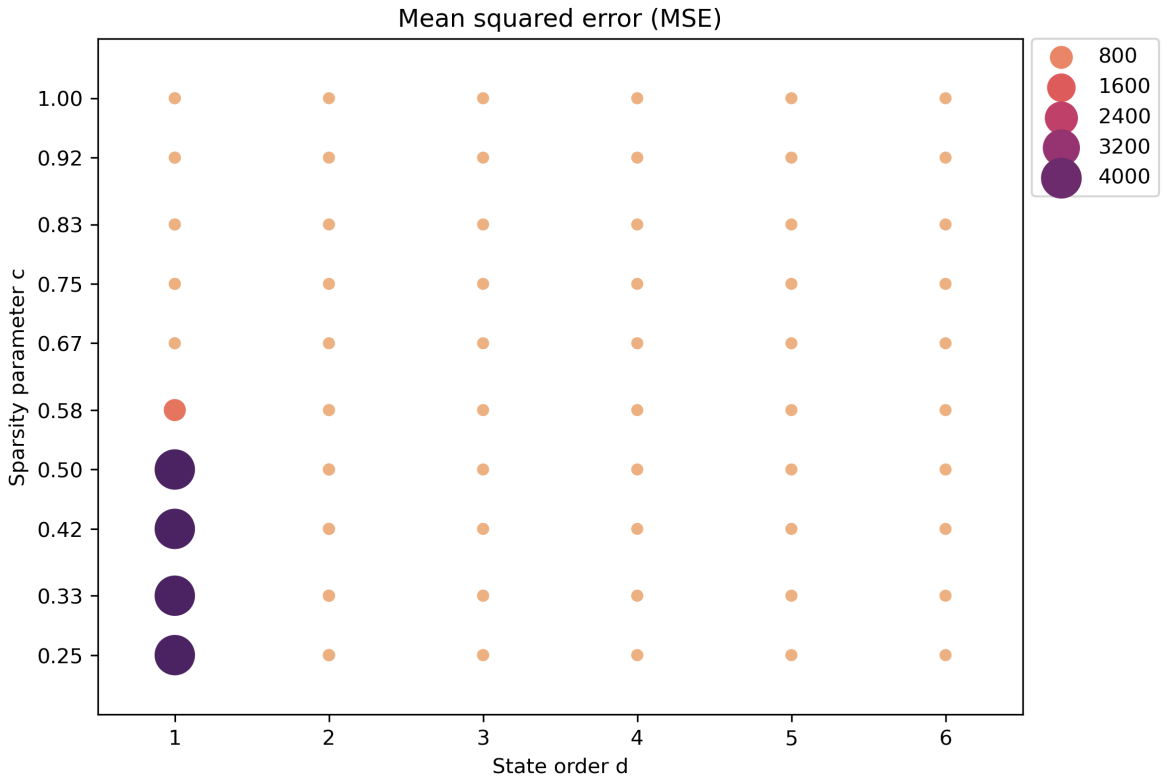


Figure 3.3. Case study I: normal demand, SCVA MSE.

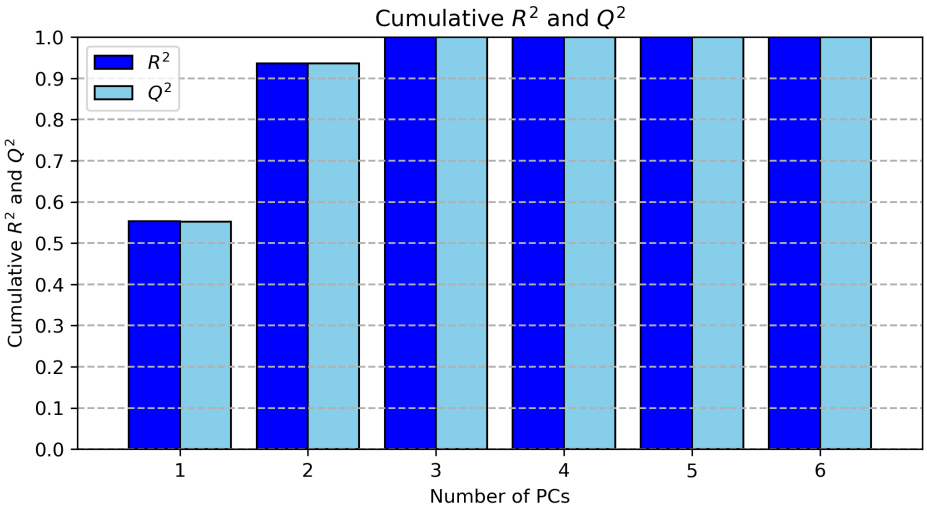


Figure 3.4. Case study I: normal demand, DPCA R^2, Q^2.

The results of dimensionality reduction and FAR using the SCVA and DPCA models are summarized in Table 3.1. First, it is observed that with a SARIMA demand model, the lag orders l and h determined according to the autocorrelation plot are larger than those with a normal demand model. This means stronger autocorrelation in demand also leads to stronger autocorrelation in the supply chain system. In general, with SARIMA demand, more latent variables are needed to model the supply chain data for SCVA and DPCA, except for the (R, S) policy, where SCVA uses fewer canonical states and shows its ability in modelling autocorrelated data. Moreover, the (R, S) policy results in stronger autocorrelation than the 1-1 policy. For different demand models and inventory policies, SCVA identifies a lower system order, which means it requires a smaller number of states than DPCA to model the supply chain. For SCVA, the FARs of T_d^2, Q, and $S_{overall}$ are less than 8% for all situations. For DPCA, the largest FAR is 10.6%, which is $S_{overall}$ under SARIMA demand and (R, S) policy. For the SARIMA demand, $S_{overall}$ of SCVA has a lower FAR than $S_{overall}$ of DPCA. In general, for both models, the FARs of the 3 monitoring statistics are at a low level. The difference of FAR between SCVA and DPCA is not significant. Both of them characterize the NOC well. Furthermore, for normal demand, the FARs are lower than SARIMA demand.

Table 3.1. Case study I: average FAR.

Demand	Policy	Lag l, h	SCVA				DPCA			
			N_{CS}	T_d^2	Q	$S_{overall}$	N_{PC}	T^2	SPE	$S_{overall}$
Normal	1-1	2	1	0.013	0.013	0.026	2	0.014	0.01	0.024
Normal	(R, S)	4	4	0.013	0.013	0.026	6	0.013	0.012	0.025
SARIMA	1-1	8	3	0.036	0.036	0.069	5	0.068	0.01	0.076
SARIMA	(R, S)	11	3	0.029	0.053	0.078	8	0.093	0.015	0.106

In order to comprehensively assess the performance of SCVA in SCMo, different supply chain faults are designed, simulated and analyzed, which include:

(1) Delay in the transportation between the wholesaler and retailer. The normal transportation time is 2 time periods, while a delayed transportation time is set to 4. This fault may cause stockout of the retailer.

(2) Delay in the transportation between the distributor and wholesaler. The normal trans-

portation time is 2 time periods, while a delayed transportation time is set to 4. This fault may affect the inventory of wholesaler, and retailer subsequently.

(3) Delay in the transportation between the factory and distributor. This fault may affect the inventory of distributor, wholesaler and retailer.

(4) Poor yield of the factory. Only 50% of the production order can be fulfilled. This fault may affect the inventory of all the agents.

(5) An increase in the customer demand. The demand is multiplied by 2. This may cause stockout of the retailer.

(6) Delays in the transportation between the distributor and wholesaler, and the transportation between the wholesaler and retailer. The delayed transportation times are set to 3 time periods.

(7) Poor yield of factory and transportation delay between factory and distributor. Only 60% of the production order can be fulfilled, and the delayed transportation time is set to 3 time periods.

Faults (1-5) are designed to validate FDD of abnormal events occurring at individual agents, and faults (6-7) are designed to test abnormal events at more than one agent. The duration of the faults are set to 4 time periods. An example of FDD using SCVA-based monitoring charts is shown in Figure 3.5, with the monitoring statistics plotted on the top and a 2-D contribution plot given on the bottom. The simulated supply chain fault is fault (1): during time period 49-53, the Retailer experiences a longer transportation delay (4 periods) for its orders. The normal delay is 2 periods. In the monitoring chart, T_d^2 and Q are plotted over time, with the axis shown on the left and right, respectively. Their 99% confidence limits are aligned for better visualization. It can be seen that when the supply chain is operated under NOC, the monitoring statistics are below their limits. When the fault occurs, the Q statistic of SCVA model exceeds its 99% limit. This implies the fault has been detected by Q and an alarm has been raised by the SCMo model. In the contribution plot shown on the bottom, the combined contribution of each variable to the monitoring statistics are plotted over time

in the form of a heatmap (Zhu and Braatz, 2014). In the 2-D contribution plot, a deeper color indicates a larger contribution. It is observed that during the faulty period, the Retailer's inventory I_t^r contributes the most to the variation of the statistics, thus it is identified as the fault-related variable. This implies the fault has affected the Retailer's inventory, and the variation of supply chain status is caused by an unusual change in the Retailer's inventory. Furthermore, during the faulty period, the contribution of the backorder B_t^r becomes larger than usual for several periods, as can be observed in Figure 3.5. This indicates the fault may cause a shortage at the Retailer. After the faulty period ends, the monitoring statistics and variable contributions return to normal.

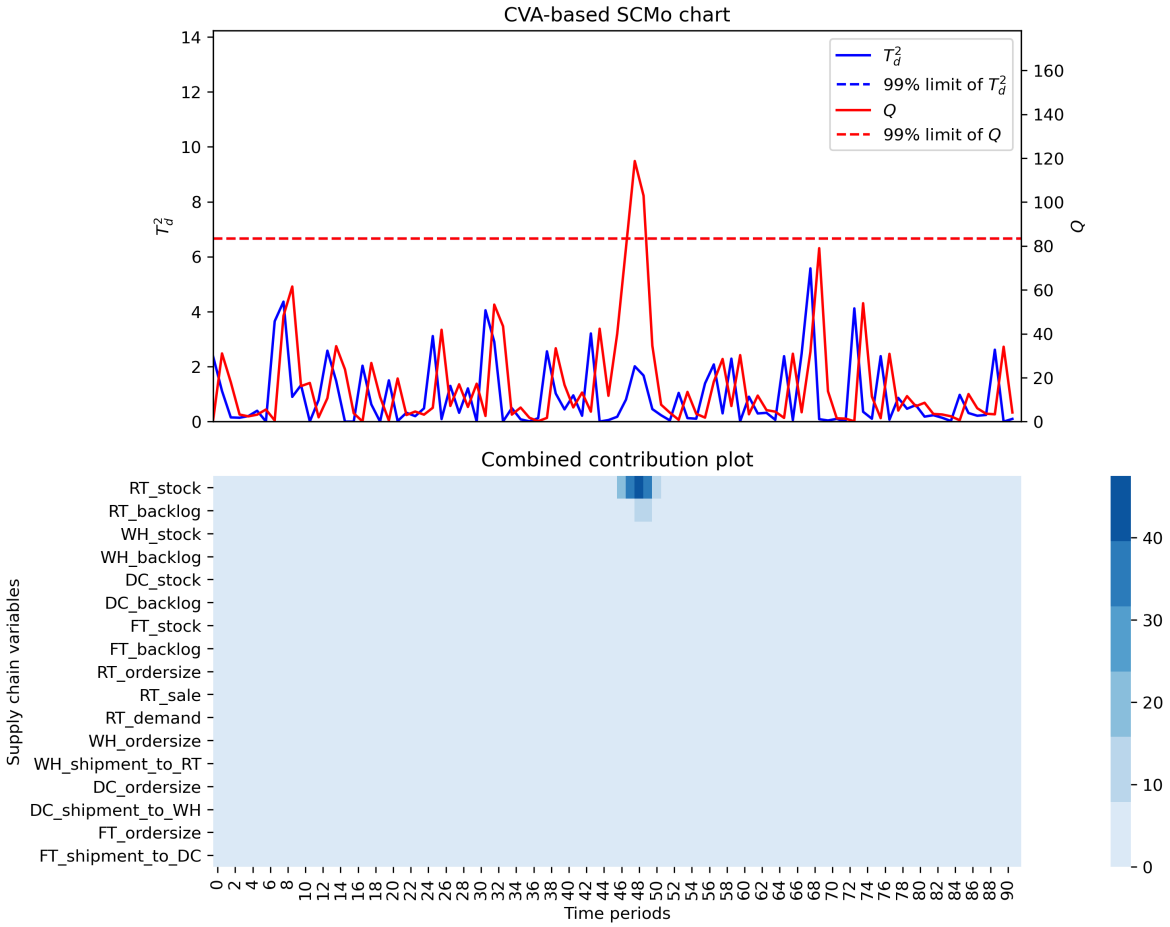


Figure 3.5. Case study I: CVA-based SCMo chart.

The SCVA-based SCMo model is tested with the two demand models and two inventory policies. For each fault, 200 demand scenarios are used in simulation and analysis. The

performance of FDD are assessed using two metrics, the missed detection rate (MDR) and detection delay. MDR, also known as the Type II error rate or false negative rate, is defined as the ratio of undetected faulty samples relative to the total number of faulty samples under a specific fault. The detection delay under a scenario is calculated as the time needed to detect a fault after it occurs. These three metrics, FAR, MDR and detection delay, are commonly used to assess SPM methods (Russell et al., 2000b; Lu et al., 2018b; Ruiz-Cárcel et al., 2015; Ruiz-Cárcel et al., 2016). In this work, the ability of a SCMo model in dimensionality reduction is also considered. It is associated to the number of canonical states that are determined to model the system for SCVA, and the number of PCs needed for DPCA.

The MDRs of SCVA and DPCA under different demand models and inventory policies are presented in Table 3.2. For SCVA, the results of 3 metrics, T_d^2, Q and $S_{overall}$, are presented. As defined in Equation (3.26), $S_{overall}$ raises an alarm when at least one of T_d^2 and Q exceeds its confidence limit. For DPCA, the results of T_2, SPE and $S_{overall}$ are presented. It is observed that T_d^2 by SCVA and T^2 by DPCA show high MDRs for the faults. For both demand models and inventory policies, the MDRs of these two statistics are greater than 70% for most of the faults, which is a high level. By contrast, Q by SCVA and SPE by DPCA have low MDRs. For most of the faults, their MDRs are less than 10%. This implies that the supply chain faults usually result in deviations from NOC in the residual space rather than in the state or score space. The monitoring statistics that quantify variations in the residual space are thus found to be more reliable for both SCVA and DPCA in SCMo. This result is consistent with the results of Russell et al. (2000b), where the statistics in the residual space are shown to be generally more sensitive to the faults than the statistics in the state or score space. The performance of $S_{overall}$ is close to the residual statistics for both models. This is because the fault is usually detected by the residual statistics, rather than T_d^2 and T^2.

The average detection delays of SCVA and DPCA are presented in Table 3.3. For a certain fault, the average detection delay is calculated as an average of the detection delay over all the test scenarios. Since T_d^2 by SCVA and T^2 by DPCA have high MDRs, the detection delays of them do not provide much useful information. Therefore, we focus on the detection delays of the residual statistics. It is observed that for some faults, the SPE by DPCA has a slightly smaller detection delay than Q by SCVA, while for other faults, SCVA Q has a

smaller detection delay. In general, the difference is insignificant. For the considered faults, SCVA is able to raise an alarm at the same time with SCVA, or at most 1 time period later. Considering that SCVA uses significantly fewer states than DPCA, the performance of SCVA is satisfactory.

In summary, SCVA and DPCA have comparable performance in general for the beer game supply chain. Both SCMo models rely on the statistic in the residual space. In terms of dimensionality reduction, SCVA is more efficient in modelling supply chain data. Compared with cross-correlated demand, autocorrelated demand requires more states or components for modelling. In both cases, SCVA determines a smaller system order than DPCA, and achieves similar performance to DPCA in terms of FAR, MDR and detection delay with significantly fewer states.

3.4.2 Case study II: A packaged liquefied gas supply chain

This case study is based on a packaged liquefied gas supply chain that has been investigated by Misra et al. (2019) and Wang et al. (2020). The configuration of this supply chain network is shown in Figure 3.6. This supply chain consists of 2 plants, 1 warehouse, and 5 customer locations. It deals with the filling and distribution of gas products to satisfy customers' needs. The gas products are stored in stock keeping units (SKUs). The customers consume filled SKUs, which are turned into empty SKUs. The warehouse holds inventory of filled and empty SKUs. It transports filled SKUs to customer locations, and retrieves empty SKUs. Empty SKUs are then transported to the upstream plants for refilling. Refilled products are sent back to warehouse. This means there are two directions of material flow: the flow of filled SKUs from upstream plants to the downstream customers, and the flow of empty SKUs in the opposite direction. This supply chain network is more complicated than the serial beer game supply chain. There are 2 gas products transported across the supply chain, A and B. The transportation time between Plants and Warehouse is set to 3 time periods, and the transportation time between the Warehouse and Customers is set to 1 time period. The refilling time of the plant is 1 period. It is assumed that a vendor managed inventory (VMI) policy is adopted. When a customer's inventory falls below a certain level s, filled SKUs are

Table 3.2. Case study I: MDR.

Demand	Policy	Fault	SCVA			DPCA		
			T_d^2	Q	$S_{overall}$	T^2	SPE	$S_{overall}$
Normal	1-1	1	0.96	0	0	0.95	0	0
Normal	1-1	2	0.94	0	0	0.94	0	0
Normal	1-1	3	0.94	0	0	0.94	0	0
Normal	1-1	4	0.94	0	0	0.93	0	0
Normal	1-1	5	0	0	0	0	0.01	0
Normal	1-1	6	0.95	0	0	0.94	0	0
Normal	1-1	7	0.94	0	0	0.93	0	0
Normal	(R, S)	1	0.92	0.05	0.05	0.97	0	0
Normal	(R, S)	2	0.97	0.02	0.02	0.97	0	0
Normal	(R, S)	3	0.94	0.02	0.02	0.95	0	0
Normal	(R, S)	4	0.89	0.41	0.40	0.91	0.02	0.02
Normal	(R, S)	5	0.01	0.04	0.01	0.01	0.01	0.01
Normal	(R, S)	6	0.97	0.4	0.4	0.97	0.01	0.01
Normal	(R, S)	7	0.9	0.43	0.43	0.92	0	0
SARIMA	1-1	1	0.81	0.02	0.01	0.73	0.03	0.02
SARIMA	1-1	2	0.8	0.02	0.01	0.74	0.03	0.02
SARIMA	1-1	3	0.8	0.02	0.01	0.74	0.03	0.02
SARIMA	1-1	4	0.82	0.05	0.05	0.74	0	0
SARIMA	1-1	5	0	0	0	0	0	0
SARIMA	1-1	6	0.75	0.35	0.33	0.72	0.22	0.20
SARIMA	1-1	7	0.77	0.08	0.07	0.71	0.02	0.02
SARIMA	(R, S)	1	0.82	0.06	0.05	0.75	0	0
SARIMA	(R, S)	2	0.79	0.07	0.07	0.7	0	0
SARIMA	(R, S)	3	0.87	0.05	0.05	0.75	0	0
SARIMA	(R, S)	4	0.87	0.03	0.03	0.71	0	0
SARIMA	(R, S)	5	0	0	0	0	0	0
SARIMA	(R, S)	6	0.84	0.55	0.50	0.71	0.11	0.1
SARIMA	(R, S)	7	0.87	0.11	0.11	0.7	0	0

Table 3.3. Case study I: detection delay.

Demand	Policy	Fault	SCVA			DPCA		
			T_d^2	Q	$S_{overall}$	T^2	SPE	$S_{overall}$
Normal	1-1	1	2.2	2.3	2.3	2.6	2.3	2.3
Normal	1-1	2	3.2	2.3	2.3	3	2.3	2.3
Normal	1-1	3	3.2	2.3	2.3	3	2.3	2.3
Normal	1-1	4	2.5	1.1	1.1	2.3	1.1	1.1
Normal	1-1	5	0.2	1.6	0.2	0.2	1.6	0.2
Normal	1-1	6	2.2	2.0	1.9	2.3	2	1.9
Normal	1-1	7	2.2	1.6	1.6	2.1	1.7	1.6
Normal	(R, S)	1	3.1	3.3	3.3	3.7	2.9	2.9
Normal	(R, S)	2	3.3	3.1	3.1	4	2.9	2.9
Normal	(R, S)	3	3.8	3	3	3.8	2.8	2.8
Normal	(R, S)	4	3.3	2.6	2.5	3.4	1.6	1.6
Normal	(R, S)	5	0.3	4.7	0.3	1.2	0.2	0.2
Normal	(R, S)	6	3.3	3.5	3.4	4	2.6	2.6
Normal	(R, S)	7	3.9	2.7	2.6	3.4	1.9	1.9
SARIMA	1-1	1	2.8	2.8	2.7	2.5	3	2.7
SARIMA	1-1	2	3	2.8	2.7	2.6	2.9	2.7
SARIMA	1-1	3	3	2.8	2.7	2.7	2.9	2.7
SARIMA	1-1	4	2.8	3.8	3.7	2.6	3.9	3.5
SARIMA	1-1	5	0	0.2	0	0.1	0.1	0
SARIMA	1-1	6	3	3.1	2.9	2.4	3.5	3.1
SARIMA	1-1	7	2.9	3.7	3.5	2.7	3.6	3.3
SARIMA	(R, S)	1	3	4	3.8	3	3.6	3.3
SARIMA	(R, S)	2	3.6	3.9	3.7	3.5	3.5	3.2
SARIMA	(R, S)	3	2.7	3.9	3.7	3.3	3.4	3.1
SARIMA	(R, S)	4	1.9	3	2.8	3.1	2.2	2.1
SARIMA	(R, S)	5	1.3	0.5	0.4	1.3	0.1	0.1
SARIMA	(R, S)	6	2.6	3.9	3.3	3	4.3	3.8
SARIMA	(R, S)	7	2.2	3.3	3.2	3.2	2.5	2.3

shipped from the Warehouse for replenishment. In this case study, the reorder levels for the customers and the two products are all set to 300, and the initial inventories of the filled SKUs are all set to 500. For the warehouse, the reorder levels are set to 3000, and the initial inventories of the filled SKUs are set to 5000. The initial inventories of empty SKUs are all set to 0.

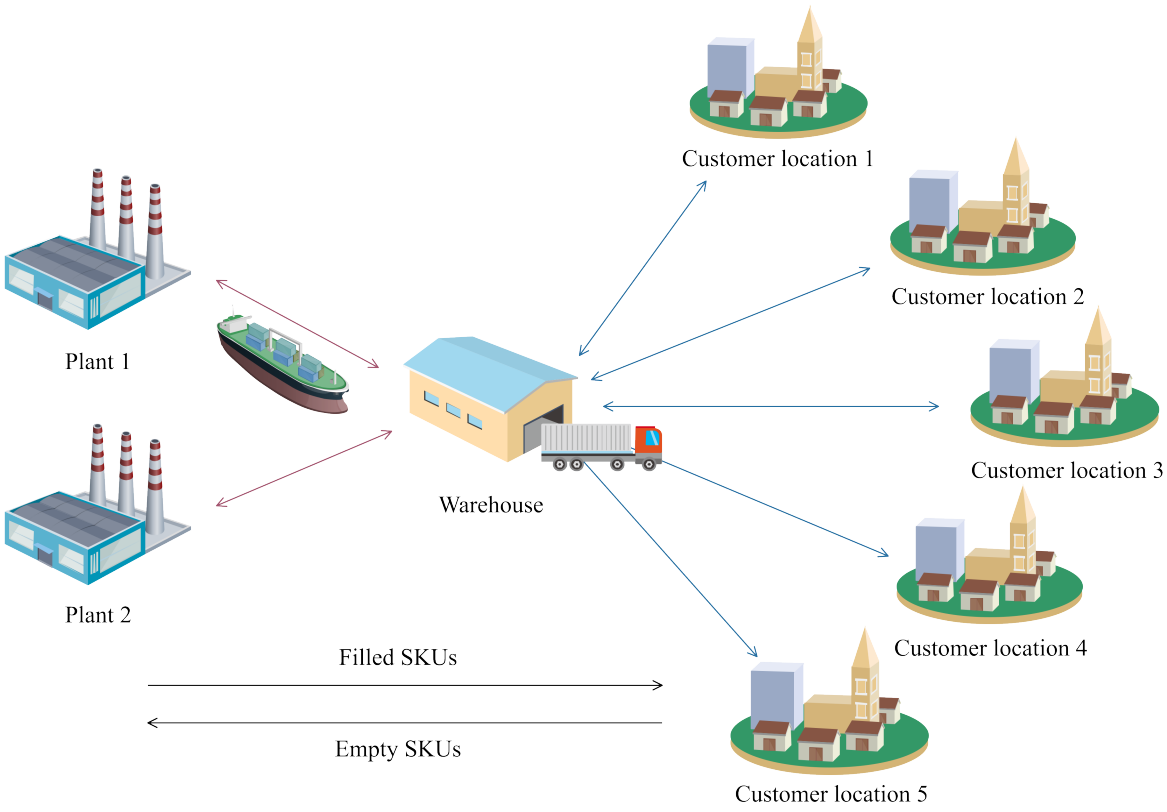


Figure 3.6. Case study II: a packaged liquefied gas supply chain.

The simulation model of the packaged liquefied gas supply chain is developed using `Python` based on ABM. It is similar to the model developed in Wang et al. (2020), where the details and results of simulation are described. Since there are multiple customers in this supply chain, multivariate demand models are investigated to take into account the cross-correlation and autocorrelation. Two demand models are considered in this case study. The first one is multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, where μ is the vector of mean and Σ is the covariance matrix. It is used to model cross-correlated demand. With this model, the customer demands have cross-correlation, but does not show autocorrelation. In this case

study, the mean of demand is set to 50 for the 5 customer locations.

In order to model multivariate autocorrelated demand, a Vector AutoRegressive Moving Average (VARMA) time series model is adopted. A K-dimensional VARMA(p, q) process y_t can be expressed as (Lütkepohl, 2005; Box et al., 2015):

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} M_j \varepsilon_{t-j} \tag{3.43}$$

where $y_t \in \mathbb{R}^K$, and ε_t is zero mean white noise with nonsingular covariance matrix.

The VARMA model for demand scenario generation is built using the `VARMAX()` API provided by the `statsmodels` library. The model is trained using data extracted from the dataset.

The input and output variables of this supply chain system are defined based on the state-space model. The outputs include inventory of filled and empty SKUs of the Warehouse and Customers, which involves 24 variables. The input variables include the shipment quantity of filled and empty SKUs between Warehouse and Customers (20 variables), and shipment quantity of filled and empty SKUs between Warehouse and Plants (4 variables), and the Customer demand of 2 products (10 variables). Therefore, there are a total of 58 variables collected for analysis.

The number of training samples is 2000. For the multivariate Gaussian demand, the autocorrelation plot of supply chain data is shown in Figure 3.7. It is observed that the autocorrelation shows a periodic pattern, with a period of 8. This is potentially related to the forward and reverse structures of the supply chain. The lag order for augmenting the observation is thus determined as $l = h = 8$. The observation vector that comprises of 58 variables are augmented into $58 \times 8 = 464$ variables. The R^2 and Q^2 using different number of PCs are shown in Figure 3.8. It can be seen that to achieve a Q^2 of approximately 70%, DPCA needs 51 PCs. The MSE for determining the state order and sparsity parameter value for SCVA is shown in Figure 3.9. The MSE value under a certain sparsity parameter value is plotted over the state order using a line plot. It is observed that with a certain sparsity parameter value, the MSE decreases when the state order increases. With a certain state order, the MSE decreases when the sparsity parameter decreases. This indicates that the ability of

SCVA in explaining the data gets stronger with more canonical states and a larger sparsity parameter. When the sparsity parameter value is within the range of $[0.7, 1]$, it does not have a significant impact on the MSE. Therefore, a state order $d = 20$ and a sparsity parameter $c = 0.8$ is determined as a good hyperparameter combination for SCVA, as this combination achieves a low MSE using a relatively small number of canonical states. Therefore, the 464 supply chain variables are reduced to 20 canonical states. This implies the strong ability of SCVA in dimensionality reduction of supply chain data.

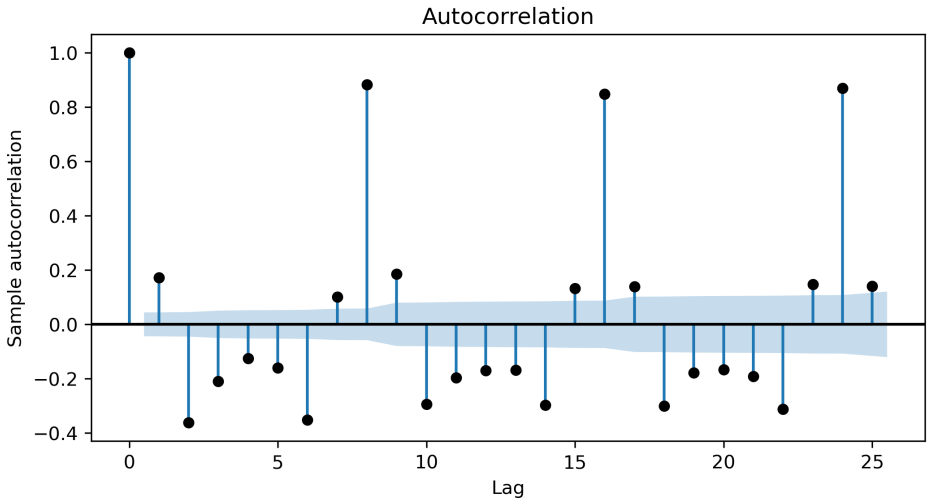


Figure 3.7. Case study II: Gaussian demand, autocorrelation plot.

For the model constructed for Gaussian demand, it is observed that both DPCA and SCVA have relatively high FARs (over 30%) for the testing data. Therefore, the confidence limits have been adjusted before comparison using the method of Russell et al. (2000b). Given a significance level α, the confidence limit of a monitoring statistic is set in such way that a $(1 - \alpha)$ portion of the samples are below the limit. α is set to 1% in this work, thus the adjusted limits correspond to a significance level of 1%. For the statistics that showed high FARs, the adjustment increased their limits and reduced the FARs. It is believed that this adjustment provides a fairer basis for the comparative study of sensitivities.

The SCMo models under VARMA demand are constructed in a similar way. The R^2 and Q^2 using different number of PCs are shown in Figure 3.10. The MSE for determining the state order and sparsity parameter for SCVA is shown in Figure 3.11. The results of dimensionality reduction and FAR using the SCVA and DPCA models are presented in Table 3.4. It is

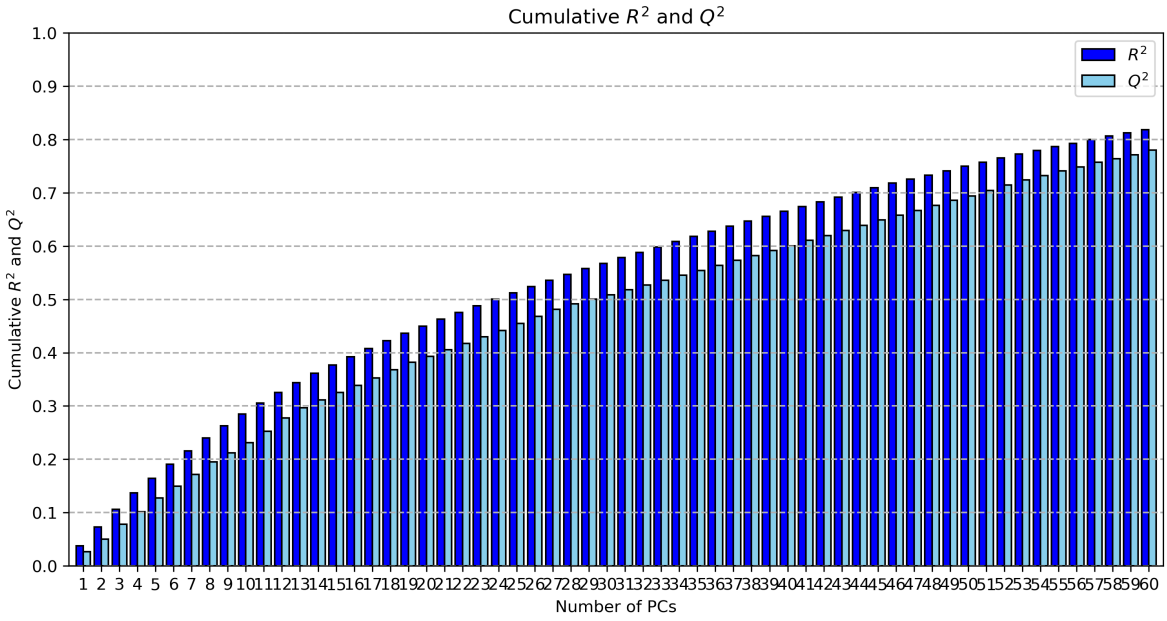


Figure 3.8. Case study II: Gaussian demand, DPCA R^2, Q^2.

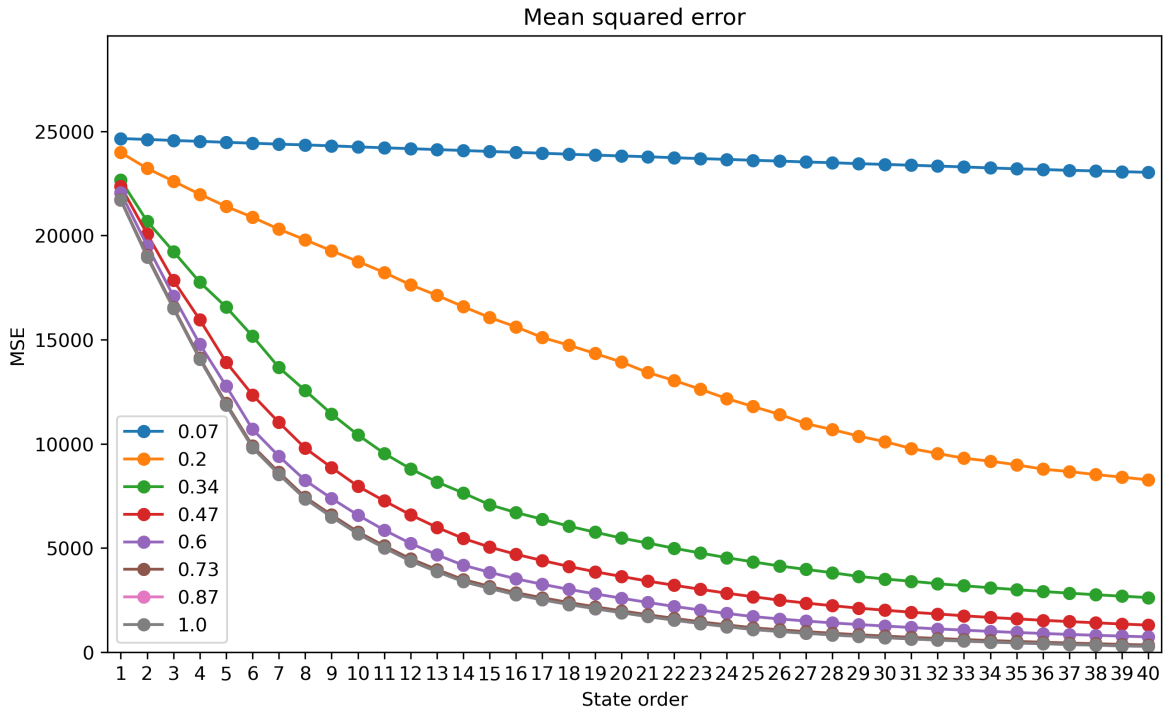


Figure 3.9. Case study II: Gaussian demand, SCVA MSE.

observed that when the demand is autocorrelated (VARMA), both SCVA needs more CSs to model the data, and DPCA needs more PCs. However, in both cases of cross-correlated and autocorrelated demand, SCVA identifies a small system order than DPCA. Therefore, in terms of dimensionality reduction for supply chain system, SCVA is more efficient than DPCA. Similar to the results of case study I, it is observed that when the demand follows a VARMA model, $S_{overall}$ of SCVA shows a lower FAR than $S_{overall}$ of DPCA. In general, both models show satisfactory FARs after the adjustment of confidence limits.

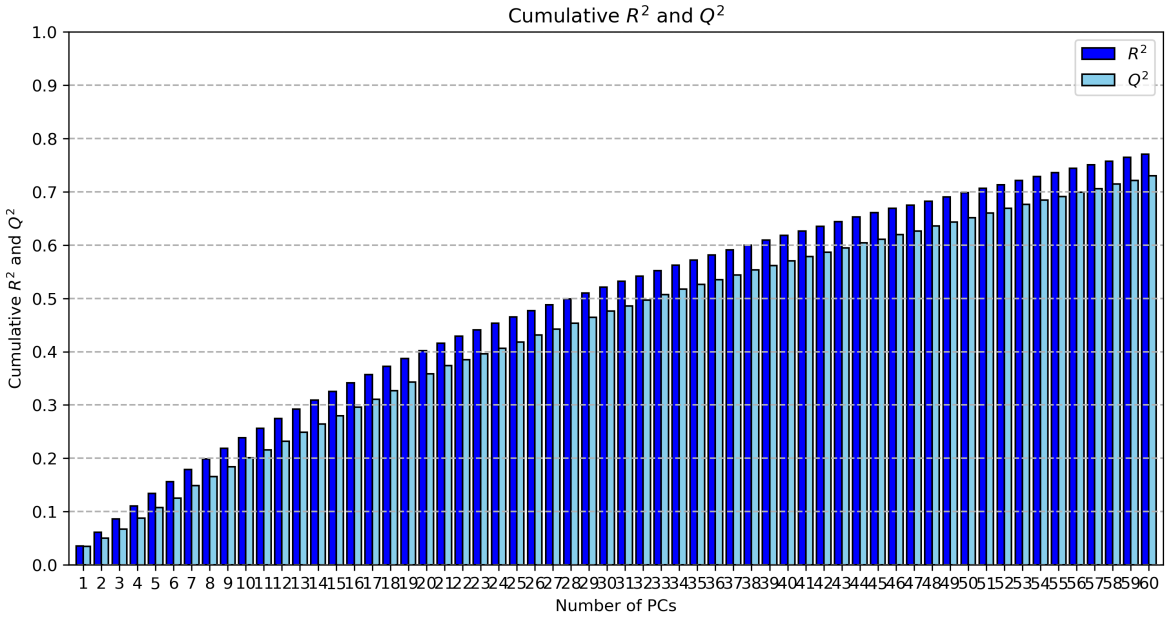


Figure 3.10. Case study II: VARMA demand, DPCA R^2, Q^2.

Table 3.4. Case study II: average FAR.

Demand	Lag l, h	SCVA				DPCA			
		N_{CS}	T_d^2	Q	$S_{overall}$	N_{PC}	T^2	SPE	$S_{overall}$
Gaussian	8	20	0.011	0.011	0.022	51	0.012	0.01	0.021
VARMA	8	30	0.001	0.041	0.041	57	0.006	0.065	0.069

In order to examine the ability of SCVA in SCMo in this case, 7 faults are designed and simulated, which are:

(1) Delay in the transportation between the Warehouse and a Customer. The faulty delay is set to 4 time periods.

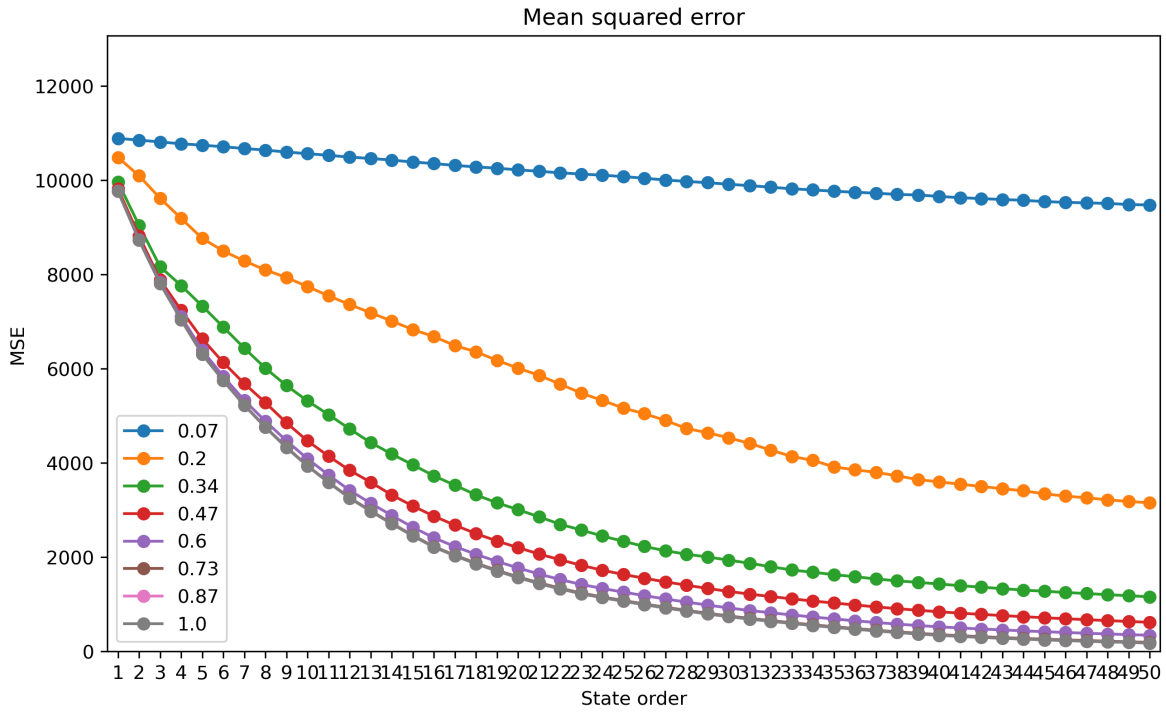


Figure 3.11. Case study II: VARMA demand, SCVA MSE.

(2) Delay in the transportation between the Plant and Warehouse. The faulty delay is set to 6 time periods.

(3) Delay in the refilling process of the Plant. The faulty delay is set to 6 time periods.

(4) An increase in the customer demand. The demand is multiplied by 2.

(5) An increase in the customer demand, and a delay in the transportation between the Warehouse and a Customer. The demand is multiplied by 1.5, and the faulty delay is set to 2 time periods.

(6) Delay in the transportation between the Plant and Warehouse, and between the Warehouse and a Customer. The faulty delays are set to 4 and 3 time periods, respectively.

(7) Delay in the refilling process of Plant, and delay in the transportation between the Plant and Warehouse. The faulty delays are set to 4 and 5 time periods, respectively.

The above faults may have an impact on the inventory of SKUs, and lead to a shortage of

filled SKUs and accumulation of empty SKUs at one or more locations. SCVA and DPCA are tested under 100 random demand scenarios. The MDRs of SCVA and DPCA under different situations are presented in Table 3.5. It is observed that for the 7 faults and 2 demand models considered in this case, T_d^2 by SCVA and T^2 by DPCA both show high MDRs. For most of the faults, their MDRs are over 80%, which is a high level. This indicates that the two statistics are very likely to miss the faults when they occur. By contrast, the Q statistic by SCVA and SPE by DPCA have low MDRs. For most of the faults, their MDRs are below 8%, which may be considered to be a satisfactory level. This implies that for both SCVA and DPCA, the monitoring statistics that quantify variations in the residual space are more sensitive and reliable in SCMo. The results of $S_{overall}$ are close to the residual statistics for both SCVA and DPCA.

The detection delays of SCVA and DPCA under different scenarios are presented in Table 3.6. Since T_d^2 by SCVA and T^2 by DPCA have high MDRs, the detection delays of them do not provide much useful information. The SPE by DPCA has a smaller detection delay than Q by SCVA overall. For the considered faults, DPCA is able to raise an alarm about 2-3 periods earlier than SCVA. Considering that SCVA uses significantly fewer states than DPCA, the performance of SCVA is acceptable.

In summary, SCVA shows advantage over DPCA in terms of the dimensionality reduction of supply chain data in this case study. Compared with cross-correlated demand, autocorrelated demand requires more states or components for modelling. In both cases, SCVA determines a smaller system order than DPCA, and achieves comparable performance to DPCA in terms of FAR, MDR and detection delay with significantly fewer states. Both SCMo models rely on the statistic in the residual space.

3.5 Conclusion

This chapter proposes a data-driven SCMo model based on CVA. This model considers a supply chain as a dynamic system and utilizes the state-space model of supply chain. The ability of the model in supply chain fault detection is validated on two simulation case studies.

Table 3.5. Case study II: MDR.

Demand	Fault	SCVA			DPCA		
		T_d^2	Q	$S_{overall}$	T^2	SPE	$S_{overall}$
Gaussian	1	0.71	0	0	0.87	0	0
Gaussian	2	0.92	0	0	0.96	0	0
Gaussian	3	0.91	0.04	0.03	0.96	0	0
Gaussian	4	0.84	0	0	0.44	0	0
Gaussian	5	0.63	0.02	0.02	0.71	0	0
Gaussian	6	0.78	0.13	0.12	0.85	0.02	0.01
Gaussian	7	0.92	0.04	0.03	0.96	0	0
VARMA	1	1	0	0	0.88	0	0
VARMA	2	0.95	0.01	0.01	0.88	0	0
VARMA	3	0.94	0.1	0.1	0.92	0	0
VARMA	4	0.9	0.06	0.06	0.19	0.07	0.06
VARMA	5	0.97	0.06	0.06	0.72	0.01	0.01
VARMA	6	0.99	0	0	0.94	0	0
VARMA	7	0.99	0.05	0.05	0.92	0	0

Table 3.6. Case study II: detection delay.

Demand	Fault	SCVA			DPCA		
		T_d^2	Q	$S_{overall}$	T^2	SPE	$S_{overall}$
Gaussian	1	13.4	8.3	8.2	13.2	4.6	4.6
Gaussian	2	6.1	16.4	15.6	4.5	13.8	13.5
Gaussian	3	7.6	16.4	15.6	4.5	13.8	13.5
Gaussian	4	11.4	1.4	1.3	6.4	0.1	0.1
Gaussian	5	9.7	4.8	4.7	10.7	2	2
Gaussian	6	13.9	9.4	9.3	13.3	6	6
Gaussian	7	6.1	16.4	15.6	4.5	13.8	13.5
VARMA	1	–	7	7	13.8	4.1	4.1
VARMA	2	17.6	13.2	13.2	16.5	10.7	10.6
VARMA	3	18.7	13.2	13.2	14.6	10.9	10.8
VARMA	4	13.8	6.5	6.5	9.2	5	4.8
VARMA	5	15	7.4	7.4	11.1	5	4.9
VARMA	6	16	8.2	8.2	13.8	4.9	4.9
VARMA	7	19	13.4	13.4	15.1	10.8	10.6

Results show that CVA is able to detect different types of operational supply chain faults, and the CVA-based variable contribution plot is able to identify the fault-related variables and thus indicating the cause of the fault.

A comparative study of CVA and PCA in SCMo is carried out. Both cross-correlated and autocorrelated demand are considered. The FAR, MDR, detection delay and dimension reduction of the two methods are investigated. Results show that the autocorrelation in data requires more CVs for CVA and more PCs for PCA to model the NOC. It is observed that the residual statistics are more reliable in SCMo for both methods. CVA identifies a smaller system order than DPCA and achieves comparable performance to DPCA with significantly fewer states. Therefore, CVA is more efficient in modelling a supply chain system and detecting abnormal supply chain operations. This study indicates the advantage of applying state-space models to supply chains.

References

Aburto, L. and R. Weber (2007). "Improved supply chain management based on hybrid demand forecasts". In: *Applied Soft Computing* 7.1, pp. 136–144.

Axsäter, S. (2015). *Inventory Control*. 3rd. Springer, Cham.

Bae, H. and Y. Seo (2007). "BPM-based integration of supply chain process modeling, executing and monitoring". In: *International Journal of Production Research* 45.11, pp. 2545–2566.

Bansal, M., A. Adhitya, R. Srinivasan, and I. Karimi (2005). "An online decision support framework for managing abnormal supply chain events". In: ed. by L. Puigjaner and A. Espuña. Vol. 20. Computer Aided Chemical Engineering, pp. 985–990.

Barata, J. C. A. and M. S. Hussein (2012). "The Moore-Penrose pseudoinverse: A tutorial review of the theory". In: *Brazilian Journal of Physics* 42, pp. 146–165.

Blos, M. F., R. M. da Silva, and H.-M. Wee (2018). "A framework for designing supply chain disruptions management considering productive systems and carrier viewpoints". In: *International Journal of Production Research* 56.15, pp. 5045–5061.

Box, G. E. P., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015). *Time Series Analysis: Forecasting and Control*. 5th. John Wiley & Sons, Inc.

Burnham, K. P. and D. R. Anderson (2004). "Multimodel inference: Understanding AIC and BIC in model selection". In: *Sociological Methods & Research* 33.2, pp. 261–304.

Chae, B. ((2009). "Developing key performance indicators for supply chain: an industry perspective". In: *Supply Chain Management* 14.6, pp. 422–428.

Chaharsooghi, S. K., J. Heydari, and S. H. Zegordi (2008). "A reinforcement learning model for supply chain ordering management: An application to the beer game". In: *Decision Support Systems* 45.4, pp. 949–959.

Chatfield, D. C., J. G. Kim, T. P. Harrison, and J. C. Hayya (2004). "The bullwhip effect–impact of stochastic lead time, information quality, and information sharing: A simulation study". In: *Production and Operations Management* 13.4, pp. 340–353.

Chen, F. and R. Samroengraja (2000). "The stationary beer game". In: *Production and Operations Management* 9.1, pp. 19–30.

Chowdhury, P., S. K. Paul, S. Kaisar, and M. A. Moktadir (2021). "COVID-19 pandemic related supply chain studies: A systematic review". In: *Transportation Research Part E: Logistics and Transportation Review* 148, p. 102271.

Fahimnia, B., C. S. Tang, H. Davarzani, and J. Sarkis (2015). "Quantitative models for managing supply chain risks: A review". In: *European Journal of Operational Research* 247.1, pp. 1–15.

Fei, R. and D. Wang (2008). "Monitoring and early-warning of the supply chain by using system dynamics and neural networks". In: *2008 International Symposium on Computer Science and Computational Technology*. Vol. 1, pp. 325–329.

Fernández, E., E. Salomone, and O. Chiotti (2012). "A model driven development approach based on a reference model for predicting disruptive events in a supply process". In: *Computers in Industry* 63.5, pp. 482–499.

Fernández, E., C. M. Toledo, M. R. Galli, E. Salomone, and O. Chiotti (2015). "Agent-based monitoring service for management of disruptive events in supply chains". In: *Computers in Industry* 70, pp. 89–101.

Gilbert, K. (2005). "An ARIMA supply chain model". In: *Management Science* 51.2, pp. 305–310.

Goh, R. S. M., Z. Wang, X. Yin, X. Fu, L. Ponnambalam, S. Lu, and X. Li (2013). "RiskVis: Supply chain visualization with risk management and real-time monitoring". In: *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 207–212.

Gupta, A. and C. D. Maranas (2003). "Managing demand uncertainty in supply chain planning". In: *Computers & Chemical Engineering* 27.8, pp. 1219–1227.

Hardoon, D. R., S. Szedmak, and J. Shawe-Taylor (2004). "Canonical correlation analysis: An overview with application to learning methods". In: *Neural Computation* 16.12, pp. 2639–2664.

Heckmann, I., T. Comes, and S. Nickel (2015). "A critical review on supply chain risk – Definition, measure and modeling". In: *Omega* 52, pp. 119–132.

Irizarry, J., E. P. Karan, and F. Jalaei (2013). "Integrating BIM and GIS to improve the visual monitoring of construction supply chain management". In: *Automation in Construction* 31, pp. 241–254.

Jaipuria, S. and S. Mahapatra (2014). "An improved demand forecasting method to reduce bullwhip effect in supply chains". In: *Expert Systems with Applications* 41.5, pp. 2395–2408.

Jiang, B. and R. D. Braatz (2017). "Fault detection of process correlation structure using canonical variate analysis-based correlation features". In: *Journal of Process Control* 58, pp. 131–138.

Jiang, B., D. Huang, X. Zhu, F. Yang, and R. D. Braatz (2015a). "Canonical variate analysis-based contributions for fault identification". In: *Journal of Process Control* 26, pp. 17–25.

Jiang, B., X. Zhu, D. Huang, and R. D. Braatz (2015b). "Canonical variate analysis-based monitoring of process correlation structure using causal feature representation". In: *Journal of Process Control* 32, pp. 109–116.

Jiang, B., X. Zhu, D. Huang, J. A. Paulson, and R. D. Braatz (2015c). "A combined canonical variate analysis and Fisher discriminant analysis (CVA-FDA) approach for fault diagnosis". In: *Computers & Chemical Engineering* 77, pp. 1–9.

Jiang, Q., X. Yan, and B. Huang (2019). "Review and perspectives of data-driven distributed monitoring for industrial plant-wide processes". In: *Industrial & Engineering Chemistry Research* 58.29, pp. 12899–12912.

Jiang, Z., J. Lamothe, and F. Benaben (2017). "A monitoring framework of collaborative supply chain for agility". In: *IFAC-PapersOnLine* 50.1, pp. 13072–13077.

Juricek, B. C., D. E. Seborg, and W. E. Larimore (2004). "Fault detection using canonical variate analysis". In: *Industrial & Engineering Chemistry Research* 43.2, pp. 458–474.

Klema, V. and A. Laub (1980). "The singular value decomposition: Its computation and some applications". In: *IEEE Transactions on Automatic Control* 25.2, pp. 164–176.

Kourti, T. and J. F. MacGregor (1995). "Process analysis, monitoring and diagnosis, using multivariate projection methods". In: *Chemometrics and Intelligent Laboratory Systems* 28.1, pp. 3–21.

Kresta, J. V., J. F. Macgregor, and T. E. Marlin (1991). "Multivariate statistical monitoring of process operating performance". In: *The Canadian Journal of Chemical Engineering* 69.1, pp. 35–47.

Ku, W., R. H. Storer, and C. Georgakis (1995). "Disturbance detection and isolation by dynamic principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 30.1, pp. 179–196.

Larimore, W. E. (1990). "Canonical variate analysis in identification, filtering, and adaptive control". In: *29th IEEE Conference on Decision and Control*. Vol. 2, pp. 596–604.

Larimore, W. E. (1997). "Canonical variate analysis in control and signal processing". In: *Statistical Methods in Control and Signal Processing*. Ed. by T. Katayama and S. Sugimoto. Marcel Dekker, New York, pp. 83–120.

Lau, H., W. K. Pang, and C. W. Wong (2002). "Methodology for monitoring supply chain performance: a fuzzy logic approach". In: *Logistics Information Management* 15.4, pp. 271–280.

Li, X. and T. E. Marlin (2009). "Robust supply chain performance via Model Predictive Control". In: *Computers & Chemical Engineering* 33.12, pp. 2134–2143.

Lu, Q., B. Jiang, R. B. Gopaluni, P. D. Loewen, and R. D. Braatz (2018a). "Locality preserving discriminative canonical variate analysis for fault diagnosis". In: *Computers & Chemical Engineering* 117, pp. 309–319.

— (2018b). "Sparse canonical variate analysis approach for process monitoring". In: *Journal of Process Control* 71, pp. 90–102.

Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

MacGregor, J. F. and A. Cinar (2012). "Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods". In: *Computers and Chemical Engineering* 47, pp. 111–120.

Mastragostino, R., S. Patel, and C. L. Swartz (2014). "Robust decision making for hybrid process supply chain systems via model predictive control". In: *Computers & Chemical Engineering* 62, pp. 37–55.

McKinney, J. H., A. Radford, A. Stathacopoulos, G. Aifadopoulou, and G. Giannopoulos (2015). "The business value of supply chain visibility and monitoring". In: *Transportation Research Record* 2479.1, pp. 86–92.

Mele, F. D., E. Musulin, and L. Puigjaner (2005). "Supply chain monitoring: a statistical approach". In: *European Symposium on Computer Aided Process Engineering-15*. Ed. by L. Puigjaner and A. Espuna. Vol. 20B. Computer Aided Chemical Engineering. Elsevier, pp. 1375–1380.

Misra, S., M. Kapadi, R. D. Gudi, and D. Saxena (2019). "Resource optimization and inventory routing of the packaged liquefied gas supply chain". In: *Industrial & Engineering Chemistry Research* 58.18, pp. 7579–7592.

Negiz, A. and A. Çinar (1997). "Statistical monitoring of multivariable dynamic processes with state-space models". In: *AIChE Journal* 43.8, pp. 2002–2020.

Norvilas, A., A. Negiz, J. DeCicco, and A. Çinar (2000). "Intelligent process monitoring by interfacing knowledge-based systems and multivariate statistical monitoring". In: *Journal of Process Control* 10.4, pp. 341–350.

Odiowei, P. P. and Y. Cao (2010). "Nonlinear dynamic process monitoring using canonical variate analysis and kernel density estimations". In: *IEEE Transactions on Industrial Informatics* 6.1, pp. 36–45.

Oroojlooyjadid, A., M. Nazari, L. V. Snyder, and M. Takáč (2022). "A deep Q-network for the beer game: Deep reinforcement learning for inventory optimization". In: *Manufacturing & Service Operations Management* 24.1, pp. 285–304.

Qin, S. J. (2012). "Survey on data-driven industrial process monitoring and diagnosis". In: *Annual Reviews in Control* 36.2, pp. 220–234.

Queiroz, M. M., D. Ivanov, A. Dolgui, and S. F. Wamba (2020). "Impacts of epidemic outbreaks on supply chains: mapping a research agenda amid the COVID-19 pandemic through a structured literature review". In: *Annals of Operations Research*. DOI: https://doi.org/10.1007/s10479-020-03685-7.

Ruiz-Cárcel, C., Y. Cao, D. Mba, L. Lao, and R. Samuel (2015). "Statistical process monitoring of a multiphase flow facility". In: *Control Engineering Practice* 42, pp. 74–88.

Ruiz-Cárcel, C., L. Lao, Y. Cao, and D. Mba (2016). "Canonical variate analysis for performance degradation under faulty conditions". In: *Control Engineering Practice* 54, pp. 70–80.

Russell, E., L. H. Chiang, and R. D. Braatz (2000a). *Data-driven Methods for Fault Detection and Diagnosis in Chemical Processes*. Springer-Verlag, London.

Russell, E. L., L. H. Chiang, and R. D. Braatz (2000b). "Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis". In: *Chemometrics and Intelligent Laboratory Systems* 51.1, pp. 81–93.

Samuel, R. T. and Y. Cao (2015). "Kernel canonical variate analysis for nonlinear dynamic process monitoring". In: *IFAC-PapersOnLine* 48.8, pp. 605–610.

Schaper, C., W. Larimore, D. Seborg, and D. Mellichamp (1994). "Identification of chemical processes using canonical variate analysis". In: *Computers & Chemical Engineering* 18.1, pp. 55–69.

Severson, K., P. Chaiwatanodom, and R. D. Braatz (2016). "Perspectives on process monitoring of industrial systems". In: *Annual Reviews in Control* 42, pp. 190–200.

Simoglou, A., E. Martin, and A. Morris (2002). "Statistical performance monitoring of dynamic multivariate processes using state space modelling". In: *Computers & Chemical Engineering* 26.6, pp. 909–920.

Sterman, J. D. (1989). "Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment". In: *Management Science* 35.3, pp. 321–339.

Tran, T. H., M. Dobrovnik, and S. Kummer (2018). "Supply chain risk assessment: a content analysis-based literature review". In: *International Journal of Logistics Systems and Management* 31.4, pp. 562–591.

Treasure, R. J., U. Kruger, and J. E. Cooper (2004). "Dynamic multivariate statistical process control using subspace identification". In: *Journal of Process Control* 14.3, pp. 279–292.

Wang, J., C. L. E. Swartz, B. Corbett, and K. Huang (2020). "Supply chain monitoring using principal component analysis". In: *Industrial & Engineering Chemistry Research* 59.27, pp. 12487–12503.

Wang, Y., D. E. Seborg, and W. E. Larimore (1997). "Process monitoring based on canonical variate analysis". In: *1997 European Control Conference (ECC)*, pp. 3089–3094.

Witten, D. M., R. Tibshirani, and T. Hastie (2009). "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis". In: *Biostatistics* 10.3, pp. 515–534.

Yang, X., W. Liu, W. Liu, and D. Tao (2021). "A survey on canonical correlation analysis". In: *IEEE Transactions on Knowledge and Data Engineering* 33.6, pp. 2349–2368.

Yao, Y. and F. Gao (2008). "Subspace identification for two-dimensional dynamic batch process statistical monitoring". In: *Chemical Engineering Science* 63.13, pp. 3411–3418.

Zhou, K. and G. Rong (2010). "Study of supply chain monitoring system based on IDEF method". In: *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*. Vol. 1, pp. 278–281.

Zhu, X. and R. D. Braatz (2014). "Two-dimensional contribution map for fault identification". In: *IEEE Control Systems Magazine* 34.5, pp. 72–77.

Chapter 4

Risk-averse Decision-making for Supply Chain Operations via Safe Reinforcement Learning

For the study in this chapter, the following paper will be submitted:

[1] Jing Wang, Christopher L.E. Swartz, and Kai Huang. "Risk-averse decision-making for supply chain operations via safe reinforcement learning". *Omega – The International*

Journal of Management Science. Reject & resubmit. To be resubmitted.

4.1 Introduction

In Chapter 2 and Chapter 3, we focused on investigating the SCMo problem under uncertainties and risks. Data-driven SCMo helps improve the awareness of the operating conditions of supply chains, while in this thesis, it is considered to be only the first step for data-driven SCM. As a further step, we aim to investigate optimal supply chain control under uncertainty using data-driven methods, which is the research objective of this chapter. In the study presented in this chapter, a data-driven control method is applied to supply chain systems to provide decisions for operational control.

As has been mentioned in Chapter 1, there are generally three levels for supply chain planning, which are strategic/long-term planning, tactical/mid-term planning, and operational/short-term planning. In this chapter, we focus on operational supply chain planning under uncertainty. The existence of supply chain uncertainties, such as stochastic customer demands, and production and transportation delays, makes it complicated to manage a supply chain. Mathematical programming has been applied to optimize supply chain decision-making under uncertainty, such as stochastic programming, robust optimization, chance-constrained programming, fuzzy logic, and interval programming (Govindan et al., 2015; Ning and You, 2019). In recent years, data-driven optimization has developed rapidly and been applied to solve supply chain problems. Data-driven approaches integrate mathematical programming and machine learning, and use historical data to characterize uncertainty, instead of assuming the uncertainty model is perfectly given a priori as in conventional optimization techniques (Ning and You, 2019).

A promising technique that has been shown to be beneficial to SCM is *reinforcement learning* (RL) (Sutton and Barto, 2018). RL is a branch of machine learning, and is also known as a data-driven method, as it learns from system data rather than utilizing the mathematical model of system (Jiang et al., 2018). The RL framework is different from the mathematical programming-based paradigms mentioned above. In this framework, an agent learns optimal

decisions when interacting with the non-deterministic environment. Then the feedback information from the environment, i.e., state and reward, is used to optimize its actions, while a pre-determined model of environment is not necessarily required. RL provides the advantage of solving complex sequential decision-making problems through learning from previous experience. Therefore, it is considered as a powerful technique for solving problems where a large number of factors need to be taken into account, such as SCM problems (Kara and Dogan, 2018; Fuji et al., 2018).

Research on using RL for supply chain control emerged around two decades ago (Pontrandolfo et al., 2002; Giannoccaro and Pontrandolfo, 2002). Some advantages of RL policies over classical inventory policies have been demonstrated in the current literature. To the best of our knowledge, however, there still exists the following two research gaps.

First, RL-based control of *closed-loop* supply chains (CLSCs). A CLSC is a production and distribution system that focuses on collecting used products from customers and recovering added value by reusing the entire or partial product. It integrates both forward and reverse logistics, and enables remanufacturing of used products in parallel with manufacturing of new products (Savaskan et al., 2004; Guide and Van Wassenhove, 2009). A CLSC plays two main roles: (1) Operating value-added processes aimed at satisfying customers' demands, as in a conventional SC (also called open-loop SC); (2) Taking back the end-of-lifecycle products from customers and determining the best ways to recover their value. Due to their extensive economic and environmental potential, CLSCs have received attention from industry and academia over the past decades. A comprehensive review of relevant research is given by Govindan et al. (2015). All of the current works focus on forward SCs, while the control of CLSCs using RL has not been investigated. A main difference between a CLSC and a forward SC is the reverse flow in CLSC, which could involve a higher degree of uncertainty, for example, uncertain return rate, quality of returned products, and remanufacturing lead time (Ilgin and Gupta, 2010; Govindan et al., 2015; Dominguez et al., 2020). This makes the control of CLSCs more challenging than conventional forward SCs.

Second, risk-averse RL of supply chains. All of the current studies utilize risk-neutral RL algorithms for supply chain control, while the concept of *risk* or *safety* has not been

considered. It should be noted that risk management forms an important component of SCM and thus has been investigated in many works, e.g., You et al. (2009). Therefore, it is worth exploring SC control using RL with consideration of risk. In this chapter, the use of *safe RL* is explored for operational supply chain control. The terminology safe RL is first proposed by García and Fernández (2015) to describe a subfield within RL. It is defined as the process of learning policies that maximize the expected return in problems where ensuring reasonable system performance or obeying safety constraints is considered important. To put it simply, safe RL describes a class of RL algorithms that takes into account the concept of safety or risk. Generally, safe RL is implemented by transforming the optimization criterion or modifying the exploration process of conventional RL. Research on applying safe RL to SC control is scant.

This chapter aims to fill the above two research gaps. The main contributions of this chapter are:

1. Development of a RL-based method for operational control of CLSCs under uncertainty;
2. Development of a safe RL-based method for risk-averse operational control of supply chains under uncertainty;
3. A comparative study on the control performance of a classical Order-Up-To (OUT) inventory policy, a conventional RL algorithm (Q-learning), and two safe RL algorithms ($\hat{Q}$-learning and β-pessimistic Q-learning).

The rest of this chapter is organized as follows. In Section 4.2, a comprehensive review of related works on RL-based supply chain control is presented, and the research gaps are then identified. In Section 4.3, RL-based supply chain control is described, including a RL algorithm (Q-learning) and two safe RL algorithms ($\hat{Q}$-learning and β-pessimistic Q-learning). In order to examine the performance of RL algorithms for SC control, a SC simulation model is developed using the programming language `Python` for case studies. The details of the simulation model are presented in Section 4.4. Two case studies are presented in Section 4.5. The first one is a two-echelon SC consisting of a manufacturer and a retailer. The second one is a CLSC, which includes a remanufacturing process and has both a forward flow and a

reverse flow. The RL policies are compared with the classical OUT policies. Conclusions are given in Section 4.6.

4.2 Literature Review

A review of related works on SC control using RL is presented in this section. The articles are reviewed from four aspects, which are the supply chain problems they investigate, the SC configurations, the RL algorithms they use, and the advantages of RL they demonstrate. Table 4.1 gives a summary of related works in an ascending order of the year of publication from the top of the table to the bottom. The four aspects are listed in the second to fifth column of the table, and described in detail in Section 4.2.1 to Section 4.2.4, respectively.

4.2.1 Supply chain problems

In research studies related to this work, RL is commonly applied to optimize inventory or ordering decisions in a stochastic environment, such as decisions on the quantities of production, ordering, and shipment (Giannoccaro and Pontrandolfo, 2002; Chaharsooghi et al., 2008; Mortazavi et al., 2015; Kemmer et al., 2018), order-up-to level and re-order point (Zhang and Bhattacharyya, 2007; Jiang and Sheng, 2009), safety factor level and safety lead time (Kim et al., 2008), and selection of supplier and transportation mode (Pontrandolfo et al., 2002). Different studies may have different focuses. Qiu et al. (2007) use RL to investigate the impact of business service mode, which is defined as a combination of business process and information scenario, on distribution systems. Kwon et al. (2008), Jiang and Sheng (2009), and Kim et al. (2010) investigate the inventory control problem with a constraint on the customer service level, in which a predefined target service level needs to be satisfied by the retailers. Kwak et al. (2009) use RL to address the Vendor Managed Inventory (VMI) problem, in which the supplier makes replenishment decisions for the retailer based on the sales and inventory information provided by the retailer. Sui et al. (2010) focus on the consignment-inventory VMI system, in which the supplier owns the inventory at the retailer and pays for

Table 4.1. A summary of related works.

Article	SC problem	SC structure	RL algorithm	Advantage
Pontrandolfo et al. (2002)	Global SCM	Network	SMART	Outperforms 2 heuristics
Giannoccaro and Pontrandolfo (2002)	Inventory management	Serial	SMART	Outperforms (R, S) policy
Zhang and Bhattacharyya (2007)	Ordering management	Network	Q-learning	
van Tongeren et al. (2007)	Competitive SCM	Serial (Beer Game)	Q-learning	Outperforms 1-1 policy; reduces bullwhip effect
Qiu et al. (2007)	Impact of business service mode	Divergent	RL	
Chaharsooghi et al. (2008)	Ordering management	Serial (Beer Game)	Q-learning	Outperforms 1-1 policy and GA
Kwon et al. (2008)	Inventory control with target service level	Divergent	Case-based myopic RL	
Kim et al. (2008)	Inventory control	Serial	Action-reward learning	Outperforms rolling horizon (r, Q)
Valluri et al. (2009)	Learning ordering rules	Serial (Beer Game)	Q-learning, Sarsa(λ), tile coding with Sarsa(λ)	
Jiang and Sheng (2009)	Inventory control with target service level	Network	Case-based RL	
Kwak et al. (2009)	VMI problem	Serial	Action-reward learning	Outperforms 2 newsboy models
Kim et al. (2010)	Inventory control with target service level	Serial	Action-reward learning	Outperforms 3 comparison models
Sui et al. (2010)	Consignment-inventory VMI system	Divergent	Q-learning, NN approximator	Outperforms the newsvendor rule
Mortazavi et al. (2015)	Ordering management	Serial	Q-learning	High service level
Kemmer et al. (2018)	Optimization of production and distribution	Divergent	Approximation Sarsa, REINFORCE	Outperforms static (r, Q) policy
Fuji et al. (2018)	Competitive SCM	Serial (Beer Game)	Deep multi-agent RL	Outperforms expert players
Kara and Dogan (2018)	Perishable inventory management	Dyadic	Q-learning, Sarsa	Outperforms GAs
Peng et al. (2019)	Capacitated SC optimization	Serial, divergent	Deep RL	Outperforms (r, Q) policy
Aghaie and Heidary (2019)	MNVPSD	Network	Q-learning	More efficient than GA
Hubbs et al. (2020)	Chemical production scheduling		Deep RL, A2C	Outperforms naive MILP; is competitive with shrinking horizon MILP
Gijsbrechts et al. (2022)	Lost sales, dual sourcing, and multi-echelon inventory problems		Deep RL, A3C	Can match advanced heuristics and other ADP methods
Oroojlooyjadid et al. (2022)	Inventory optimization	Serial (Beer Game)	Deep RL, SRDQN	Outperforms a base-stock policy; robust to changes in costs

the holding cost. The supplier is not paid until the item is sold, and is penalized for stock-out. Valluri et al. (2009) conduct comparative study on the performance of three different RL algorithms in learning supply chain operating rules. van Tongeren et al. (2007) and Fuji et al. (2018) investigate the management of competitive supply chains, where the agents make decisions individually and independently in a distributed environment based on partial or local information, and also cooperate to guarantee the performance of the entire supply chain. Kara and Dogan (2018) focus on perishable inventory management where the age information of products is taken into account, and use RL approaches to determine ordering policies that can strike a balance between the shortage cost and outdating cost resulting from discarded products. Aghaie and Heidary (2019) consider the multi-period newsvendor problem with supplier disruption (MNVPSD), and use RL to optimize behavior of the risk-sensitive retailer with respect to forward and option contracts. Peng et al. (2019) focus on the capacitated supply chain optimization problem, where there are capacity constraints on the plant production and retailer storage. Hubbs et al. (2020) apply deep RL to a chemical production scheduling process and achieve efficient online scheduling. Gijsbrechts et al. (2022) evaluate the performance of deep RL in three classic inventory problems, which are lost sales, dual-sourcing, and multi-echelon inventory management. Oroojlooyjadid et al. (2022) apply deep RL to play the beer game, which is a decentralized multi-agent problem where the agents cooperate to minimize the total cost with only local information available.

4.2.2 Supply chain configurations

Generally, the configuration of traditional supply chains can be categorized into six types (Beamon and Chen, 2001; Giard and Sali, 2013): (1) Dyadic, a simple structure that consist of a single supplier and a single customer; (2) Serial, a linear structure where each node is connected to at most one predecessor and one successor; (3) Divergent, a tree structure where each node has at most one predecessor, but may have multiple successors; (4) Convergent, a tree structure where each node has at most one successor, but may have multiple predecessors; (5) Conjoined, a combination of convergent and divergent structures; (6) Network, a general structure that does not fall within the previous classes.

Kara and Dogan (2018) investigate a dyadic system, where a single retailer manages inventory to meet customer demands. In terms of the reviewed works, the most commonly used configuration is the serial supply chain, which has only one agent for each echelon. Serial supply chains consisting of two to four echelons are used by, e.g., Giannoccaro and Pontrandolfo (2002), Kim et al. (2008), Kwak et al. (2009), Kim et al. (2010), Mortazavi et al. (2015), Kemmer et al. (2018), and Peng et al. (2019). In particular, a benchmark four-echelon serial supply chain, the Beer Distribution Game, is used as a case study by van Tongeren et al. (2007), Chaharsooghi et al. (2008), Valluri et al. (2009), and Fuji et al. (2018).

Two-echelon divergent supply chains consisting of single supplier or distributor and multiple retailers are investigated by Qiu et al. (2007), Kwon et al. (2008), and Sui et al. (2010). A three-echelon divergent supply chain is considered by Peng et al. (2019). Two-echelon or multi-echelon supply chain networks with multiple agents in each echelon are investigated by Pontrandolfo et al. (2002), Zhang and Bhattacharyya (2007), Jiang and Sheng (2009), and Aghaie and Heidary (2019).

4.2.3 RL algorithms

In terms of the RL algorithms that have been applied to supply chain control, Q-learning is the most popular one in the reviewed works (Zhang and Bhattacharyya, 2007; Chaharsooghi et al., 2008; Mortazavi et al., 2015; Kemmer et al., 2018; Kara and Dogan, 2018; Aghaie and Heidary, 2019). Sarsa, another tabular solution method, is used by Kara and Dogan (2018). Pontrandolfo et al. (2002) and Giannoccaro and Pontrandolfo (2002) employ the semi-Markov average reward technique (SMART), which is based on value iteration and solves semi-Markov decision processes over the infinite time horizon under an average reward criterion. Kwon et al. (2008) and Jiang and Sheng (2009) investigate the case-based RL approach, which uses the case-based reasoning to discretize the state space dynamically. Kim et al. (2008), Kwak et al. (2009), and Kim et al. (2010) adopt action-reward learning, which is a RL model simpler than the Markovian learning models, but has the ability to learn when the environment is dynamically changing.

RL with function approximation has been applied to supply chain control to learn a functional mapping of the action-value function. Valluri et al. (2009) employ linear functions to approximate the state-action pair values of Q-learning, Sarsa(λ), and tile coding with Sarsa(λ). Kemmer et al. (2018) use linear function to approximate the Sarsa algorithm. Sui et al. (2010) use a back-propagated neural network as the approximator of the Q-factors of Q-learning. Using deep neural networks (NNs) for value and policy approximations is called the *deep RL* in the literature. Deep RL algorithms that have been applied to supply chain control include, e.g., the Vanilla Policy Gradient algorithm (Peng et al., 2019), the Advantage Actor-Critic (A2C) algorithm (Hubbs et al., 2020), the Asynchronous Advantage Actor Critic (A3C) algorithm (Gijsbrechts et al., 2022), and the shaped-reward deep Q-network (SRDQN) algorithm (Oroojlooyjadid et al., 2022). Through combining the actor-critic deep RL with learning management, Fuji et al. (2018) propose a deep multi-agent RL technique for decision making in multi-agent system.

4.2.4 Advantages of RL in supply chain control

It has been demonstrated by comparative studies in relevant works that RL policies have advantages over some classical inventory policies. The superiority of RL in supply chain control, as inferred through case studies, include:

(1) Outperforms expert players in the Beer Game (Fuji et al., 2018);

(2) Outperforms some classical inventory policies in terms of cost or profit, for example, the one-for-one (1-1) policy (van Tongeren et al., 2007; Chaharsooghi et al., 2008), static or rolling horizon (r, Q) policy (Kim et al., 2008; Kemmer et al., 2018; Peng et al., 2019), (R, S) policy (Giannoccaro and Pontrandolfo, 2002), newsvendor or newsboy model for VMI system (Sui et al., 2010);

(3) Performs better or almost better than genetic algorithms (GAs) (Chaharsooghi et al., 2008; Kara and Dogan, 2018; Aghaie and Heidary, 2019);

(4) Outperforms naive mixed-integer linear programming (MILP) approaches, and is com-

petitive with the MILP with a shrinking horizon for chemical production scheduling (Hubbs et al., 2020);

(5) Can match advanced heuristics and other approximate dynamic programming (ADP) methods (Gijsbrechts et al., 2022);

(6) Can reduce the bullwhip effect (van Tongeren et al., 2007);

(7) Is robust against changes in the holding and shortage costs (Oroojlooyjadid et al., 2022).

Based on the review of current literature, it is observed that although RL has been applied to solve many SCM problems, two research gaps still exist. The first one is the control of CLSC using RL. Due to the existence of reverse flow, CLSC problems could involve more uncertain parameters than conventional forward SC problems, and thus are more complicated. The promising RL technique is worth investigating for control of CLSC under uncertainty. The second research gap is the consideration on risk in supply chain control. Since the risk-neutral criterion may not satisfy the need of risk-sensitive or risk-averse decision makers, we believe it is worth exploring safe RL algorithms for supply chain control. Therefore, this work has two objectives: (1) to examine the performance of RL for control of CLSCs; (2) to explore the use of safe RL for risk-averse SC control.

4.3 Reinforcement Learning-Based Supply Chain Control

This section describes the methodology for supply chain control using RL. Section 4.3.1 introduces the RL framework and the *Q-learning* algorithm. Section 4.3.2 introduces the concept of safe RL and two algorithms, $\hat{Q}$*-learning* and β*-pessimistic Q-learning*. The procedure of applying RL to supply chain control is described in Section 4.3.3.

4.3.1 Reinforcement learning

Generally speaking, machine learning can be categorized into three paradigms: supervised learning, unsupervised learning, and RL (Lee et al., 2018; Shin et al., 2019). Supervised learning is aimed at constructing a mapping from input to output based on labelled data, while unsupervised learning discovers patterns from unlabeled data. RL is a different machine learning task. The goal of RL is to learn a mapping from situations to actions such that a numerical reward signal is maximized for the decision maker.

RL has two major components: (1) the *agent*, which is the learner and decision maker; (2) the *environment*, which comprises everything outside the agent and interacts with the agent. RL explicitly considers the problem of an active decision-making agent seeking to achieve a goal despite uncertainty in the environment. A diagram for the interaction between the agent and the environment is shown in Figure 4.1 (Sutton and Barto, 2018). Let $t = 0, 1, 2 \ldots$ be the index of discrete time steps. At each time step t, the agent observes the *state* of environment $s_t \in \mathcal{S}$, and selects an action $a_t \in \mathcal{A}(s_t)$ according to a *policy* π. The policy π is a mapping from states to probabilities of selecting each possible action. For example, $\pi(a_t|s_t)$ is the probability of choosing action a_t when the state is s_t. One step later, partially due to its action, the agent receives a numerical *reward* $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and a new state s_{t+1} of the environment. The environment is modelled as a *Markov Decision Process* (MDP) that has the following *Markov property*:

$$Pr(s_{t+1}, r_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = Pr(s_{t+1}, r_{t+1}|s_t, a_t), \tag{4.1}$$

Equation (4.1) indicates that the probability of each possible value of state s_{t+1} and reward r_{t+1} depends only on the immediately preceding state s_t and action a_t, while not on any earlier trajectory given s_t and a_t. If the sets of states, actions, and rewards have a finite number of elements, then the MDP is said to be a finite MDP.

The goal of the agent is to maximize the expectation of the cumulative sum of the reward it receives, or the *expected return* that is defined as some function of the reward sequence. If

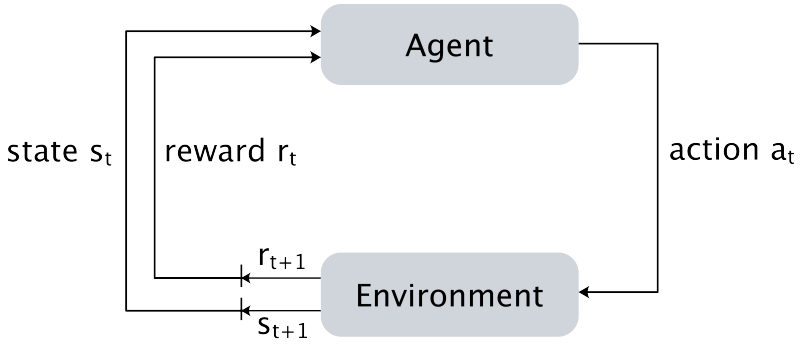


Figure 4.1. The agent-environment interaction in RL.

the present value of future reward is taken into account, the expected *discounted return* is typically defined as $G_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma \in [0, 1]$ is the *discount rate*. The agent is said to be 'myopic' if $\gamma = 0$, while it becomes more farsighted as γ approaches 1.

The expected return for the agent when it starts from a state s and follows policy π thereafter is called the *value function* of s under π, denoted as $v_\pi(s)$:

$$v_\pi(s) := \mathbb{E}_\pi \left[G_t | s_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \tag{4.2}$$

Similarly, the expected return for the agent when it starts from state s, takes action a, and follows policy π thereafter is called the value of taking a in s under π , denoted as $q_\pi(s, a)$:

$$q_\pi(s, a) := \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \tag{4.3}$$

v_π and q_π are called the *state-value function* and the *action-value function* for policy π, respectively. Solving a RL problem means finding an optimal policy $\pi^\star$ that outperforms all other policies across all possible scenarios, which yields an optimal action-value function $q^\star$ (Spielberg et al., 2020):

$$q^\star(s, a) = \max_\pi q_\pi(s, a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{4.4}$$

For an optimal policy, we have:

$$q^\star(s,a) = \mathbb{E}\left[r_{t+1} + \gamma v^\star(s_{t+1}) | s_t = s, a_t = a\right] \tag{4.5}$$

By selecting actions with the highest values, $\pi^\star$ can be recovered from $q^\star$:

$$\pi^\star(a|s) = \begin{cases} 1, & \text{if} \quad a = \arg\max_{a' \in \mathcal{A}(s)} q_\pi(s,a'), \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

Q-learning, a temporal-difference (TD) control algorithm proposed by Watkins and Dayan (1992), has been widely used to solve finite Markov decision problems. This algorithm approximates $q^\star$ through a learned action-value function Q using the following update:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{4.7}$$

where $\alpha \in (0,1]$ is the step-size parameter or learning rate.

Algorithm 4.1: Q-learning

Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$

Initialize $Q(s,a)$ arbitrarily (e.g., 0), $\forall s, a$, and $Q(terminal, \cdot) = 0$
for *each episode* **do**
  Initialize state s
  for *each step of episode* **do**
    Choose a using policy derived from Q, e.g., ε-greedy
    Take action a, observe reward r and new state s'
    Update: $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$
    $s \leftarrow s'$
    Until s is terminal
  end
end

The Q-learning procedure is given by Algorithm 4.1 (Sutton and Barto, 2018). It is typically implemented as a tabular method based on a Q-table, which is a 2-D lookup table for storing the Q-values of all the state-action pairs. For problems with finite state and action sets,

Q-learning applies directly. For supply chain control problems where the state and action spaces are usually large or continuous, the first step is to discretize the spaces appropriately, thus they have a reasonable number of elements (Chaharsooghi et al., 2008; Mortazavi et al., 2015; Kara and Dogan, 2018). The Q-table can be created by using the discrete states and actions as columns and rows, with all entries initialized with arbitrary values, such as 0. The training process takes place through a series of episodes. For each step of an episode, the agent takes the action with the highest Q-value according to the Q-table and the current state. Then it observes the reward and new state, and uses this information to update corresponding entry of the Q-table.

Q-learning does not require a model of the environment, and is well suited for step-by-step incremental computation. It is classified as *off-policy*, because the *target policy* it approximates and the *behavior policy* it follows are different. The target policy is greedy, i.e., taking the action with the highest reward, while the behavior policy can be, e.g., ε-greedy, which means the agent has a probability of $1-\varepsilon$ for exploiting the current optimal action, and also allows for a probability of ε for exploring the action space randomly. Thus, the compromise between *exploitation* and *exploration* is determined by the parameter ε.

4.3.2 Safe RL

Supply chain risks include operational risks related to inherent uncertainties and disruption risks caused by disasters (Tang, 2006). The risk attitude of a decision maker, i.e., risk-neutral, risk-averse, or risk-seeking, has a major impact on the assessment of supply chain performance and the decision-making (Heckmann et al., 2015). The optimization criterion Q-learning uses, i.e., Equations (4.3) and (4.4), is an expectation-based criterion. It is said to be risk-neutral and is based on long-term consideration that the decision process is repeated for sufficiently many times under the same conditions. However, it should be noted that maximizing the long-term expected return does not guarantee the avoidance of occurrences of large negative outcomes. Thus, the expected value criterion is not always considered to be reliable in operations management (Heger, 1994; García and Fernández, 2015; Govindan and Fattahi, 2017). For risk-averse decision makers, a criterion that takes into account the risk

could be a better option. When the probability distribution of uncertain parameters is known, risk measures can be defined for assessment or incorporated into the objective functions of supply chain problems, such as the variance, downside risk, conditional value-at-risk, and the worst-case cost (Gupta and Maranas, 2003; You et al., 2009; Gebreslassie et al., 2012; Heckmann et al., 2015). In case the probability distribution of uncertainty is unknown, robust optimization approaches that use a minimax objective or the worst-case criterion can be used to obtain robust solutions (Pishvaee et al., 2011; Govindan and Fattahi, 2017; Ning and You, 2019).

In this work, the use of safe RL for risk-averse or risk-avoidance SC control is explored. Safe RL algorithms are RL algorithms that take into account the concept of safety or risk. Generally, safe RL algorithms can be categorized into two frameworks (García and Fernández, 2015): (1) transforming the optimization criterion, e.g., using worst-case criterion, risk-sensitive criterion, or constrained criterion; (2) modifying the exploration process of conventional RL by incorporating external knowledge or using a risk measure. This work focuses on RL based on the worst-case criterion or the maximin criterion, which is given by:

$$\max_{\pi} \min_{\omega} \mathbb{E}_{\pi,\omega} \left(\sum_{t=0}^{\infty} \gamma^t r_t \right) \tag{4.8}$$

where ω represents the trajectory under policy π. Under optimality criterion Equation (4.8), a policy is considered to be optimal if its worst-case return is superior (García and Fernández, 2015).

$\hat{Q}$-learning, proposed by Heger (1994), is a safe RL algorithm that adopts the worst-case criterion. In $\hat{Q}$-learning, the state-value function and action-value function are defined by Equations (4.9a) and (4.9b), respectively.

$$\hat{v}_\pi(s) := \sup \{ r \in \mathbb{R} | p(G_t > r | s_t = s) > 0 \} \tag{4.9a}$$

$$\hat{q}_\pi(s,a) := \max_{s_{t+1}} \{ r_{t+1} + \gamma \hat{v}_\pi(s_{t+1}) | s_t = s, a_t = a \} \tag{4.9b}$$

$\hat{v}_\pi(s)$ is the worst-case return that the agent could receive if it starts from state s and follows

policy π thereafter. For an optimal policy, we have:

$$\hat{q}^{\star}(s,a) = \max_{\pi} \hat{q}_{\pi}(s,a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{4.10a}$$

$$\hat{q}^{\star}(s,a) = \max_{s_{t+1}} \{r_{t+1} + \gamma \hat{v}^{\star}(s_{t+1}) | s_t = s, a_t = a\} \tag{4.10b}$$

where $\hat{v}^{\star}$ is the optimal state-value function.

The $\hat{Q}$-learning algorithm approximates $\hat{q}^{\star}(s,a)$ by calculating $\hat{Q}(s,a)$ through the following update:

$$\hat{Q}(s,a) \leftarrow \min \left\{ \hat{Q}(s,a), r + \gamma \max_{a'} \hat{Q}(s',a') \right\} \tag{4.11}$$

The $\hat{Q}$-value provides a lower bound on the value of action. As a counterpart to Q-learning with respect to the worst-case criterion, $\hat{Q}$-learning is aimed at decision making in situations where avoiding risk is considered to be important (García and Fernández, 2015).

In comparison to the expectation-based criterion, the worst-case criterion may be conservative in some cases because it takes into account severe scenarios which may occur with a small probability. To this end, Gaskett (2003) proposes the β-pessimistic Q-learning as a compromise between the optimism of standard Q-learning and the pessimism of minimax approaches. It adopts the following update by weighting the maximum and the minimum action-values with a parameter β:

$$Q_{\beta}(s,a) \leftarrow Q_{\beta}(s,a) + \alpha \left\{ r + \gamma \left[(1-\beta) \max_{a'} Q_{\beta}(s',a') + \beta \min_{a''} Q_{\beta}(s',a'') \right] - Q_{\beta}(s,a) \right\} \tag{4.12}$$

The β-pessimistic action-values (Q_{β}-values) approximate the expected return of taking an action followed by actions with highest values with probability $1-\beta$ or lowest values with probability β. The weight β can be adjusted to represent different levels of risk-averse attitude. For example, if $\beta = 0$, the update Equation (4.12) is standard Q-learning; if $\beta = 1$, then it is a minimax approach (Gaskett, 2003). β can be interpreted as the probability of a potential malicious adversary attacking the agent in the next state, taking over control, and aiming to minimize the expected return for the agent (Klima et al., 2019).

Gaskett (2003) uses β-pessimistic Q-learning to solve the Cliff Walking task, and finds that it is robust to uncertainty in the action-selection. Following the idea of using a weighted sum of the maximum and minimum for a robust policy, Klima et al. (2019) generalize the algorithm with a κ-operator and propose the Q(κ) and Expected SARSA(κ) methods, of which Q(κ) is comparable to the β-pessimistic Q-learning. They use the κ methods to solve the Cliff Walking and Puddle World tasks, and find that for the more expressive Puddle World task, they outperform classical RL methods. Especially, Q(κ) is superior for all considered levels of probability of attack. Zorowitz et al. (2020) apply the β-pessimistic Q-learning to the domain of computational psychiatry, and use it to model the bias in beliefs. Their results offer novel insights to some phenomena related to anxiety disorders.

The pseudo-code of $\hat{Q}$-learning and β-pessimistic Q-learning are given in Algorithm 4.2. The main difference between these two algorithms and Q-learning is the update of the Q-table, i.e., Equations (4.11), (4.12), and (4.7). It should be noted that for $\hat{Q}$-learning, the Q-table is initialized optimistically, i.e., $Q(s,a) \geq \hat{q}^{\star}(s,a)$. In this work, the performance of the three RL algorithms for supply chain control are examined and compared.

Algorithm 4.2: $\hat{Q}$-learning and β-pessimistic Q-learning

Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$

Initialize $Q(s,a)$, $\forall s,a$, optimistically for $\hat{Q}$-learning
for *each episode* **do**
  Initialize state s
  for *each step of episode* **do**
    Choose a using policy derived from Q, e.g., ε-greedy
    Take action a, observe reward r and new state s'
    ($\hat{Q}$-learning): $Q(s,a) \leftarrow \min\{Q(s,a), r + \gamma \max_{a'} Q(s',a')\}$
    (β-pessimistic Q-learning): $Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma[(1-\beta)\max_{a'} Q(s',a') + \beta \min_{a''} Q(s',a'')] - Q(s,a)\}$
    $s \leftarrow s'$
    Until s is terminal
  end
end

4.3.3 RL-based supply chain control

Figure 4.2 shows the diagram of using RL for supply chain control. Within the RL modelling framework, the supply chain is considered as the stochastic environment and is modelled as a MDP. The agent is the learner and decision maker – the supply chain controller in this case. For RL-based supply chain control, the state and action vectors need to be constructed with proper information. Furthermore, appropriate discretization of the state and action spaces may be required if tabular methods are used. In this work, the state of environment comprises the inventory positions of the supply chain actors, and the action of the agent comprises the order and production quantities. The goal of agent is to minimize the total cost over all time periods, which is the summation of total inventory holding cost and shortage cost. Since the RL algorithms typically maximize the expected return, the reward signal is defined as the negative total cost.

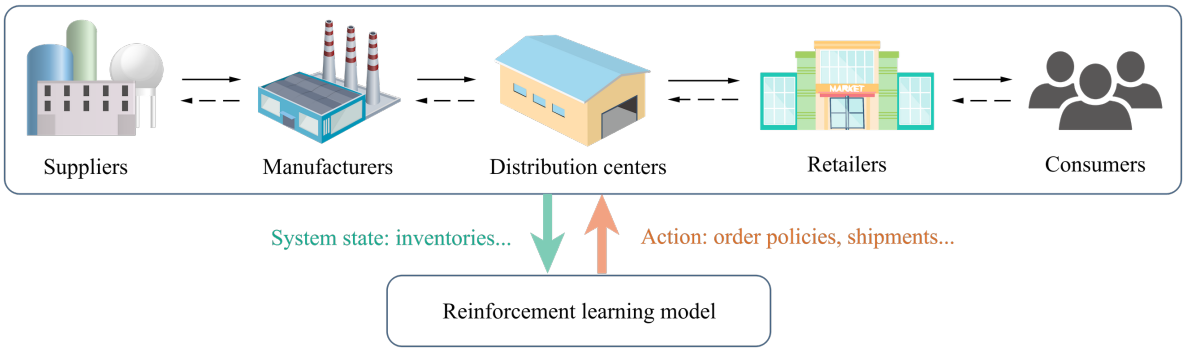


Figure 4.2. RL-based supply chain control.

In the training phase, for each time period, the agent chooses an action, i.e., determines the ordering and production quantities according to its policy. One time step later, the state of supply chain (inventory positions) and the reward signal (total cost for that period) is received by the agent. Then the corresponding action-value in the Q-table is updated by using the RL algorithms. After training for a sufficiently large number of episodes, the algorithm converges and a controller is obtained. In comparison to mathematical programming-based control techniques such as model predictive control and stochastic programming, the Q-learning algorithm has the feature of being model-free. Here, it is worth mentioning that Q-learning is said to be model-free because it does not require an estimate of the Markov

transition model of the environment (Mnih et al., 2015; Shin et al., 2019). Instead, it solves the RL problem directly using information from the environment or a simulator that mimics the environment. In practice, a simulation model is often needed to generate data for training the RL agent (Hubbs et al., 2020; Oroojlooyjadid et al., 2022). In the context of supply chain decision-making, the RL controller can be trained using high-quality supply chain simulation tools first, and then be applied online for real supply chain control.

4.4 Supply Chain Simulation

RL is sometimes said to be a simulation-based optimization approach as the training process often requires simulation of interaction between the agent and the environment. To apply RL to supply chain control, a supply chain simulator is needed to train the controller. Generally, the supply chain simulation tools that are used in the current literature can be classified into three types (Othman and Mustaffa, 2012): (1) spreadsheet simulation (Boute and Lambrecht, 2009); (2) simulation software, e.g., Arena (Giannoccaro and Pontrandolfo, 2002), AnyLogic (Mortazavi et al., 2015), and NetLogo (Jinqi et al., 2017; Aghaie and Heidary, 2019); (3) general purpose programming language, e.g., Java (Li et al., 2010; Dominguez and Framinan, 2013; Dominguez et al., 2018) and Python (Wang et al., 2020).

In order to get flexible customization for different supply chain configurations and functions, in this work a supply chain simulator is developed using the open-source programming language `Python 3.8`. Section 4.4.1 gives a description of the simulator, and the details of simulation procedure are presented in Section 4.4.2.

4.4.1 Supply chain simulator

This work adopts the agent-based modelling (ABM) technique to develop the supply chain simulator. ABM is a well-known paradigm for modelling distributed systems, and has been commonly applied to supply chain modelling (Zhang and Bhattacharyya, 2007; Valluri et al., 2009; Mortazavi et al., 2015; Ponte et al., 2017; Aghaie and Heidary, 2019). Note that the

"agent" in ABM has different meaning from the "agent" in the RL framework. In the ABM framework, the components of a system are modelled as autonomous agents called actors, which have the ability to communicate and make decisions (Mortazavi et al., 2015; Ponte et al., 2017). Thus, for the modelling and simulation of supply chain system, which is complex and often involves nonlinear behaviour, ABM is a good option.

In our simulation model, five supply chain agents are defined, which are:

(1) Manufacturer/Factory agent, which issues production orders and transforms raw materials into products;

(2) Distributor agent, which places orders to the Manufacturer and satisfies orders placed by the Retailer agent;

(3) Retailer agent, which places orders to the Distributor and satisfies the Customer's orders;

(4) Customer agent, which simulates the behaviour of customers;

(5) Remanufacturer agent, which processes used products returned by customers, and delivers remanufactured products to the Retailer. Through defining the Remanufacturer agent, this model can be used to simulate not only forward supply chains, but also CLSCs.

The simulation model is coded in the programming language `Python 3.8` and object-oriented programming is implemented. Each of the five supply chain agents is defined as a class object. The structure of the simulator and the interaction between the agents are shown in Figure 4.3. The Manufacturer, Distributor, and Retailer agents have some attributes in common, such as stock of product on hand, backorder, inventory level, and inventory position. Single product is considered in this work, and the inventory level and inventory position of an agent a are defined in the same way as used by Axsäter (2015):

$$\begin{aligned} IL_{a,t} &= SH_{a,t} - B_{a,t} \\ IP_{a,t} &= SH_{a,t} + OS_{a,t} - B_{a,t} \end{aligned} \tag{4.13}$$

where $IL_{a,t}$, $IP_{a,t}$, and $SH_{a,t}$ are the inventory level, inventory position, and stock on hand of agent a in period t, respectively. They are updated according to the simulation procedure

presented later in Section 4.4.2. The inventory position $IP_{a,t}$ is used in the OUT policy to determine the ordering or production quantity for agent a in period t, as described later in Section 4.5.1. $B_{a,t}$ represents the backorders of agent a in period t, which is the quantity of products that have been demanded by the downstream agent but not yet delivered; $OS_{a,t}$ represents the outstanding orders of agent a in period t, which is the quantity of orders that have been placed to the upstream agent but not yet arrived.

The order and return are also defined as classes, and thus can be used to link the agents and pass information or material between them. These classes are defined in separate Python modules, and can be used to construct simulation model for different supply chains. In this work, we use serial forward flow and reverse flow for example, which means each echelon involves only one agent, as shown in Figure 4.3, while multiple agents can also be simulated using the simulator.

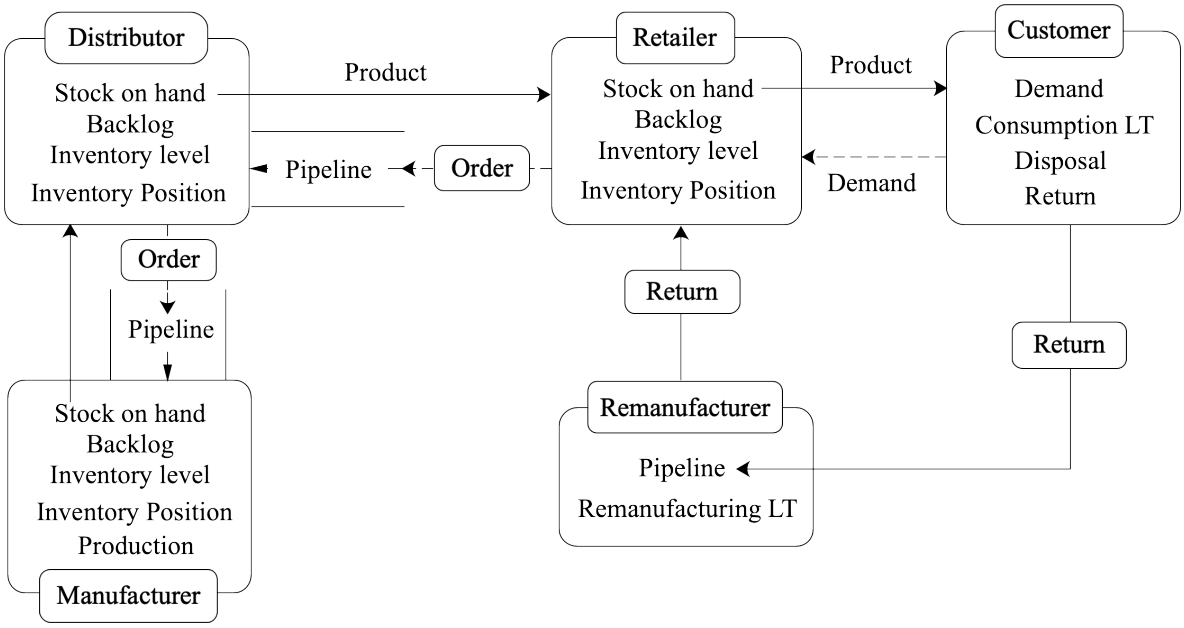


Figure 4.3. Framework of the supply chain simulator.

4.4.2 Simulation procedure

This section describes the details of the five supply chain agents defined in the simulation model. The sequence of actions of a Retailer agent r during time period t is given as follows

(Dominguez et al., 2018; Dominguez et al., 2020):

Step 1. Places an order to the upstream Distributor agent d. An Order object is created and passed to d. The order quantity is determined by an inventory policy, e.g., OUT policy or RL policy.

Step 2. Receives products from d. The update of stock can be understood as $SH_{r,t} \leftarrow SH_{r,t-1} + S_{d,t-\delta_{d,r}}$, where $SH_{r,t}$ is the stock on hand of r in period t, $S_{d,t-\delta_{d,r}}$ is the quantity of shipment from d to r in period $t - \delta_{d,r}$, and $\delta_{d,r}$ is the transportation delay between d and r.

Step 3. Receives finished products from the Remanufacturer agent rm. The update of stock for time period t can be understood as $SH_{r,t} \leftarrow SH_{r,t} + RT_{rm,t-\delta_{rm,r}}$, where $RT_{rm,t-\delta_{rm,r}}$ is the quantity of finished products shipped by rm in period $t - \delta_{rm,r}$, and $\delta_{rm,r}$ is the transportation delay between rm and r. It is assumed that the Remanufacturer follows a push policy, i.e., the used products returned by the Customer agent are processed as soon as they are received. If there is no reverse flow in the supply chain, then this step can be skipped.

Step 4. Satisfies backorders and demand of the Customer agent. The backorders in period $t-1$, $B_{r,t-1}$, are considered prior to the demand in period t, $D_{c,t}$. If current stock on hand $SH_{r,t}$ is sufficient to satisfy $B_{r,t-1}$ and $D_{c,t}$, then products are shipped to fill them immediately; otherwise the unfilled part are recorded as backorders, which causes shortage cost or backlog penalty, and will be processed first once sufficient stock is available. This step can be understood as $(SH_{r,t}, B_{r,t}) \leftarrow (\max\{0, SH_{r,t} - B_{r,t-1} - D_{c,t}\}, \max\{0, B_{r,t-1} + D_{c,t} - SH_{r,t}\})$.

Step 5. Updates inventory profiles. The inventory level and inventory position are updated by $IL_{r,t} \leftarrow SH_{r,t} - B_{r,t}$, and $IP_{r,t} \leftarrow SH_{r,t} + OS_{r,t} - B_{r,t}$, respectively.

The sequence of actions of the Distributor agent d is similar, and is briefly described as follows:

Step 1. Places an order to the upstream Manufacturer agent m.

Step 2. Receives products from m: $SH_{d,t} \leftarrow SH_{d,t-1} + S_{m,t-\delta_{m,d}}$, where $S_{m,t-\delta_{m,d}}$ is the quantity of shipment from m to d in period $t - \delta_{m,d}$, and $\delta_{m,d}$ is the transportation delay between m and d.

Step 3. Satisfies backorders and orders of downstream Retailer r. Distributor d delivers product of quantity $S_{d,t}$ to r to satisfy as many backorders and orders as it can: $S_{d,t} \rightarrow r$. Then the stock on hand is updated by: $SH_{d,t} \leftarrow SH_{d,t} - S_{d,t}$.

Step 4. Updates inventory profiles: $IL_{d,t} \leftarrow SH_{d,t} - B_{d,t}$, $IP_{d,t} \leftarrow SH_{d,t} + OS_{d,t} - B_{d,t}$.

The sequence of actions of the Manufacturer agent m are defined in a similar way, only that in the first step, it determines the quantity of product to be manufactured and issues a production order.

For the Remanufacturer agent rm, the sequence of actions is:

Step 1. Accepts used products that are returned by the Customer agent c: $rm \leftarrow RT_{c,t}$, where $RT_{c,t}$ is the quantity of returned products in period t.

Step 2. Remanufactures products: $RT_{rm,t} \leftarrow RT_{c,t-\delta_{rm}}$, where $RT_{rm,t}$ is the output of remanufacturing process in period t, and δ_{rm} is the remanufacturing lead time.

Step 3. Transports finished products to Retailer: $RT_{rm,t} \rightarrow r$.

The Customer agent c simulates the behavior of consumers, and the sequence of actions is:

Step 1. Passes demand information to Retailer r: $D_{c,t} \rightarrow r$.

Step 2. Returns used products to rm: $RT_{c,t} \leftarrow \alpha_t \hat{D}_{c,t-\delta_c}$, $RT_{c,t} \rightarrow rm$, where $\hat{D}_{c,t-\delta_c}$ is the satisfied demand in period $t - \delta_c$, which is then consumed during the consumption lead time δ_c, and is returned or disposed of in period t; $\alpha_t \in [0, 1]$ is the return rate in period t. The disposal in period t is $(1 - \alpha_t)\hat{D}_{c,t-\delta_c}$.

The complete simulation procedure for a CLSC is presented by Algorithm 4.3 in the form of pseudo-code. For a forward supply chain, the procedure is similar except that the Remanufacturer is excluded.

Algorithm 4.3: Supply chain simulation

Parameters : Number of time periods of simulation N_{step}

Initialize agents
Build forward and reverse flows
for $t \leftarrow 1$ **to** N_{step} **do**
  for *each Retailer* **do**
    Place an order to upstream: $O_{r,t} \rightarrow$ Distributor d
    Receive past order: $SH_{r,t} \leftarrow SH_{r,t-1} + S_{d,t-\delta_{d,r}}$
    Receive return from Remanufacturer: $SH_{r,t} \leftarrow SH_{r,t} + RT_{rm,t-\delta_{rm,r}}$
    Satisfy backorder and Customer demand: $(SH_{r,t}, B_{r,t}) \leftarrow (\max\{0, SH_{r,t} - B_{r,t-1} - D_{c,t}\}, \max\{0, B_{r,t-1} + D_{c,t} - SH_{r,t}\})$
    Update inventory profiles: $IL_{r,t} \leftarrow SH_{r,t} - B_{r,t}$, $IP_{r,t} \leftarrow SH_{r,t} + OS_{r,t} - B_{r,t}$
  end
  for *each Distributor* **do**
    Place an order to upstream: $O_{d,t} \rightarrow$ Manufacturer m
    Receive past order: $SH_{d,t} \leftarrow SH_{d,t-1} + S_{m,t-\delta_{m,d}}$
    Satisfy downstream orders: $S_{d,t} \rightarrow$ Retailer r, $SH_{d,t} \leftarrow SH_{d,t} - S_{d,t}$
    Update inventory profiles: $IL_{d,t} \leftarrow SH_{d,t} - B_{d,t}$, $IP_{d,t} \leftarrow SH_{d,t} + OS_{d,t} - B_{d,t}$
  end
  for *each Manufacturer* **do**
    Issue a production order: $O_{m,t}$
    Receive past production order: $SH_{m,t} \leftarrow SH_{m,t-1} + O_{t-\delta_m}$
    Satisfy downstream orders: $S_{m,t} \rightarrow$ Distributor d, $SH_{m,t} \leftarrow SH_{m,t} - S_{m,t}$
  end
  for *each Remanufacturer* **do**
    Accept return from Customer: $rm \leftarrow RT_{c,t}$
    Remanufacturing process: $RT_{rm,t} \leftarrow RT_{c,t-\delta_{rm}}$
    Transport finished products to Retailer: $RT_{rm,t} \rightarrow r$
  end
  for *each Customer* **do**
    Passes demand to Retailer: $D_{c,t} \rightarrow r$
    Returns used products to Remanufacturer: $RT_{c,t} \leftarrow \alpha_t \hat{D}_{c,t-\delta_c}$, $RT_{c,t} \rightarrow rm$
    Disposal: $(1 - \alpha_t)\hat{D}_{c,t-\delta_c}$
  end
end

A schematic of applying RL to the control of a CLSC is shown in Figure 4.4. A CLSC differs from a forward supply chain in the reverse flow of materials, which needs to be taken into account when determining an optimal control policy. For a classical policy based on demand forecasting, the remanufactured products need to be considered in the calculation of net demand (Dominguez et al., 2020). With the RL framework, the remanufacturing process can be considered implicitly by including it in the environment. The RL-based supply chain controller is the agent, and the supply chain simulator acts as the environment. For each time step, the simulator returns state (inventory positions) and reward (cost) to the controller, and the controller determines and improves its action such as ordering and production quantities. In this work, the RL algorithms are also implemented in `Python`. The Q-table is created as a `pandas.DataFrame` with hierarchical indexing using `pandas.MultiIndex`.

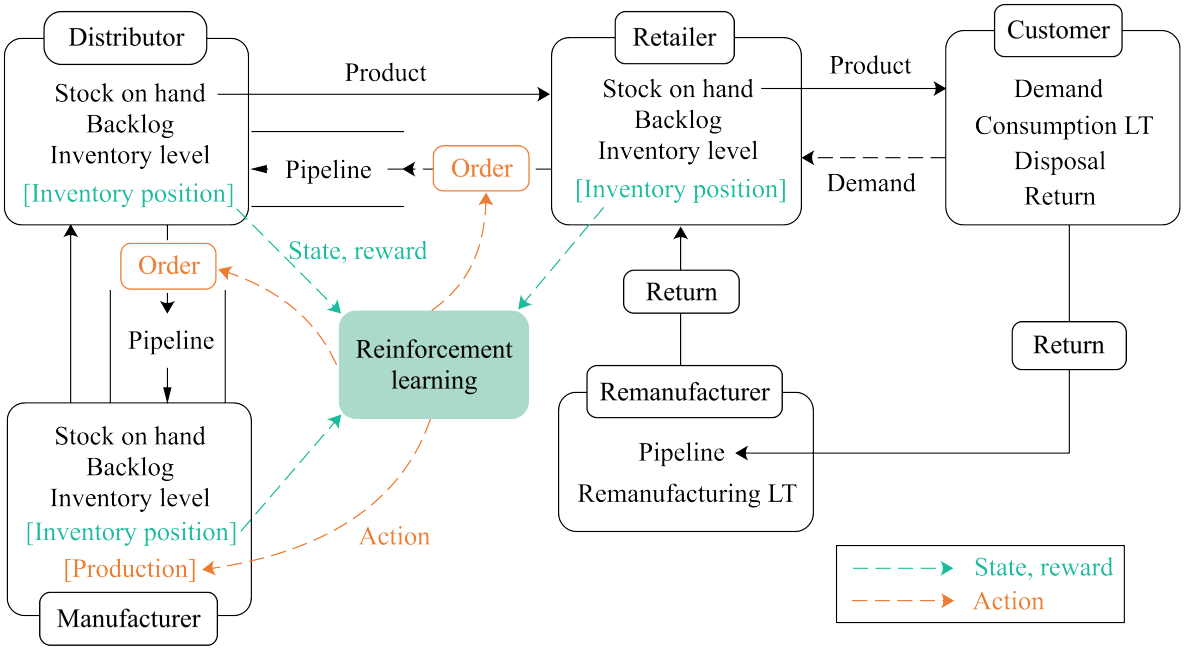


Figure 4.4. RL-based control of CLSC.

4.5 Case Studies

In order to evaluate the performance of the RL policies in supply chain control, two case studies are carried out. A classical Order-Up-To (OUT) policy is chosen as the baseline

inventory policy for comparative study, as it is commonly adopted in the literature and in real-world SCM (Chatfield, 2013; Costantino et al., 2015; Dominguez et al., 2020). The details of OUT policy are described in Section 4.5.1. The first case study involves a two-echelon serial supply chain that consists of a manufacturer and a retailer. The results are presented in Section 4.5.2. The second case study is focused on a CLSC, which does not only have a forward flow of products, but also include a reverse flow of returned products. The results are presented in Section 4.5.3.

4.5.1 Baseline: OUT policy

For the baseline, it is assumed that each supply chain agent employs an adaptive (R, S) policy with periodic review. Here R is the review period, and S is the desired OUT level that is updated in every time period. In this work, $R = 1$ and the ordering quantity of each agent in period t is determined according to the following procedure (Chatfield et al., 2004; Chatfield, 2013; Dominguez et al., 2020):

(1) Demand and lead times forecasting. The mean and variance of demand are estimated using the τ-period moving average and variance techniques, respectively:

$$\overline{D}_{a,t} = \frac{1}{\tau} \sum_{k=1}^{\tau} D_{a,t-k} \tag{4.14a}$$

$$s^2_{D_{a,t}} = \frac{1}{\tau - 1} \sum_{k=1}^{\tau} \left(D_{a,t-k} - \overline{D}_{a,t} \right)^2 \tag{4.14b}$$

where τ is the forecasting period (Dominguez et al., 2018; Dominguez et al., 2020), and $\overline{D}_{a,t}$ and $s^2_{D_{a,t}}$ are the estimated mean and variance of demand at agent $a \in \{r, d, m\}$ in period t, respectively. For Retailer r, $D_{r,t}$ is the Customer's demand in period t, i.e., $D_{r,t} = D_{c,t}$; for a CLSC where a remanufacturer rm exists, $D_{r,t}$ is the net demand in period t, i.e., $D_{r,t} = D_{c,t} - RT_{rm,t}$. For Distributor d, $D_{d,t}$ is the quantity of order placed by Retailer r in period t, i.e., $D_{d,t} = O_{r,t}$; for Manufacturer m, $D_{m,t}$ is the quantity of order placed by Distributor d in period t, i.e., $D_{m,t} = O_{d,t}$.

In case where the lead times are stochastic, they can be estimated through the running average and variance technique using all previous data available. The estimated mean and variance of the ordering or production lead time at agent a in period t is denoted as $\overline{\delta}_{a,t}$ and $s^2_{\delta_{a,t}}$, respectively.

(2) Determine the OUT level $OL_{a,t}$ of agent a in period t:

$$OL_{a,t} = (\overline{\delta}_{a,t} + R)\overline{D}_{a,t} + z\sqrt{(\overline{\delta}_{a,t} + R)s^2_{D_{a,t}} + \overline{D}^2_{a,t}s^2_{\delta_{a,t}}} \tag{4.15}$$

where z is the safety factor. Assuming that the demand during the lead time follows a normal distribution, $z = 2$ corresponds to a customer service level of 97.72%, and $z = 3.09$ corresponds to a customer service level of 99.9%. These two levels are commonly adopted in the literature (Chatfield et al., 2004; Chatfield, 2013; Cannella et al., 2016; Dominguez et al., 2018; Dominguez et al., 2020), and thus are used as the baselines in this work.

(3) Determine the ordering or production quantity $O_{a,t}$ of agent a in period t:

$$O_{a,t} = \max\{0, OL_{a,t} - IP_{a,t-1}\} \tag{4.16}$$

When inventory position $IP_{a,t-1}$ is above the OUT level $OL_{a,t}$, agent a does not place any order in period t; otherwise, agent a places an order with the quantity of $OL_{a,t} - IP_{a,t-1}$.

4.5.2 Case study I: A two-echelon serial supply chain

A two-echelon serial supply chain for single product is used for the first case study. This supply chain consists of a factory and a retailer, and its configuration is shown in Figure 4.5. This example can be seen as a simplified version of the well-known Beer Distribution Game (BDG) (Sterman, 1989; Chen and Samroengraja, 2000). The BDG supply chain is a benchmark four-echelon serial supply chain, which deals with the production and distribution of beer product and consists of a factory, a distributor, a warehouse and a retailer. This simulation example has been widely used in operation management to illustrate the *bullwhip effect* in

supply chains, and has also been used as an example application for supply chain control (Chaharsooghi et al., 2008; Fuji et al., 2018; Oroojlooyjadid et al., 2022). Here, two echelons of the BDG supply chain are taken to construct a simpler example: the factory and the retailer.

Figure 4.5. Case study I: a two-echelon serial supply chain.

In each time period, the Customer agent c purchases products from the Retailer agent r. The Retailer places an order to the Factory m according to its ordering policy. The Factory determines the production quantity and fills Retailer's orders. Customer demand $D_{c,t}$ is the uncertain parameter in this case. It is assumed that $D_{c,t}$ is subject to a normal distribution, as this assumption is widely adopted in research studies, e.g., Chatfield et al. (2004) and Dominguez et al. (2020). The mean of demand is set to $\mu_{D_{c,t}} = 30$ and the coefficient of variation (c.v.) $\sigma_{D_{c,t}} / \mu_{D_{c,t}}$ is set to 20%, i.e., $D_{c,t} \sim \mathcal{N}(30, 6^2)$. Since the demand is always non-negative, in simulation it is sampled from a truncated normal distribution using the `scipy.stats.truncnorm()` application programming interface (API) provided by the `SciPy` library to avoid negative values. The manufacturing lead time of the Factory is set to $\delta_m = 4$, and the transportation delay between the Factory and Retailer is set to $\delta_{m,r} = 2$.

Two types of cost are considered in this case study, inventory holding cost C_{inv} and shortage cost (backlog penalty) C_{bkl}. The unit inventory holding cost $C_{a,inv}$ and unit shortage cost $C_{a,bkl}$ for agent $a \in \{m, r\}$ are set to have a ratio of $1:10$, i.e., $C_{a,inv} = 1$ per unit per period, and $C_{a,bkl} = 10$ per unit per period (Chen and Samroengraja, 2000; Kwak et al., 2009). C_{inv} and C_{bkl} are calculated as:

$$C_{inv} = \sum_t \sum_{a \in \{m,r\}} C_{a,inv} SH_{a,t}, \quad C_{bkl} = \sum_t \sum_{a \in \{m,r\}} C_{a,bkl} B_{a,t} \tag{4.17}$$

The total cost is the summation of inventory holding cost and backlog penalty: $C_{tot} =$

$C_{inv} + C_{bkl}$. The goal is to obtain an optimal ordering policy that minimizes the total cost by determining the ordering quantity of the Retailer and the production quantity of the Factory. It is obvious that if the two agents hold excess safety stock, then the total inventory holding cost will be high. On the other hand, if their inventory levels are too low, then shortage could occur and lead to high backlog penalty. Thus, a good policy should find suitable average inventory and achieve a balance between the two types of cost.

The performance of the RL policies is examined and compared with that of the baseline policy. For the baseline OUT policy, two sets of parameters are used, which are a forecasting period $\tau = 15$ with a safety factor $z = 2$ or $z = 3.09$ (Dominguez et al., 2020). The two baselines are denoted as $OUT^{z=2}_{\tau=15}$ and $OUT^{z=3.09}_{\tau=15}$, respectively. For the RL models, the state comprises inventory positions of the two agents: $s_t = [IP_{r,t}, IP_{m,t}]$. $IP_{r,t}$ is discretized into 8 intervals, which are $[-\infty, 0)$, $[0, 20)$, $[20, 40)$, $[40, 60)$, $[60, 80)$, $[80, 100)$, $[100, 120)$, $[120, \infty)$, and coded as -1, 0, 1, 2, 3, 4, 5, 6, respectively. $IP_{m,t}$ is discretized into 10 intervals, which are $[-\infty, 0)$, $[0, 20)$, $[20, 40)$, $[40, 60)$, $[60, 80)$, $[80, 100)$, $[100, 120)$, $[120, 140)$, $[140, 160)$, $[160, \infty)$, and coded as -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, respectively. Thus, the system has $8 \times 10 = 80$ different states. The action $a_t = [O_{r,t}, O_{m,t}]$ consists of the order quantity of Retailer in period t, $O_{r,t}$, and the production quantity of Factory in period t, $O_{m,t}$. For both Retailer and Factory, 7 actions are available for each state, i.e., $O_{r,t}, O_{m,t} \in \{0, 10, 20, 30, 40, 50, 60\}$. The 7 actions are coded as $0, 1, 2, 3, 4, 5, 6$, respectively. Thus, there are $7^2 = 49$ different actions available for each state, and the dimension of the Q-table is 80×49. The objective is to minimize the total cost while a RL algorithm maximizes the reward, thus the reward signal in period t is calculated as the negative total cost for period t:

$$r_t = -\sum_{a \in \{m,r\}} C_{a,inv} SH_{a,t} - \sum_{a \in \{m,r\}} C_{a,bkl} B_{a,t} \tag{4.18}$$

The RL algorithms examined include Q-learning, $\hat{Q}$-learning, and β-pessimistic Q-learning. For β-pessimistic Q-learning, three values of β are considered, which are $\beta = 0.1, 0.3, 0.5$, and the algorithms are denoted as $Q_{\beta=0.1}$, $Q_{\beta=0.3}$, and $Q_{\beta=0.5}$, respectively. The discount rate is set to $\gamma = 0.95$. The learning rate is set to $\alpha = 0.1$. For the ε-greedy policy, ε is

set to 0.1 for the probability of exploration. Since supply chain operation is a continuing task rather than an episodic task, there is no terminal state for an episode. Therefore, each episode is set to have $N_{step} = 100$ steps. All of the five RL models are trained for the same number of episodes, i.e., 20,000 episodes. The simulation and training procedure is presented in Algorithm 4.4 in the form of pseudo-code.

Algorithm 4.4: Training RL controllers

Parameters: number of steps per episode N_{step}, number of episodes N_{eps}, learning rate α, discount rate γ, weight β

Initialize supply chain simulator
Initialize Q-table: $Q(s,a) \leftarrow 0, \forall s, a$
for $n \leftarrow 1$ **to** N_{eps} **do**
  Initialize state $\tilde{s}_0 = [IP_{r,0}, IP_{m,0}]$; $s_0 \leftarrow encode(\tilde{s}_0)$
  Sample uncertainty: $\omega = [D_{c,1}, D_{c,2}, \ldots, D_{c,N_{step}}] \leftarrow$ `scipy.stats.truncnorm()`
  for $t \leftarrow 1$ **to** N_{step} **do** `/* ε-greedy */`
    if *numpy.random()* $< 1 - \varepsilon$ **then** `/* exploitation */`
      Choose action $a_t \in \{a | Q(s_{t-1}, a) = \max_{a' \in \mathcal{A}(s_{t-1})} Q(s_{t-1}, a')\}$ with a random tiebreaker
    else `/* exploration */`
      Choose action $a_t \in \mathcal{A}(s_{t-1})$ randomly
    end
    Agent takes action a_t: $a_t \rightarrow$ simulator
    Run simulator for 1 step with $\tilde{s}_{t-1}, a_t, \omega$
    Observe state: $\tilde{s}_t = [IP_{r,t}, IP_{m,t}]$; $s_t \leftarrow encode(\tilde{s}_t)$
    Calculate reward: $r_t \rightarrow$ agent
    `/* Update Q-table: */`
    if *using Q-learning* **then**
      $Q(s_{t-1}, a_t) \leftarrow Q(s_{t-1}, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_t, a') - Q(s_{t-1}, a_t) \right]$
    else if *using* $\hat{Q}$*-learning* **then**
      $Q(s_{t-1}, a_t) \leftarrow \min \{ Q(s_{t-1}, a_t), r_t + \gamma \max_{a'} Q(s_t, a') \}$
    else if *using* β*-pessimistic Q-learning* **then**
      $Q(s_{t-1}, a_t) \leftarrow Q(s_{t-1}, a_t) +$
      $\alpha \{ r_t + \gamma [(1-\beta) \max_{a'} Q(s_t, a') + \beta \min_{a''} Q(s_t, a'')] - Q(s_{t-1}, a_t) \}$
  end
end

After the RL controllers are trained, they are compared with the baselines to evaluate their performance. A main objective of this study is to examine the robustness of the RL policies

to uncertainty in the environment. Hubbs et al. (2020) show that the performance of a RL policy drops dramatically when the demand distribution changes, because the model-free RL is not capable enough to respond to large changes in the environment. In this work, 3 different environment settings are designed for testing, which are:

(1) Normal environment setting, where the distribution of demand in testing data is the same as that in training process, i.e., $D'_{c,t} \sim \mathcal{N}(30, 6^2)$.

(2) A change in the demand pattern, where the mean of demand is assumed to increase by 10% and the coefficient of variation remains the same, i.e., $D'_{c,t} \sim \mathcal{N}(33, 6.6^2)$ in testing data. This setting is designed to evaluate the robustness of policies to unexpected change of uncertainty in the environment.

(3) A larger change in the demand pattern, where the mean of demand is assumed to increase by 20% and the coefficient of variation remains the same, i.e., $D'_{c,t} \sim \mathcal{N}(36, 7.2^2)$. The deviation of testing environment from training environment in this setting is larger than that in setting (2).

Figure 4.6 shows the comparison of inventory levels and accumulated cost under a random demand scenario under the normal environment setting. It can be seen that during the first 10 time periods, the Retailer and Factory experience a stockout. Then their inventory levels increase and the system reaches a steady state after around 20 time periods. Thus, in this case, the first 20 time periods are taken as the so-called warm-up period (Cannella et al., 2016; Dominguez et al., 2018; Dominguez et al., 2019; Dominguez et al., 2020), which are used to set up the system and not taken into account when calculating the accumulated cost. In this scenario, Retailers using the safe RL policies have higher inventory level than those using OUT and Q-learning policies.

Since using only 1 scenario is not sufficient to evaluate the performance of policies, statistical analysis is needed. For each environment setting, 200 random demand scenarios are generated and simulated, and the costs are collected. Figure 4.7 shows the comparison of different policies under 3 environment settings. The results are presented using box plots. A box plot is a statistical plot that aims to display the main features of a dataset, such as the sample median,

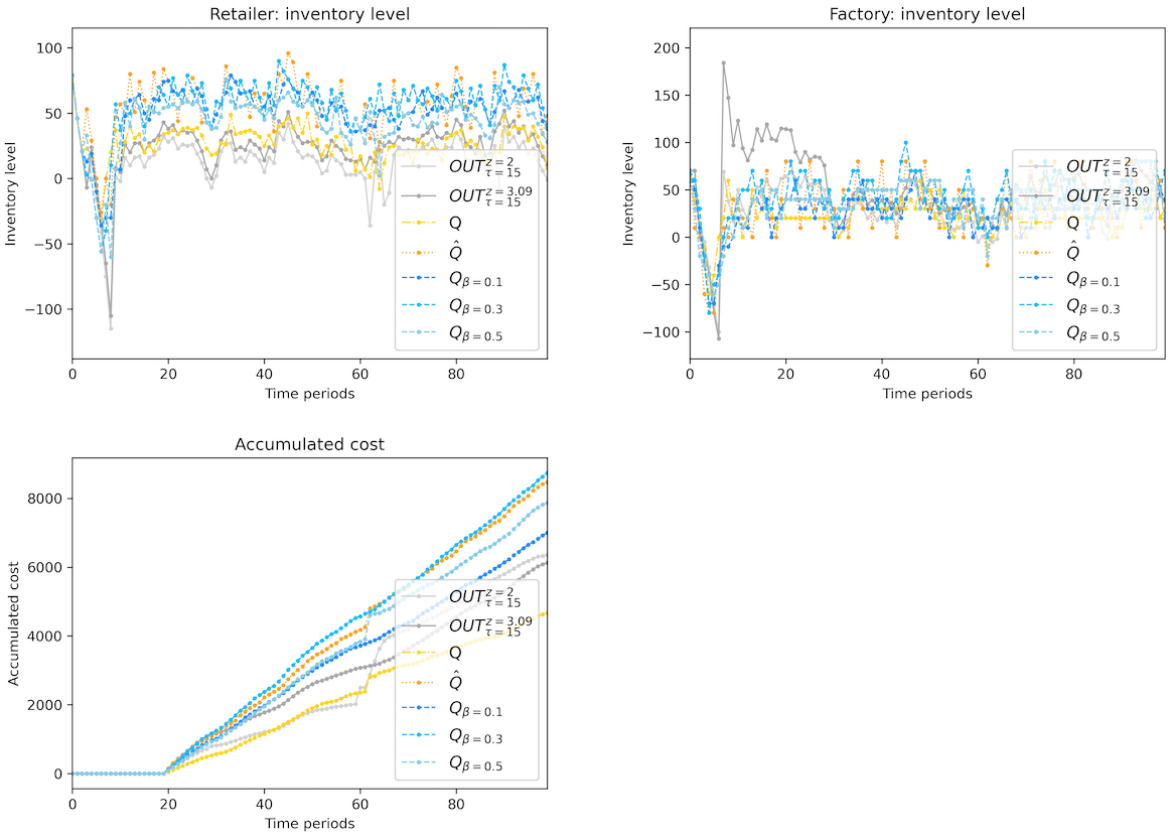


Figure 4.6. Case study I: comparison of inventory level and total cost.

the first quartile or 25th percentile Q_1, the third quartile or 75th percentile Q_3 (Frigge et al., 1989). The interquartile range (IQR) is $Q_3 - Q_1$, and half of the data are contained by the box. Two whiskers are determined based on $Q_1 - k * IQR$ and $Q_3 + k * IQR$. A $k = 1.5$ is used in this work, and any observation outside the whiskers is regarded as an outlier. A box plot is suitable for comparing the metrics of different algorithms under different scenarios, and thus has been used to compare supply chain control techniques by, e.g., Cui et al. (2017) and Rajendran and Ravi Ravindran (2019). In this work, the box plots are created using the `seaborn.boxplot()` API provided by `seaborn`, which is a `Python` library for data visualization.

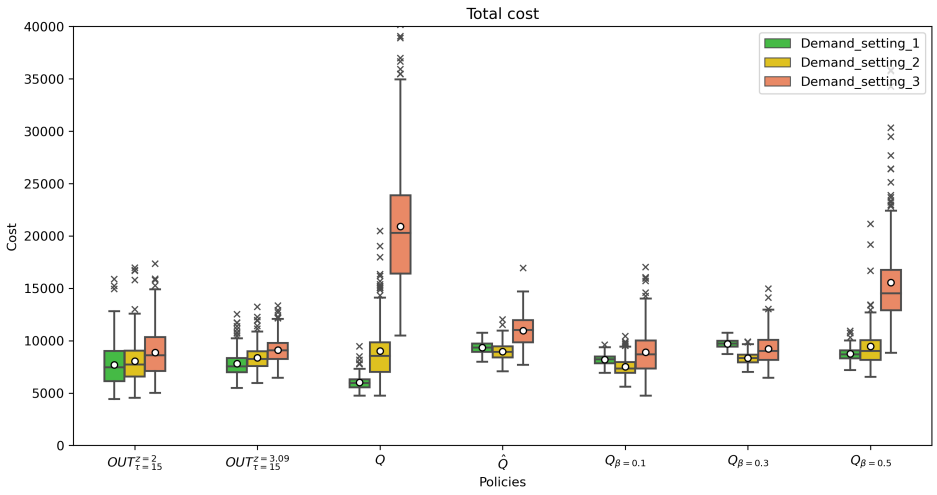
The results of total cost, inventory holding cost, and shortage cost are shown in Figures 4.7a, 4.7b, and 4.7c, respectively. The environment settings (1), (2), and (3) are colored in green, yellow, and red, respectively. It is observed from Figure 4.7a that as the demand average increases, the median, minimum, and maximum of total cost using OUT policies and Q-learning increase. The variance using OUT policies is not affected by the demand change, while that using $OUT^{z=3.09}_{\tau=15}$ is smaller than that using $OUT^{z=2}_{\tau=15}$. By contrast, the median, minimum, and maximum of total cost using $\hat{Q}$-learning, $Q_{\beta=0.1}$-learning, and $Q_{\beta=0.3}$-learning are lowest in setting (2) and highest in setting (3). Among all the policies, Q-learning has the best performance under the normal setting in terms of the statistics of total cost. However, it is the most sensitive to changes in the demand setting, because the median, minimum, maximum, and variance of total cost increase on a large scale in settings (2) and (3), in comparison with setting (1). Thus, it is not as robust to changes in the uncertainty as the safe RL policies are.

The total cost consists of inventory holding cost and shortage cost. Figure 4.7b shows that the median, minimum, and maximum of inventory holding cost using OUT policies increase as the demand average increases. By contrast, those statistics decrease for all the RL policies. A potential reason is that when determining the ordering quantities, the OUT policies take into account the demand change by demand forecasting. However, the RL agent does not have demand information, and the ordering quantities do not change when demand pattern changes. Therefore, the inventory levels and inventory cost become lower when demand increases. It can be seen from Figure 4.7c that under the normal setting, the $OUT^{z=2}_{\tau=15}$ policy

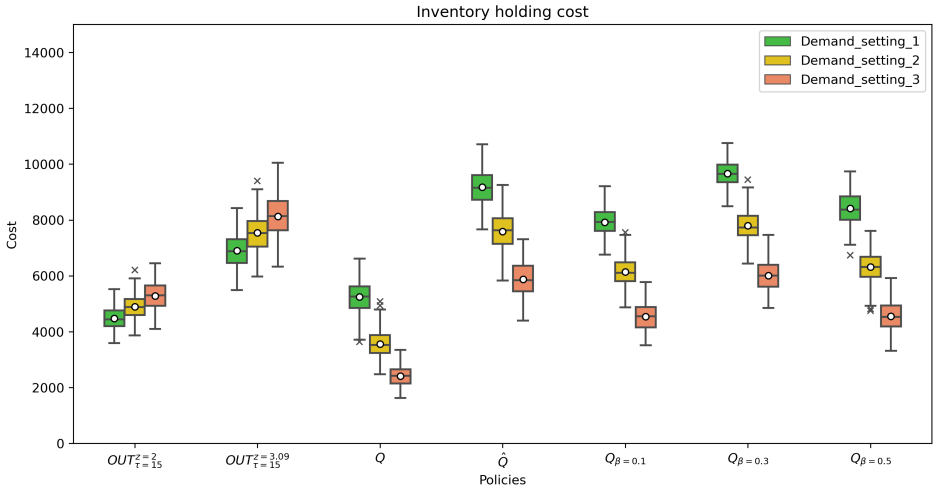
has the highest shortage cost, while the $OUT_{\tau=15}^{z=3.09}$ policy that uses a larger safety factor reduces the shortage. The variance using OUT policies is not affected by the demand setting, while that using $OUT_{\tau=15}^{z=3.09}$ is smaller than that using $OUT_{\tau=15}^{z=2}$. The safe RL policies achieve a lower shortage cost than OUT and Q-learning policies. Q-learning is the most sensitive to changes in the demand setting, as the statistics of shortage cost increase by a large amount in settings (2) and (3), compared with setting (1). The variance in the shortage cost contributes to that of the total cost. By comparison, safe RL policies are more robust to changes in the environment.

In order to explore the reason for the difference between RL policies in robustness, the state-action pairs under the policies are investigated. Since the supply chain is two-echelon and the state vector is two-dimensional, it is possible to visualize the state-action pairs of a RL policy. This work adopts the heat map for visualization and comparison, as shown in Figure 4.8. The heat maps are created using the `seaborn.heatmap()` API. Figure 4.8a displays the state-action pairs of the Retailer (left) and Factory (right) under the Q-learning policy. There are 8 states for the Retailer and 10 states for the Factory, therefore the heat map are gridded by 8×10. Different actions are plotted in different colors, and the legend is shown on the right to the figure. The state-action pairs of $\hat{Q}$-learning and $Q_{\beta=0.1}$-learning are displayed in Figure 4.8b and 4.8c, respectively. It is observed that in general, $\hat{Q}$-learning generates larger order quantity for the Retailer than Q-learning does, for example, for state $[IP_r = 4, IP_m]$ (the 6th column of heat map). This results in higher safety stock and inventory level for $\hat{Q}$-learning, and thus it is more robust to demand increase.

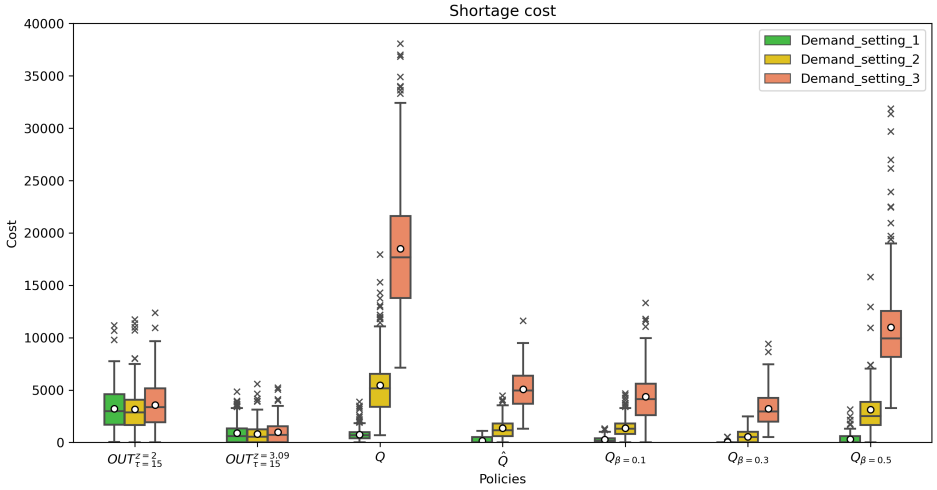
The bullwhip effect using different policies is also investigated in this work. Here, the Bullwhip Slope (BwSl) metric is adopted to quantify the bullwhip effect in a supply chain. The BwSl metric proposed by Cannella et al. (2013) has been used to study supply chain dynamics by, e.g., Dominguez et al. (2015), Dominguez et al. (2018), and Tombido et al. (2020). BwSl is defined based on an echelon-level metric Order Variance Ratio (OVR) that measures the amplification of order variability. The OVR of echelon e is defined as the ratio between the variance of the orders placed by echelon e and the variance of customer demand. Under the assumption that all customer demands are independent and each node places orders



(a) Total cost.

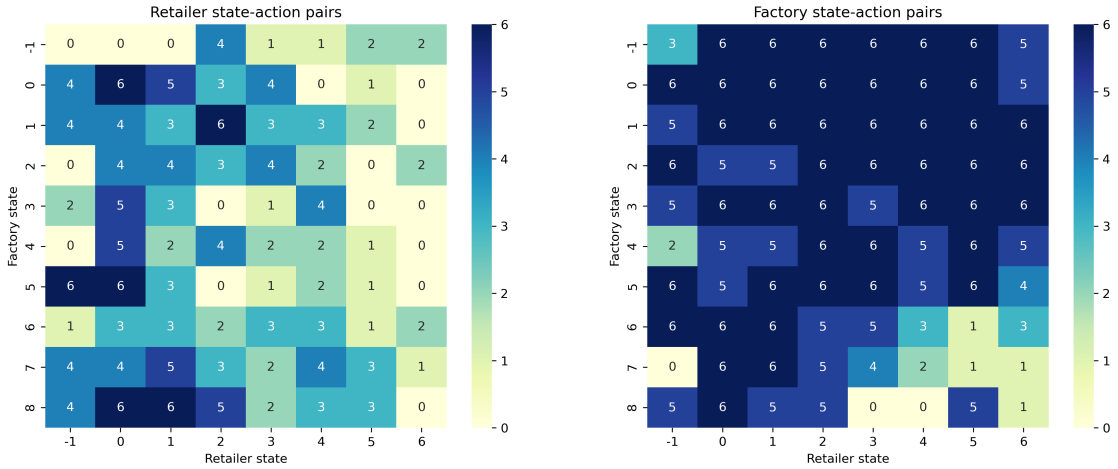


(b) Inventory holding cost.

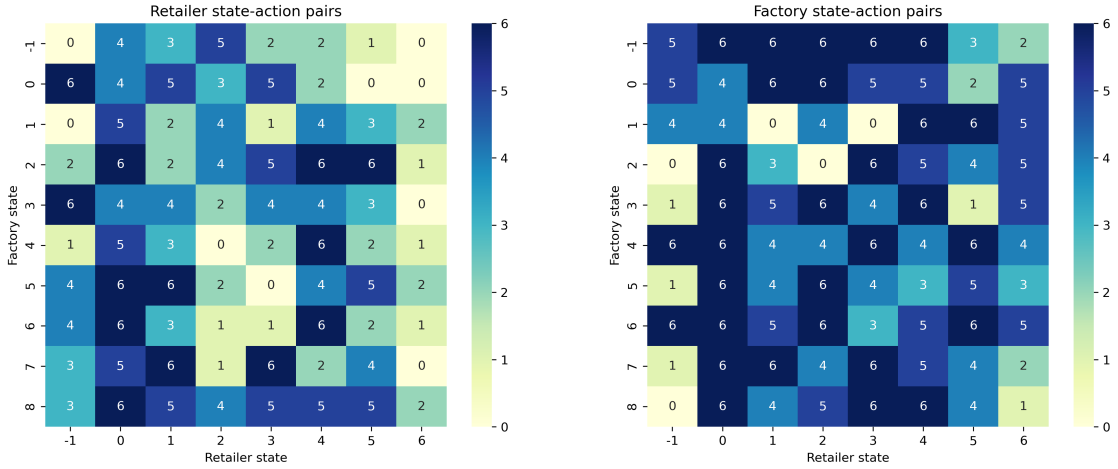


(c) Shortage cost.

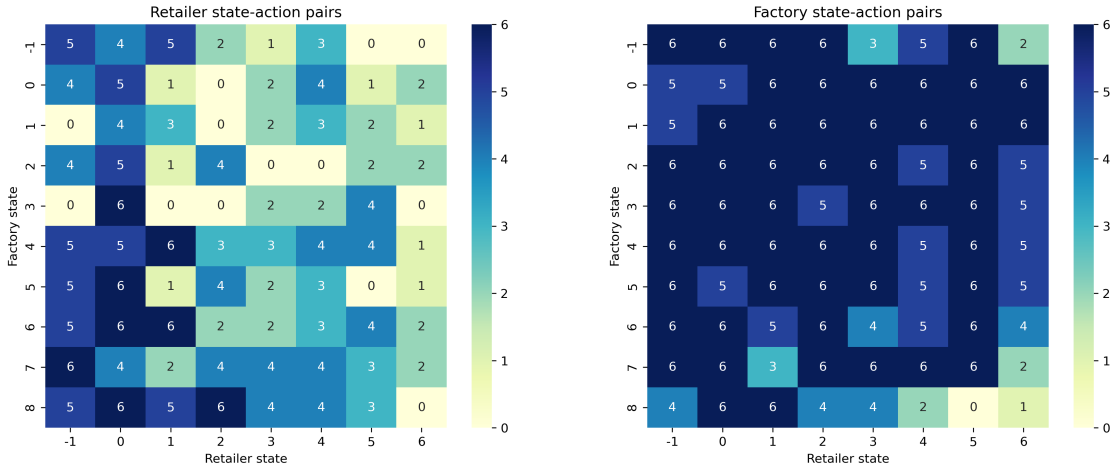
Figure 4.7. Case study I: comparison of costs under different environment settings.



(a) Q-learning.



(b) $\hat{Q}$-learning.



(c) $Q_{\beta=0.1}$-learning.

Figure 4.8. Case study I: state-action pairs.

independently, the OVR of echelon e can be expressed as (Tombido et al., 2020):

$$OVR_e = \frac{s^2_{O_e}}{s^2_D} = \frac{\sum_j s^2_{O_{e,j}}}{\sum_c s^2_{D_c}} \tag{4.19}$$

where $O_e = \sum_j O_{e,j}$ is the aggregate order of echelon e, $O_{e,j}$ represents the order quantity of agent j in echelon e, and $s^2_{O_{e,j}}$ is the variance of $O_{e,j}$; $s_D = \sum_c s_{D_c}$ is the aggregate customer demand, and $s^2_{D_c}$ is the variance of customer c's demand D_c.

OVR_e is said to be the bullwhip suffered by echelon e (Coppini et al., 2010). The BwSl for a supply chain is defined as the slope of the linear regression of OVR value on echelon position:

$$BwSl = \frac{E\sum_{e=1}^{E} p_e OVR_e - \sum_{e=1}^{E} p_e \sum_{e=1}^{E} OVR_e}{E\sum_{e=1}^{E} p_e^2 - (\sum_{e=1}^{E} p_e)^2} \tag{4.20}$$

where p_e is the position of the e-th echelon in Dejonckheere et al. (2004)'s curve, and E is the total number of echelons. BwSl quantifies the bullwhip propagation through the supply chain. It is a system-level metric and allows for comprehensive comparison between different supply chains. A low BwSl indicates smooth propagation of the bullwhip effect, while a high BwSl indicates fast propagation.

The comparison of bullwhip slope under different demand scenarios in the normal settings is presented in Figure 4.9. It is observed that except for $Q_{\beta=0.3}$-learning that achieves the lowest of median, minimum, maximum, and variance of BwSl, there is no obvious advantage in RL policies over the baselines.

4.5.3 Case study II: A closed-loop supply chain

The second case study is focused on a closed-loop supply chain. The supply chain configuration is adapted from Dominguez et al. (2020) and is shown in Figure 4.10. This supply chain consists of a Manufacturer m, a Distributor d, a Retailer r, and a Remanufacturer rm. The forward flow deals with production and distribution of products to Customer c, which is $m \rightarrow d \rightarrow r \rightarrow c$. A main difference between this case study and the first one is the

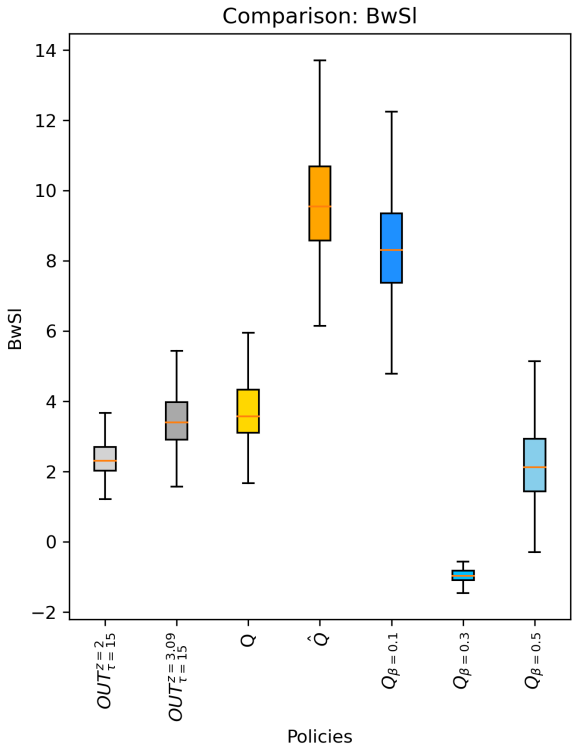


Figure 4.9. Case study I: comparison of bullwhip slope.

existence of a reverse flow, which is $c \rightarrow rm \rightarrow r$ and deals with remanufacturing and return of used products.

During each time period, the Customer agent purchases products from the Retailer agent, and returns used products to the Remanufacturer agent. The Remanufacturer receives used products and sends finished products to the Retailer. The Retailer places an order to the Distributor. The Distributor fills orders of the Retailer and places an order to the Manufacturer. The Manufacturer determines its production quantity and fills the Distributor's orders. In this case, the uncertain parameters are the demand and return rate of the Customer.

The supply chain parameters are listed in Table 4.2. It is assumed that the demand $D_{c,t}$ and return rate $\alpha_{c,t}$ follow normal distributions. The mean and coefficient of variation of $D_{c,t}$ are set to $\mu_{D_{c,t}} = 50$ and $\sigma^2_{D_{c,t}} = 20\%$, respectively, i.e., $D_{c,t} \sim \mathcal{N}(50, 10^2)$. The mean and coefficient of variation of $\alpha_{c,t}$ are set to $\mu_{\alpha_{c,t}} = 0.5$ and $\sigma^2_{\alpha_{c,t}} = 20\%$, respectively, i.e., $\alpha_{c,t} \sim \mathcal{N}(0.5, 0.1^2)$. A truncated normal distribution is used in the simulation for non-negative samples. The customer consumption lead time is set to $\delta_c = 32$. The manufacturing

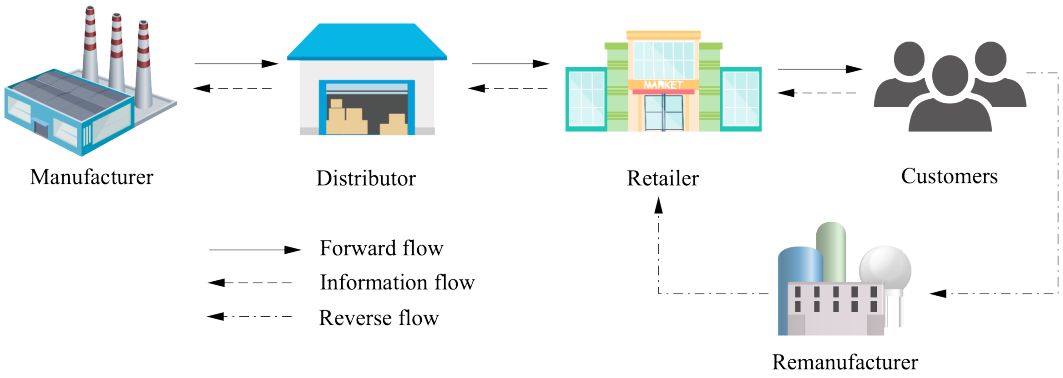


Figure 4.10. Case study II: a closed-loop supply chain.

lead time of the Manufacturer, δ_m, and the remanufacturing lead time of the Remanufacturer, δ_{rm}, are set to 4. The transportation delay between the Manufacturer and Distributor, $\delta_{m,d}$, and the transportation delay between the Distributor and Retailer, $\delta_{d,r}$, are set to 2.

Table 4.2. Case study II: supply chain parameters.

Parameters	Values
Demand average $\mu_{D_{c,t}}$	50 units per period
Demand variance $\sigma^2_{D_{c,t}}$	10^2
Return rate average $\mu_{\alpha_{c,t}}$	0.5
Return rate variance $\sigma^2_{\alpha_{c,t}}$	0.1^2
Review period R	1
Transportation lead time $\delta_{m,d}, \delta_{d,r}$	2
Manufacturing lead time δ_m	4
Remanufacturing lead time δ_{rm}	4
Consumption lead time δ_c	32
Inventory holding cost $C_{a,inv}, \forall a \in \{m,d,r\}$	1 per unit per period
Shortage cost $C_{a,bkl}, \forall a \in \{m,d,r\}$	10 per unit per period

Inventory holding and shortage costs are considered. The unit inventory holding cost is $C_{a,inv} = 1$ per unit per period, and the unit shortage cost is $C_{a,bkl} = 10$ per unit per period, for $a \in \{m,d,r\}$. The total cost is the summation of inventory holding cost and shortage cost:

$$C_{tot} = \sum_t \sum_{a\in\{m,d,r\}} C_{a,inv} SH_{a,t} + \sum_t \sum_{a\in\{m,d,r\}} C_{a,bkl} B_{a,t} \tag{4.21}$$

The goal is to obtain an optimal ordering policy that minimizes the total cost by determining

the ordering quantities of the Retailer and Distributor, and the production quantity of the Manufacturer. The performance of the RL policies is examined and compared with that of a baseline OUT policy. For the OUT policy, two sets of parameters are considered, which are a forecasting period $\tau = 15$ with a safety factor $z = 2$ or $z = 3.09$. The two baselines are denoted as $OUT_{\tau=15}^{z=2}$ and $OUT_{\tau=15}^{z=3.09}$, respectively.

For the RL models, the state comprises inventory positions of Retailer, Distributor, and Manufacturer, i.e., $s_t = [IP_{r,t}, IP_{d,t}, IP_{m,t}]$. $IP_{r,t}$, $IP_{d,t}$, and $IP_{m,t}$ are all discretized into 10 intervals. The intervals and corresponding codes are listed in Table 4.3. Thus, the system has a total of $10^3 = 1000$ different states. The action of the agent is $a_t = [O_{r,t}, O_{d,t}, O_{m,t}]$, where $O_{r,t}$ and $O_{d,t}$ are the ordering quantities of the Retailer and Distributor in period t, respectively; and $O_{m,t}$ is the production quantity of the Manufacturer in period t. For the three supply chain agents, 7 actions are available for each state, i.e., $O_{r,t}, O_{d,t}, O_{m,t} \in \{0, 10, 20, 30, 40, 50, 60\}$. The 7 actions are coded as $0, 1, 2, 3, 4, 5, 6$, respectively. Therefore, there are a total of $7^3 = 343$ different actions available for each state, and the dimension of the Q-table is 1000×343. The reward signal in period t is the negative total cost for period t:

$$r_t = -\sum_{a\in\{m,d,r\}} C_{a,inv} SH_{a,t} - \sum_{a\in\{m,d,r\}} C_{a,bkl} B_{a,t} \tag{4.22}$$

Table 4.3. Case study II: CLSC, states and codes.

$IP_{r,t}$	$IP_{d,t}$	$IP_{m,t}$	Code
$[-\infty, 0)$	$[-\infty, 0)$	$[-\infty, 0)$	-1
$[0, 20)$	$[0, 20)$	$[0, 25)$	0
$[20, 40)$	$[20, 40)$	$[25, 50)$	1
$[40, 60)$	$[40, 60)$	$[50, 75)$	2
$[60, 80)$	$[60, 80)$	$[75, 100)$	3
$[80, 100)$	$[80, 100)$	$[100, 125)$	4
$[100, 120)$	$[100, 120)$	$[125, 150)$	5
$[120, 140)$	$[120, 140)$	$[150, 175)$	6
$[140, 160)$	$[140, 160)$	$[175, 200)$	7
$[160, \infty)$	$[160, \infty)$	$[200, \infty)$	8

The RL algorithms examined include Q-learning, $\hat{Q}$-learning, and β-pessimistic Q-learning. For β-pessimistic Q-learning, three values of β are considered, which are $\beta = 0.1, 0.3, 0.5$.

The discount rate and learning rate are set to $\gamma = 0.95$ and $\alpha = 0.1$, respectively. For the ε-greedy policy, ε is set to 0.1 for the probability of exploration. Each episode is set to have $N_{step} = 400$ steps. All of the five RL models are trained for the same number of episodes, i.e., 200,000 episodes. The simulation and training procedure is similar to Algorithm 4.4.

After the RL controllers are trained, they are compared with the baselines to test their performance and robustness to uncertainty in the environment. Three different environment settings are considered, which are:

(1) Normal environment, where the distribution of demand in testing data is the same as in training data, i.e., $D'_{c,t} \sim \mathcal{N}(50, 10^2)$.

(2) A change in the demand pattern, where the mean of demand is assumed to increase by 10% and the coefficient of variation remains the same, i.e., $D'_{c,t} \sim \mathcal{N}(55, 11^2)$. The purpose of this setting is to evaluate the robustness of policies to unexpected change of uncertainty in the environment.

(3) A larger change in the demand pattern, where the mean of demand is assumed to increase by 20% and the coefficient of variation remains the same, i.e., $D'_{c,t} \sim \mathcal{N}(60, 12^2)$. The deviation of testing environment from training environment is larger than setting (2).

Figure 4.11 shows the comparison of inventory levels and accumulated cost under a random demand scenario in the normal setting. It can be seen that it takes less than 100 time periods for the system to reach a steady state. In this case, the first 100 periods are taken as the warm-up periods, which are not considered in the calculation of accumulated cost. Under this scenario, the RL policies achieve lower total cost than OUT policies do.

For each environment setting, 200 random demand scenarios are generated and simulated. The costs are collected and presented using box plots. Figure 4.12 shows the comparison of different policies under 3 environment settings. The results of total cost, inventory holding cost, and shortage cost are shown in Figures 4.12a, 4.12b, and 4.12c, respectively. The boxes corresponding to the environment setting (1), (2), and (3) are colored in green, yellow, and red, respectively. It is observed from Figure 4.12a that the median, minimum, and maximum

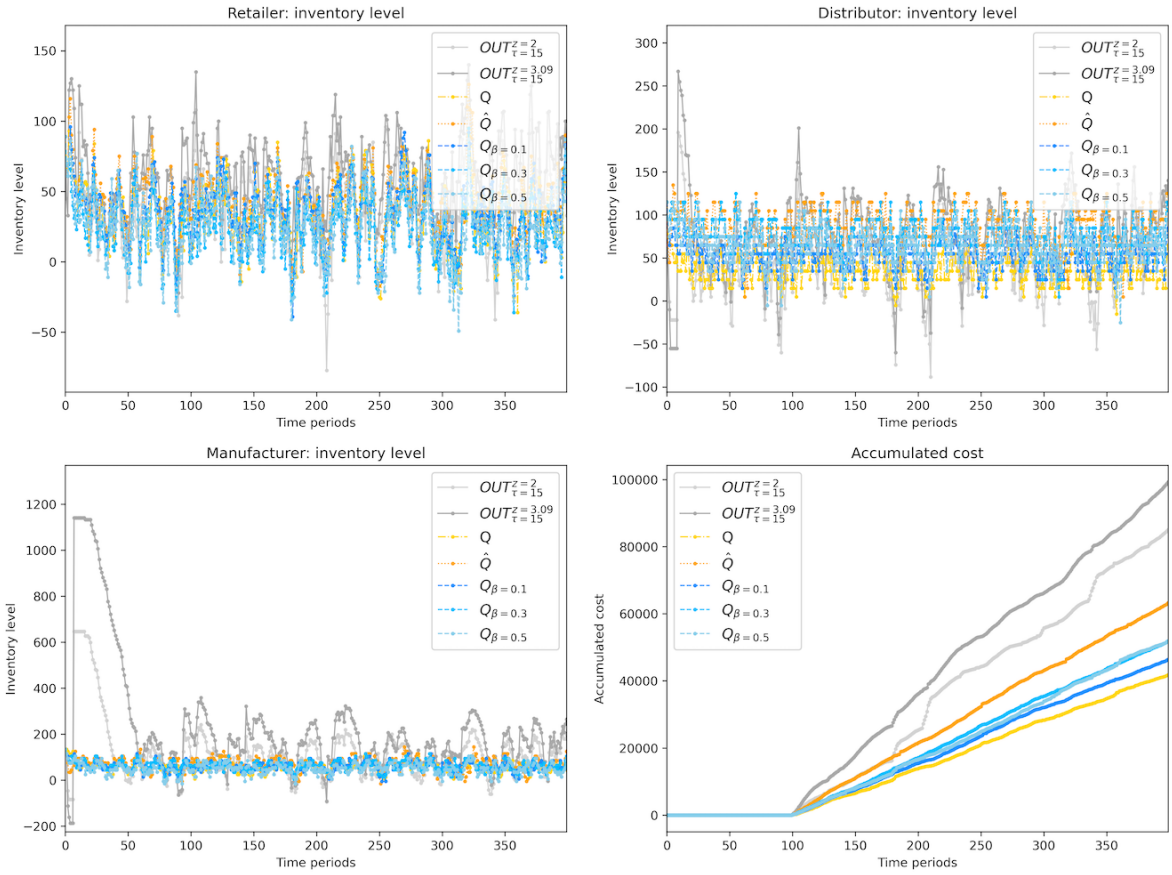
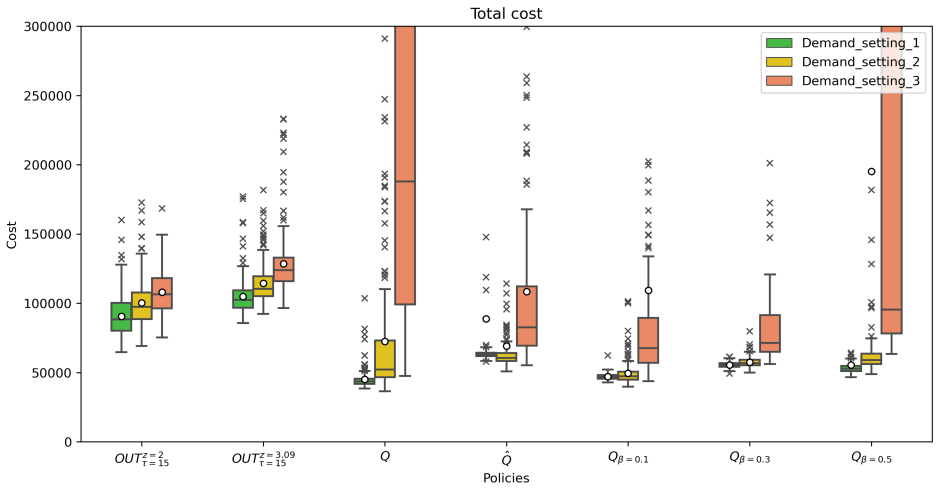


Figure 4.11. Case study II: comparison of inventory level and total cost.

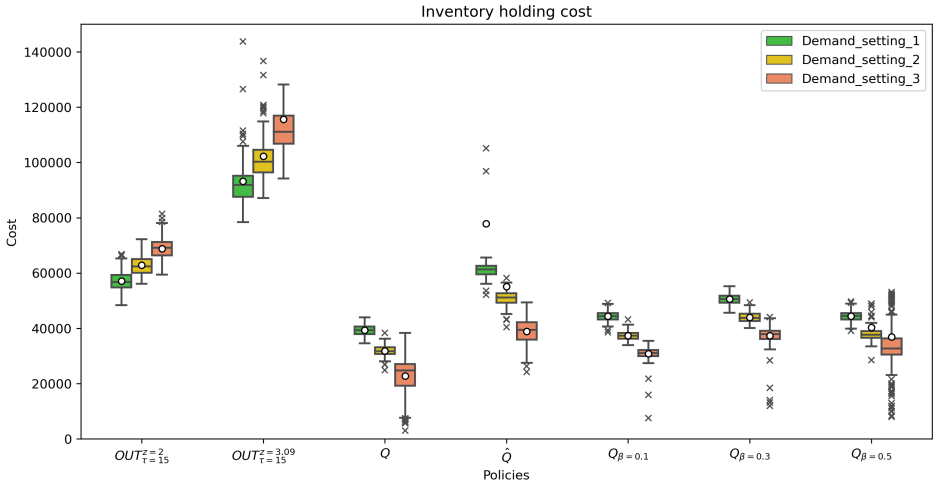
of total cost using OUT policies increase as the demand average increases. The variance using OUT policies is not significantly affected by the demand pattern, while that using $OUT_{\tau=15}^{z=3.09}$ is slightly smaller than that using $OUT_{\tau=15}^{z=2}$. Under the normal setting, the performance of safe RL policies outperforms that of the two baselines in terms of the median, minimum, maximum, and variance of total cost. Among all the policies, Q-learning has the best performance under the normal setting in terms of total cost. However, it is the most sensitive to changes in the uncertain demand, because compared with the normal setting (1), the cost statistics increase substantially in settings (2) and (3). Therefore, it is not as robust to changes in the uncertainty as $\hat{Q}$-learning, $Q_{\beta=0.1}$-learning, and $Q_{\beta=0.3}$-learning polices are. It is worth mentioning that the performance of $Q_{\beta=0.5}$-learning under setting (3) is significantly affected, and the variance of the total cost becomes large, as in Q-learning. This indicates that $\beta = 0.5$ may not be a good choice for a robust policy in this case.

Figure 4.12b shows that the median, minimum, and maximum of inventory holding cost using OUT policies increase as the demand average increases. In contrast, those statistics decrease for all the RL policies. This pattern is similar to the first case study. The reason is that the OUT policies take into account the demand increase by demand forecasting, while the RL agent does not have demand information. Among all the policies, Q-learning achieves the lowest values of the statistics under a normal setting. It is observed from Figure 4.12c that under a normal setting, the $OUT_{\tau=15}^{z=3.09}$ policy that uses a larger safety factor achieves a lower shortage cost than the $OUT_{\tau=15}^{z=2}$ policy, and all the five RL algorithms outperform the two OUT policies, in terms of the median, minimum, and maximum of shortage cost. The variance using OUT policies is not affected by the demand setting, while that using $OUT_{\tau=15}^{z=3.09}$ is smaller than that using $OUT_{\tau=15}^{z=2}$. When the demand deviates from its normal pattern, the Q-learning policy is found to be the most sensitive one, because the total cost statistics increase by a large amount in settings (2) and (3). By comparison, $\hat{Q}$-learning, $Q_{\beta=0.1}$-learning, and $Q_{\beta=0.3}$-learning policies are less affected and are more robust to changes in the environment. Similar to the total cost, the $Q_{\beta=0.5}$-learning policy leads to a large variance of shortage cost in setting (3), and it shows no advantage over Q-learning in this case.

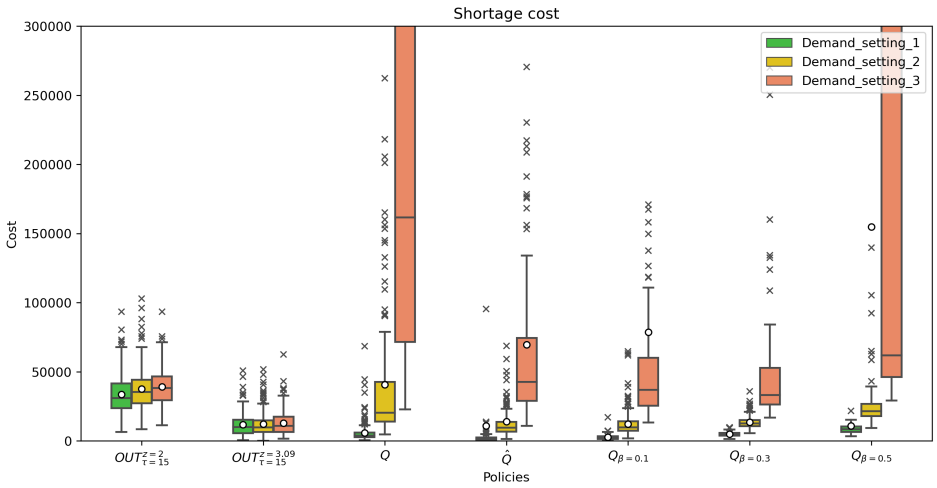
The bullwhip effect is investigated using the bullwhip slope metric. The comparison of the



(a) Total cost.



(b) Inventory holding cost.



(c) Shortage cost.

Figure 4.12. Case study II: comparison of costs under different environment settings.

bullwhip slope under different demand scenarios in the normal settings are presented in Figure 4.13. It can be seen that all the five RL policies achieve a lower BwSl than the baselines, in terms of the median, minimum, maximum, and variance. By comparing Figures 4.9 and 4.13, it is found that in the CLSC case, RL policies have more advantage than the baselines in terms of the bullwhip slope. By comparing Figures 4.7a and 4.12a, it is observed that in the CLSC case, RL policies have more advantage than the baselines in terms of the total cost. However, the first case study uses a 2-echelon forward supply chain, while the second one uses a 3-echelon CLSC, thus it is difficult to identify the cause of the difference.

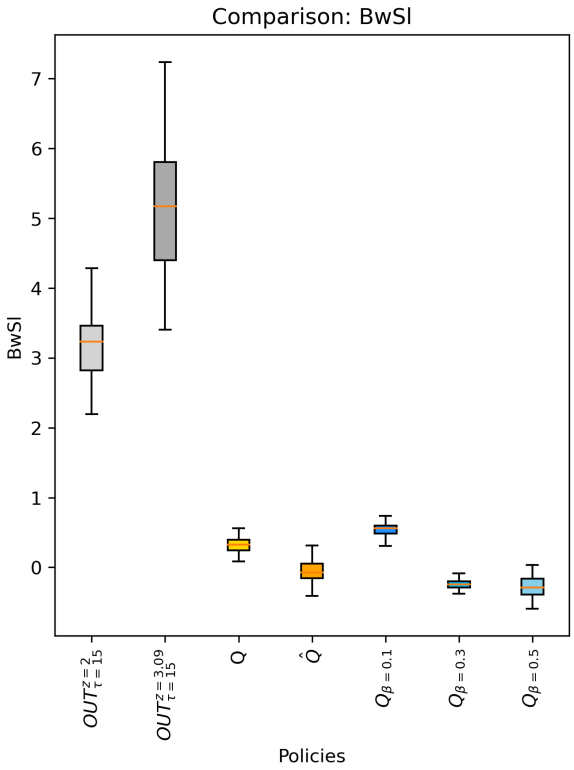


Figure 4.13. Case study II: comparison of bullwhip slope in the normal setting (1).

In order to determine whether the reason is the difference in number of echelons or the existence of reverse flow, the algorithms are tested on a 3-echelon forward supply chain for comparison. The simulation and training are implemented by setting the return rate to 0. The discretization of the state space is adjusted since uncertainty in the environment has changed. The states and corresponding codes are given in Table 4.4. For each supply chain agent, the actions available for each state are within $\{0, 15, 30, 45, 60, 75, 90\}$.

Table 4.4. Case study II: forward SC, states and codes.

$IP_{r,t}$	$IP_{d,t}$	$IP_{m,t}$	Code
$[-\infty, 0)$	$[-\infty, 0)$	$[-\infty, 0)$	-1
$[0, 30)$	$[0, 30)$	$[0, 40)$	0
$[30, 60)$	$[30, 60)$	$[40, 80)$	1
$[60, 90)$	$[60, 90)$	$[80, 120)$	2
$[90, 120)$	$[90, 120)$	$[120, 160)$	3
$[120, 150)$	$[120, 150)$	$[160, 200)$	4
$[150, 180)$	$[150, 180)$	$[200, 240)$	5
$[180, 210)$	$[180, 210)$	$[240, 280)$	6
$[210, 240)$	$[210, 240)$	$[280, 320)$	7
$[240, \infty)$	$[240, \infty)$	$[320, \infty)$	8

The total cost under 3 demand settings and 200 scenarios is plotted using a box plot, as shown in Figure 4.14. It can be seen that Q-learning has the best performance under the normal setting, while $\hat{Q}$-learning outperforms the other policies in terms of the median, average, and IQR when there is a change in the demand pattern. However, the advantage of RL in reducing total cost is not as obvious as in the CLSC case. For β-pessimistic Q-learning, $\beta = 0.1$ does not give satisfactory results for all 3 settings. $\beta = 0.3$ and 0.5 show good performance in normal settings (1), and exhibit robustness in setting (2), but are not robust in setting (3) where the deviation in the environment is larger.

In summary, based on the results of the two case studies, we have the following observations: (1) When the environment is normal as expected, Q-learning has the best performance among the policies under investigation; (2) Under abnormal setting where there is a change in the environment, $\hat{Q}$-learning and β-pessimistic Q-learning with appropriate β perform better, and they are more robust to changes in the environment; (3) Compared with OUT policy, RL policies have a greater impact on reducing the total cost in the CLSC case than in the serial supply chain case; (4) Compared with OUT policy, RL can reduce the bullwhip effect in CLSCs.

The reason for the different performance of Q-learning and $\hat{Q}$-learning is given as follows. By using the expectation-based criterion, Q-learning assumes the nature always selects the successor state with some time-invariant probability, and the agent plans for the average

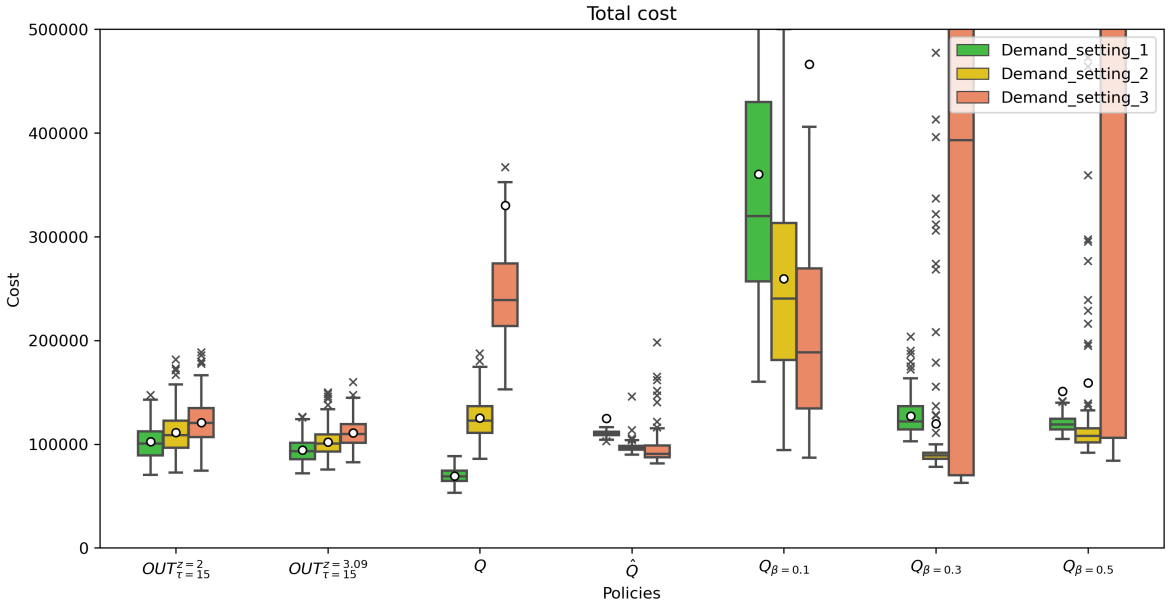


Figure 4.14. Case study II: comparison of total cost in a 3-echelon forward supply chain.

(Koenig and Simmons, 1996). Therefore, when the environment is normal, it naturally falls within the framework of Q-learning, and it has the best performance among the RL algorithms studies in this work. However, an unexpected change in the environment breaks the time-invariant assumption, and makes the state-action pairs not optimal anymore. Thus, Q-learning is sensitive to changes in the environment, and its performance could get worse in this situation. In contrast, by using the worst-case criterion, $\hat{Q}$-learning assumes the nature always selects the worst successor state for the agent (Koenig and Simmons, 1996). Thus it is risk-averse. An unexpected change in the environment can be seen as an extreme event that could occur with a small probability, thus it falls within the framework of $\hat{Q}$-learning. Therefore, $\hat{Q}$-learning is more robust to changes in the environment.

The performance of β-pessimistic Q-learning depends on the weight parameter β. In the case studies in this work, a small β such as 0.1 or 0.3 is generally a good option for a robust supply chain control policy. However, the robustness of the policy cannot necessarily be maintained when a larger value of β such as 0.5 is used, instead, its performance could get worse. This counter-intuitive behaviour of β-pessimistic Q-learning policy is also observed in the Cliff Walking task investigated by Gaskett (2003). The results in Gaskett (2003) show that when $\beta = 0.5$ is used, the agent does not choose a safer path than $\beta = 0.1$ or 0.2,

but jumps off the cliff at the beginning. Therefore, β needs to be chosen appropriately in order to obtain a robust β-pessimistic Q-learning policy. Moreover, it is worth mentioning that in the work of Klima et al. (2019), the actual probability of the agent being attacked and failing to choose the optimal action is assumed to range from 0.001 to 0.2. This models the inconsistency between the training and testing environments. The weight parameter $\varkappa$ (equivalent to β in this work) in the κ methods is set within the same range. Their results show that a small weight parameter can yield a superior policy.

4.6 Conclusion

The contribution of this work is twofold. First, the control of CLSC using RL is presented; second, risk-averse supply chain control using safe RL is explored. Two safe RL algorithms, $\hat{Q}$-learning and β-pessimistic Q-learning, are examined and compared with classical OUT policy and Q-learning. A Python-based supply chain simulator is developed using the ABM technique to examine the performance of RL algorithms in supply chain control. Two case studies are presented, which involve a two-echelon serial supply chain and a CLSC, respectively. Results show that in normal environment, Q-learning has the best performance in terms of the total cost. However, when there is an unexpected change in the environment, $\hat{Q}$-learning and β-pessimistic Q-learning with appropriate β perform better, and thus they are more robust to uncertainty. RL policies can reduce the bullwhip effect in CLSC. A potential reason is that Q-learning uses the expectation-based criterion, and thus is better when the environment is as expected and its pattern does not change over time. By comparison, $\hat{Q}$-learning uses the worst-case criterion and assumes the nature always selects the worst successor state for the agent. Therefore, $\hat{Q}$-learning optimizes the worst-case return and is suitable for risk-averse supply chain control. β-pessimistic Q-learning, which compromises between Q-learning and $\hat{Q}$-learning, can provide a robust policy when the value of β is selected appropriately.

Future research directions include (1) investigation of the impact of β on the performance of β-pessimistic Q-learning in supply chain control; (2) investigation of the robustness of

safe RL algorithms to different sources of supply chain uncertainty, such as transportation lead time, production yield, and return rate; and (3) exploration of the safe RL with function approximation, applicable to supply chain control with continuous state and action spaces.

References

Aghaie, A. and M. H. Heidary (2019). "Simulation-based optimization of a stochastic supply chain considering supplier disruption: Agent-based modeling and reinforcement learning". In: *Scientia Iranica Transactions E: Industrial Engineering* 26.6, pp. 3780–3795.

Axsäter, S. (2015). *Inventory Control*. 3rd. Springer, Cham.

Beamon, B. M. and V. C. P. Chen (2001). "Performance analysis of conjoined supply chains". In: *International Journal of Production Research* 39.14, pp. 3195–3218.

Boute, R. N. and M. R. Lambrecht (2009). "Exploring the bullwhip effect by means of spreadsheet simulation". In: *INFORMS Transactions on Education* 10.1, pp. 1–9.

Cannella, S., A. P. Barbosa-Póvoa, J. M. Framinan, and S. Relvas (2013). "Metrics for bullwhip effect analysis". In: *Journal of the Operational Research Society* 64, pp. 1–16.

Cannella, S., M. Bruccoleri, and J. M. Framinan (2016). "Closed-loop supply chains: What reverse logistics factors influence performance?" In: *International Journal of Production Economics* 175, pp. 35–49.

Chaharsooghi, S. K., J. Heydari, and S. H. Zegordi (2008). "A reinforcement learning model for supply chain ordering management: An application to the beer game". In: *Decision Support Systems* 45.4, pp. 949–959.

Chatfield, D. C. (2013). "Underestimating the bullwhip effect: a simulation study of the decomposability assumption". In: *International Journal of Production Research* 51.1, pp. 230–244.

Chatfield, D. C., J. G. Kim, T. P. Harrison, and J. C. Hayya (2004). "The bullwhip effect–impact of stochastic lead time, information quality, and information sharing: A simulation study". In: *Production and Operations Management* 13.4, pp. 340–353.

Chen, F. and R. Samroengraja (2000). "The stationary beer game". In: *Production and Operations Management* 9.1, pp. 19–30.

Coppini, M., C. Rossignoli, T. Rossi, and F. Strozzi (2010). "Bullwhip effect and inventory oscillations analysis using the beer game model". In: *International Journal of Production Research* 48.13, pp. 3943–3956.

Costantino, F., G. Di Gravio, A. Shaban, and M. Tronci (2015). "SPC forecasting system to mitigate the bullwhip effect and inventory variance in supply chains". In: *Expert Systems with Applications* 42.3, pp. 1773–1787.

Cui, Y. Y., Z. Guan, U. Saif, L. Zhang, F. Zhang, and J. Mirza (2017). "Close loop supply chain network problem with uncertainty in demand and returned products: Genetic artificial bee colony algorithm approach". In: *Journal of Cleaner Production* 162, pp. 717–742.

Dejonckheere, J., S. Disney, M. Lambrecht, and D. Towill (2004). "The impact of information enrichment on the Bullwhip effect in supply chains: A control engineering perspective". In: *European Journal of Operational Research* 153.3, pp. 727–750.

Dominguez, R., S. Cannella, A. P. Barbosa-Póvoa, and J. M. Framinan (2018). "Information sharing in supply chains with heterogeneous retailers". In: *Omega* 79, pp. 116–132.

Dominguez, R., S. Cannella, and J. M. Framinan (2015). "On returns and network configuration in supply chain dynamics". In: *Transportation Research Part E: Logistics and Transportation Review* 73, pp. 152–167.

Dominguez, R., S. Cannella, B. Ponte, and J. M. Framinan (2020). "On the dynamics of closed-loop supply chains under remanufacturing lead time variability". In: *Omega* 97, p. 102106.

Dominguez, R. and J. M. Framinan (2013). "A decision management tool: Modelling the order fulfilment process by multi-agent systems". In: *International Journal of Management and Decision Making* 12.3, pp. 240–258.

Dominguez, R., B. Ponte, S. Cannella, and J. M. Framinan (2019). "On the dynamics of closed-loop supply chains with capacity constraints". In: *Computers & Industrial Engineering* 128, pp. 91–103.

Frigge, M., D. C. Hoaglin, and B. Iglewicz (1989). "Some implementations of the boxplot". In: *The American Statistician* 43.1, pp. 50–54.

Fuji, T., K. Ito, K. Matsumoto, and K. Yano (2018). "Deep multi-agent reinforcement learning using DNN-weight evolution to optimize supply chain performance". In: *Proceedings of the 51st Hawaii International Conference on System Sciences*.

García, J. and F. Fernández (2015). "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16, pp. 1437–1480.

Gaskett, C. (2003). "Reinforcement learning under circumstances beyond its control". In: *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*.

Gebreslassie, B. H., Y. Yao, and F. You (2012). "Design under uncertainty of hydrocarbon biorefinery supply chains: Multiobjective stochastic programming models, decomposition algorithm, and a Comparison between CVaR and downside risk". In: *AIChE Journal* 58.7, pp. 2155–2179.

Giannoccaro, I. and P. Pontrandolfo (2002). "Inventory management in supply chains: A reinforcement learning approach". In: *International Journal of Production Economics* 78.2, pp. 153–161.

Giard, V. and M. Sali (2013). "The bullwhip effect in supply chains: a study of contingent and incomplete literature". In: *International Journal of Production Research* 51.13, pp. 3880–3893.

Gijsbrechts, J., R. N. Boute, J. A. Van Mieghem, and D. J. Zhang (2022). "Can deep reinforcement learning improve inventory management? Performance on lost sales, dual-sourcing, and multi-echelon problems". In: *Manufacturing & Service Operations Management* 24.3, pp. 1349–1368.

Govindan, K. and M. Fattahi (2017). "Investigating risk and robustness measures for supply chain network design under demand uncertainty: A case study of glass supply chain". In: *International Journal of Production Economics* 183, pp. 680–699.

Govindan, K., H. Soleimani, and D. Kannan (2015). "Reverse logistics and closed-loop supply chain: A comprehensive review to explore the future". In: *European Journal of Operational Research* 240.3, pp. 603–626.

Guide, V. D. R. and L. N. Van Wassenhove (2009). "OR FORUM–The evolution of closed-loop supply chain research". In: *Operations Research* 57.1, pp. 10–18.

Gupta, A. and C. D. Maranas (2003). "Managing demand uncertainty in supply chain planning". In: *Computers & Chemical Engineering* 27.8, pp. 1219–1227.

Heckmann, I., T. Comes, and S. Nickel (2015). "A critical review on supply chain risk – Definition, measure and modeling". In: *Omega* 52, pp. 119–132.

Heger, M. (1994). "Consideration of risk in reinforcement learning". In: *Machine Learning Proceedings 1994*. Ed. by W. W. Cohen and H. Hirsh. San Francisco (CA): Morgan Kaufmann, pp. 105–111.

Hubbs, C. D., C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick (2020). "A deep reinforcement learning approach for chemical production scheduling". In: *Computers & Chemical Engineering* 141, p. 106982.

Ilgin, M. A. and S. M. Gupta (2010). "Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art". In: *Journal of Environmental Management* 91.3, pp. 563–591.

Jiang, C. and Z. Sheng (2009). "Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system". In: *Expert Systems with Applications* 36.3, Part 2, pp. 6520–6526.

Jiang, H., H. Zhang, Y. Cui, and G. Xiao (2018). "Robust control scheme for a class of uncertain nonlinear systems with completely unknown dynamics using data-driven reinforcement learning method". In: *Neurocomputing* 273, pp. 68–77.

Jinqi, P., P. Taiyang, and R. Lei (2017). "The supply chain network on cloud manufacturing environment based on COIN model with Q-learning algorithm". In: *2017 5th International Conference on Enterprise Systems (ES)*, pp. 52–57.

Kara, A. and I. Dogan (2018). "Reinforcement learning approaches for specifying ordering policies of perishable inventory systems". In: *Expert Systems with Applications* 91, pp. 150–158.

Kemmer, L., H. von Kleist, D. de Rochebouet, N. Tziortziotis, and J. Read (2018). "Reinforcement learning for supply chain optimization". In: *European Workshop on Reinforcement Learning 14*.

Kim, C. O., I.-H. Kwon, and J.-G. Baek (2008). "Asynchronous action-reward learning for nonstationary serial supply chain inventory control". In: *Applied Intelligence* 28, pp. 1–16.

Kim, C. O., I.-H. Kwon, and C. Kwak (2010). "Multi-agent based distributed inventory control model". In: *Expert Systems with Applications* 37.7, pp. 5186–5191.

Klima, R., D. Bloembergen, M. Kaisers, and K. Tuyls (2019). "Robust temporal difference learning for critical domains". In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Montreal, Canada.

Koenig, S. and R. G. Simmons (1996). "The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms". In: *Machine Learning* 22, pp. 227–250.

Kwak, C., J. S. Choi, C. O. Kim, and I.-H. Kwon (2009). "Situation reactive approach to Vendor Managed Inventory problem". In: *Expert Systems with Applications* 36.5, pp. 9039–9045.

Kwon, I.-H., C. O. Kim, J. Jun, and J. H. Lee (2008). "Case-based myopic reinforcement learning for satisfying target service level in supply chain". In: *Expert Systems with Applications* 35.1, pp. 389–397.

Lee, J. H., J. Shin, and M. J. Realff (2018). "Machine learning: Overview of the recent progresses and implications for the process systems engineering field". In: *Computers & Chemical Engineering* 114, pp. 111–121.

Li, J., Z. Sheng, and H. Liu (2010). "Multi-agent simulation for the dominant players' behavior in supply chains". In: *Simulation Modelling Practice and Theory* 18.6, pp. 850–859.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis (2015). "Human-level control through deep reinforcement learning". In: *Nature (London)* 518.7540, pp. 529–533.

Mortazavi, A., A. Arshadi Khamseh, and P. Azimi (2015). "Designing of an intelligent self-adaptive model for supply chain ordering management system". In: *Engineering Applications of Artificial Intelligence* 37, pp. 207–220.

Ning, C. and F. You (2019). "Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming". In: *Computers & Chemical Engineering* 125, pp. 434–448.

Oroojlooyjadid, A., M. Nazari, L. V. Snyder, and M. Takáč (2022). "A deep Q-network for the beer game: Deep reinforcement learning for inventory optimization". In: *Manufacturing & Service Operations Management* 24.1, pp. 285–304.

Othman, S. N. and N. H. Mustaffa (2012). "Supply chain simulation and optimization methods: An overview". In: *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, pp. 161–167.

Peng, Z., Y. Zhang, Y. Feng, T. Zhang, Z. Wu, and H. Su (2019). "Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty". In: *2019 Chinese Automation Congress (CAC)*, pp. 3512–3517.

Pishvaee, M. S., M. Rabbani, and S. A. Torabi (2011). "A robust optimization approach to closed-loop supply chain network design under uncertainty". In: *Applied Mathematical Modelling* 35.2, pp. 637–649.

Ponte, B., E. Sierra, D. de la Fuente, and J. Lozano (2017). "Exploring the interaction of inventory policies across the supply chain: An agent-based approach". In: *Computers & Operations Research* 78, pp. 335–348.

Pontrandolfo, P., A. Gosavi, O. G. Okogbaa, and T. K. Das (2002). "Global supply chain management: A reinforcement learning approach". In: *International Journal of Production Research* 40.6, pp. 1299–1317.

Qiu, M., H. Ding, J. Dong, C. Ren, and W. Wang (2007). "Impact of business service modes on distribution systems: A reinforcement learning approach". In: *IEEE International Conference on Services Computing (SCC 2007)*, pp. 294–299.

Rajendran, S. and A. Ravi Ravindran (2019). "Inventory management of platelets along blood supply chain to minimize wastage and shortage". In: *Computers & Industrial Engineering* 130, pp. 714–730.

Savaskan, R. C., S. Bhattacharya, and L. N. Van Wassenhove (2004). "Closed-loop supply chain models with product remanufacturing". In: *Management Science* 50.2, pp. 239–252.

Shin, J., T. A. Badgwell, K.-H. Liu, and J. H. Lee (2019). "Reinforcement Learning – Overview of recent progress and implications for process control". In: *Computers & Chemical Engineering* 127, pp. 282–294.

Spielberg, S., A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. B. Gopaluni (2020). "Deep reinforcement learning for process control: A primer for beginners". In: arXiv:2004.05490v1.

Sterman, J. D. (1989). "Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment". In: *Management Science* 35.3, pp. 321–339.

Sui, Z., A. Gosavi, and L. Lin (2010). "A reinforcement learning approach for inventory replenishment in Vendor-Managed Inventory systems with consignment inventory". In: *Engineering Management Journal* 22.4, pp. 44–53.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. 2nd. MIT Press.

Tang, C. S. (2006). "Perspectives in supply chain risk management". In: *International Journal of Production Economics* 103.2, pp. 451–488.

Tombido, L., L. Louw, and J. van Eeden (2020). "The bullwhip effect in closed-loop supply chains: A comparison of series and divergent networks". In: *Journal of Remanufacturing* 10.3, pp. 207–238.

Valluri, A., M. J. North, and C. M. Macal (2009). "Reinforcement learning in supply chains". In: *International Journal of Neural Systems* 19.05, pp. 331–344.

van Tongeren, T., U. Kaymak, D. Naso, and E. van Asperen (2007). "Q-learning in a competitive supply chain". In: *2007 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1211–1216.

Wang, J., C. L. E. Swartz, B. Corbett, and K. Huang (2020). "Supply chain monitoring using principal component analysis". In: *Industrial & Engineering Chemistry Research* 59.27, pp. 12487–12503.

Watkins, C. J. and P. Dayan (1992). "Q-learning". In: *Machine Learning* 8, pp. 279–292.

You, F., J. M. Wassick, and I. E. Grossmann (2009). "Risk management for a global supply chain planning under uncertainty: Models and algorithms". In: *AIChE Journal* 55.4, pp. 931–946.

Zhang, Y. and S. Bhattacharyya (2007). "Effectiveness of Q-learning as a tool for calibrating agent-based supply network models". In: *Enterprise Information Systems* 1.2, pp. 217–233.

Zorowitz, S., I. Momennejad, and N. D. Daw (2020). "Anxiety, avoidance, and sequential evaluation". In: *Computational Psychiatry* 4.

Chapter 5

Deep Learning-Based Model Predictive Control for Real-Time Supply Chain Optimization

For the study in this chapter, the following paper will be submitted:

[1] Jing Wang, Christopher L.E. Swartz, and Kai Huang. "Deep learning-based model predictive control for real-time supply chain optimization". *Journal of Process Control*. To be submitted.

5.1 Introduction

In this chapter, we continue exploring the operational supply chain control problem, where the focus is developing a control method to improve the computational efficiency in real time. In the era of Supply Chain 4.0, speed and real-time decision-making are considered as important features of modern supply chains. However, for a large-scale supply chain system, the constrained optimization models are often large and complex. Solving them in a reasonable time is challenging in many applications, and optimal solutions may be computationally intractable (Schildbach and Morari, 2016; Abbasi et al., 2020). It is therefore the motivation of this chapter.

For a supply chain optimization problem, a computation burden exists when integer decision variables and nonlinearity are involved, and when uncertainty is considered. For example, Schulz et al. (2005) formulate the supply chain model for a petrochemical complex as a multiperiod mixed-integer nonlinear program (MINLP). The model includes decisions on production, delivery, inventory management, and operating conditions for each plant in the complex. The problem involves thousands of variables and constraints, and takes more than 1 hour to solve. Moreover, uncertainty is usually associated with the parameters of the optimization problems. When uncertainty is taken into account, the problem formulation becomes more complex. A widely-used method for operational supply chain optimization under uncertainty is stochastic programming (Papageorgiou, 2009; Mastragostino et al., 2014; Grossmann et al., 2016). In two-stage stochastic programming, uncertain parameters are represented by a set of scenarios, and each scenario corresponds to a possible realization of the uncertainty according to a discretized probability distribution. The expectation of the objective function over the set of scenarios is optimized subject to common first-stage decisions. In order to characterize the uncertainty to a satisfactory extent, the number of scenarios could be huge. Stochastic programs are often difficult to solve due to their large size and complexity that grows with the number of scenarios. Although decomposition algorithms have been developed to solve stochastic linear programming (LP) or mixed-integer linear programming (MILP) problems, finding the optimal solution may still take hours (Torres et al.,

2022). Mastragostino et al. (2014) proposed a robust model predictive control (MPC) strategy for supply chain systems that captures uncertainty by stochastic programming. Compared with the nominal MPC, the robust MPC formulation achieves significant improvement of the performance, while it also increases the solution time substantially.

The above examples indicate that real-time supply chain optimization is intractable in some cases. In order to meet the real-time property required by Supply Chain 4.0, a real-time control technique is needed. To this end, this chapter investigates a deep learning-based explicit MPC technique for supply chain operations. MPC is a multivariable control method that utilizes an explicit system model to predict future system states and determine optimal control inputs. This technique has been widely applied to industrial processes (Qin, 2003). Due to advantages such as the ability to handle constraints explicitly, the ability to address interactions and time delays in the system, and flexible problem formulation, MPC has been applied to supply chains more recently (Perea-López et al., 2003; Mestan et al., 2006; Li and Marlin, 2009; Subramanian et al., 2012; Subramanian et al., 2013; Mastragostino et al., 2014; Schildbach and Morari, 2016). For a MPC problem in the form of a linear or quadratic program (QP), it can be formulated as a multi-parametric LP or QP by considering the current system state as parameters, and the optimal solution is a piece-wise affine (PWA) function and only depends on the current state (Bemporad et al., 2002; Faísca et al., 2007). The MPC problem needs to be solved for updated system states at every time step. In order to avoid repetitive online computation and reduce solution time, this PWA function can be precomputed offline and stored for online use. This is known as the *explicit MPC* technique. However, as the planning horizon and the number of constraints increase, the required storage increases exponentially, which makes it difficult to obtain and apply the explicit MPC law. For this reason, an artificial neural network (NN) can be used to approximate the explicit MPC laws, i.e., to construct a map from system state to optimal control inputs. This is known as the *deep learning-based explicit MPC* technique (Karg and Lucia, 2020). If the MPC problem is not a LP or QP, e.g., for a nonlinear or mixed-integer MPC problem, the deep learning-based method can still be applied, as it has been established by Hornik et al. (1989) that multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any measurable function to any desired degree of accuracy,

provided a sufficient number of hidden units are available. Thus they are a class of universal approximators.

NNs with single hidden layer are referred to as *shallow neural networks* (Lopez-Martin et al., 2019), and NNs with two or more hidden layer are called *deep neural networks* (DNNs) (Abiodun et al., 2018; Saikia et al., 2020; Karg and Lucia, 2020). It is worth mentioning that in the current literature, the research on NN-based system control generally has two paradigms, which are:

(1) Using a NN to model the dynamic system or process, as investigated by, e.g., Punjani and Abbeel (2015), Deepa and Baranilingesan (2018), Wu et al. (2020), Bieker et al. (2020), Hassanpour et al. (2022), and Bonassi et al. (2022). This paradigm is referred to as the system identification (Punjani and Abbeel, 2015; Hassanpour et al., 2022) or the *surrogate model* (Bieker et al., 2020).

(2) Using a NN to approximate the optimal control law, as investigated by, e.g., Karg and Lucia (2020) and Kumar et al. (2021). This paradigm is referred to as the *surrogate optimizer* (Krishnamoorthy and Skogestad, 2019).

In some works, the two paradigms are investigated in an integrated manner (Wu et al., 1992; Karg and Lucia, 2021). In this chapter, we focus on the second paradigm. A review of relevant literature is presented as follows. Research on using (feedforward) NNs as approximate controllers to reduce online computation burden has emerged since the 1990s. Early research used shallow NNs and focused on nonlinear control. Wu et al. (1992) used a shallow NN to approximate the nonlinear control of turbogenerators. The NN regulator showed satisfactory control performance and compared well with an adaptive controller under different operating conditions. Parisini and Zoppoli (1995) used a shallow NN to approximate the receding-horizon regulator of nonlinear systems such as a space robot. Simulation results demonstrated the effectiveness of the approach. Ortega and Camacho (1996) used a shallow NN to approximate the nonlinear model predictive controller for mobile robot navigation. Experiments on a mobile robot testbed validated the method. Cavagnari et al. (1999) investigated the use of a shallow NN for the nonlinear receding-horizon control

of a seesaw. Results show that the NN controller guarantees good performance. Nayeri et al. (2004) investigated the nonlinear control of spacecraft slew maneuver. Simulation results show that smooth control of pitch angle and effective vibration suppression can be achieved by the NN controller. Åkesson et al. (2005) applied a shallow NN for the model predictive control of a nonlinear pH neutralization process. The study showed that this approach achieves good control performance and reduces online computations significantly. Åkesson and Toivonen (2006) further investigated the use of the approach for the MPC of a non-isothermal continuous stirred tank reactor (CSTR). Numerical examples showed that the NN model predictive controller achieved near-optimal control performance. Csekő et al. (2015) looked into the NN approximation of a mixed-integer QP and applied it to quarter car semiactive suspension model. Results show that a well-tuned NN can replace the explicit MPC controller.

The above studies have shown that NNs with single hidden layer is already able to be a satisfactory approximate controller. In the current literature, DNNs with two or more hidden layers have also been investigated for approximating optimal control inputs. Ahmed and Al-Dajani (1998) investigated the use of a NN with two hidden layers for the control of nonlinear plants and a continuous flow stirred tank reactor. The results of simulation on nonlinear plants demonstrated the practical usefulness of the proposed schemes in regulating nonlinear plants. Furuta et al. (2017) investigated the use of a NN with two hidden layers for the dynamic manipulation of a robot. The method is validated by experiments. Karg and Lucia (2018) investigated MPC in the form of mixed-integer QP for the control of an energy management system. Results show that a DNN achieves a significantly smaller training error than a shallow NN with the same total number of neurons. Moreover, the number of weights needed to represent the DNN is smaller than to represent the shallow NN, which leads to a smaller memory footprint. Chen et al. (2018) employed a DNN to approximate the optimal MPC control law for constrained linear quadratic regulator systems. An efficient policy gradient method is proposed to train the model. Drgoňa et al. (2018) applied a deep time delay NN to the model predictive building control problem. Hardware-in-the-loop simulation results show that the method outperforms the regression tree method. Hirose et al. (2018) applied the deep learning-based MPC policy to the human following control of a

personal robot. The method is shown to be effective and significantly faster than solving the QP. Pon Kumar et al. (2018) developed a long short-term memory (LSTM) supported NN to approximate the ideal MPC. The LSTM part of the controller ensures that the current control action depends on the past control actions, the current system output and the target output. It is observe that the controller outperforms the LSTM-only and the NN-only controllers, and demonstrates better generalization over the larger process state space. Hertneck et al. (2018) proposed a supervised learning framework for the approximate model predictive controller with guarantees on stability and constraint satisfaction.

In more recent works, approximation of MPC problem in the form of an LP or QP received more attention (Karg and Lucia, 2020; Chen et al., 2022). Karg and Lucia (2020) show that DNNs with rectified linear units (ReLUs) as activation functions can exactly represent the piecewise affine explicit MPC law, and not only approximate it arbitrarily well. A comparative study shows that the deep learning-based explicit MPC achieves better performance than other approximate explicit MPC methods such as multivariate polynomials, with significantly smaller memory requirements. Kumar et al. (2021) applied the DNN design approach to a CSTR and an industrial crude distillation unit model, and compared its performance with the online QP-based MPC. Their study shows that DNNs can implement MPC significantly faster than an QP solver with the closed-loop performance loss less than 1%. Furthermore, deep learning-based approximation of nonlinear MPC (Cao and Gopaluni, 2020; Lucia et al., 2021; Adhau et al., 2021), robust nonlinear MPC (Lucia and Karg, 2018; Karg and Lucia, 2019; Karg and Lucia, 2021; Karg et al., 2021), Gaussian processes-based multistage MPC (Bonzanini et al., 2021), mixed-integer MPC (Karg and Lucia, 2018; Masti and Bemporad, 2019), economic MPC (Krishnamoorthy et al., 2021), and MPC of linear parameter-varying systems (Zhang et al., 2021) have also been investigated in the current literature.

It is worth mentioning that this technique is also called deep MPC, surrogate optimizer (Krishnamoorthy and Skogestad, 2019), neural (network) regulator, NN MPC, MPC learning (Furuta et al., 2017), MPC policy learning (Hirose et al., 2018), and approximate MPC (Pin et al., 2013; Hertneck et al., 2018; Nubert et al., 2020; Paulson and Mesbah, 2020) in some works. The series of papers Chang-Yun Seong and Widrow (2001a), Chang-Yun Seong and Widrow (2001b), and Chang-Yun Seong and Widrow (2001c) presented the connection

between the NN-based approach and dynamic programming, and applies the approach to several systems including control of autonomous vehicles and of a robot arm. The method is referred to as neural dynamic optimization, as a method of optimal feedback control for nonlinear systems. In this work, *deep learning-based MPC* (DLBMPC) is used. To the best of the author's knowledge, despite the various applications of the deep learning-based MPC technique, the use of it for supply chain optimization has not been investigated. Considering the computation burden of optimal control of large-scale supply chains, this technique has great potential in aiding real-time supply chain control. To this end, this study aims at addressing the research gap, and focuses on applying deep learning-based MPC to the operational supply chain optimization.

The deep learning-based MPC method enables decision-making in real time, which facilitates the development of Supply chain 4.0. Studies on physical or fast embedded systems have shown that the method can provide near-optimal solutions for MPC problems at an extremely low computational cost. Various studies have shown that DLBMPC is efficient in reducing online computation time for the control of physical or embedded systems. It is able to provide near-optimal solutions for nonlinear or mixed-integer MPC problems in seconds, or even in milliseconds (Karg and Lucia, 2018; Lucia and Karg, 2018; Karg and Lucia, 2019; Karg and Lucia, 2020; Karg and Lucia, 2021; Kumar et al., 2021). Therefore, this technique has great potential in real-time supply chain management.

This technique represents a shift toward removing the cap on the complexity of the online optimization problem, and thus more sophisticated formulations can be adopted for real-time decisions. Long computation times may limit the use of optimization techniques in real-time supply chain optimization. In order to obtain an optimization problem that can be solved online in a short time, the prediction horizon and the number of scenarios may have to be set to small values to avoid heavy computation load. The simplification of the problem may result in suboptimal decisions. DLBMPC mitigates the cap on the complexity of the online optimization problem by taking advantage of the offline solution process. Since the optimal solutions are obtained offline before the online use of the controller, complicated problem formulations and accurate solution methods can be adopted for better decisions. The online computation time depends on the evaluation of the neural network, which is very

fast compared to solving the original optimization problem. As mentioned in Pon Kumar et al. (2018), this technique can approximate the best-in-class/ideal MPC, without considering the real-time computation load.

Another advantage of using the DLBMPC technique is that it does not require an optimization solver for real-time supply chain decision-making. This benefits many industries that may have limited access to commercial solvers or have limited computing resource for their daily operational decision-making (Abbasi et al., 2020; Kumar et al., 2021). The DLBMPC avoids the need for long-term access to high-performance computing facilities in order to solve large-scale supply chain optimization problems. What it needs is an initial short-term use of high-performance computing resource for learning the optimal decisions. When the trained model is applied to real-time supply chain management, near-optimal decisions are provided immediately at an extremely low computation cost.

The main contributions of this chapter are:

(1) Investigation of the deep learning-based MPC in the context of supply chain optimization;

(2) An approach to implement the deep learning-based MPC method when there are delayed/lagged terms in the system;

(3) A heuristic approach to feasibility recovery for deep learning-based mixed-integer MPC, with the binary decision variables taken into account;

(4) A data generation method that utilizes both random samples and closed-loop simulation samples.

The remainder of this chapter is organized as follows. Section 5.2 presents the method of deep learning-based MPC for supply chain optimization. Section 5.3 presents two case studies. The first case study is conducted on a benchmark supply chain and is an LP. The second case study is based on a more complicated supply chain and involves a mixed-integer MPC problem. The conclusion is given in Section 5.4.

5.2 Deep Learning-Based MPC for Supply Chain Optimization

This section presents the deep learning-based MPC technique for supply chain optimization. Section 5.2.1 describes the supply chain model and the MPC problem. Section 5.2.2 describes a method to model the delayed terms in the system, based on which the deep learning-based MPC technique is developed. Section 5.2.3 presents the deep learning-based MPC technique for supply chain optimization, as well as a heuristic method to deal with the binary decision variables for deep learning-based mixed-integer MPC. Section 5.2.4 discusses data generation.

5.2.1 MPC of supply chains

Consider a general supply chain network, the structure of which is shown in Figure 5.1 (Mastragostino et al., 2014). The supply chain consists of a set of suppliers S_S, a set of plant sites S_P, and a set of distribution sites S_{DC}. It deals with the procurement of raw materials, production and distribution of products to satisfy customers' demands. At each plant site $p \in S_P$, there are a set of production schemes S^p_{PS} installed. A plant site p purchases a set of raw materials S^p_R from the suppliers, and transforms them into a set of intermediate products S^p_I and a set of final products S_F through the production schemes. The final products are shipped to the distribution sites, where customers' orders are received and satisfied.

The mass balance for the plant sites is given by Equation (5.1). Equations (5.1a), (5.1b), and (5.1c) represent the mass balance for the raw materials, intermediate products, and final products, respectively.

$$I^p_{j,t+1} = I^p_{j,t} + \sum_{s \in S_S} O^{s,p}_{j,t-\delta^{s,p}} - \sum_{ps \in O^p_j} \mu^p_{ps,j} W^p_{ps,t}, \qquad \forall p \in S_P, j \in S^p_R, t \tag{5.1a}$$

$$\begin{aligned} I^p_{j,t+1} = {} & I^p_{j,t} + \sum_{ps \in I^p_j} \mu^p_{ps,j} \beta^p_{ps} W^p_{ps,t-\delta^p_{ps}} - \sum_{ps} \mu^p_{ps,j} W^p_{ps,t} \\ & - \sum_{p' \neq p} Q^{p,p'}_{j,t} + \sum_{p' \neq p} Q^{p',p}_{j,t-\delta^{p',p}}, \qquad \forall p \in S_P, j \in S^p_I, t \end{aligned} \tag{5.1b}$$

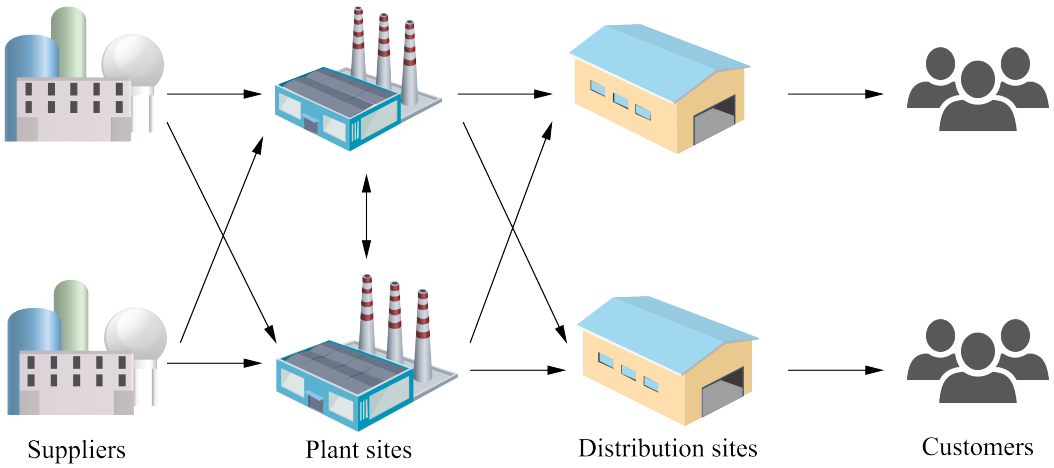


Figure 5.1. A supply chain network.

$$I^p_{j,t+1} = I^p_{j,t} + \sum_{ps \in O^p_j} \mu^p_{ps,j} \beta^p_{ps} W^p_{ps,t-\delta^p_{ps}} - \sum_{dc \in S_{DC}} Q^{p,dc}_{j,t}, \qquad \forall p \in S_P, j \in S_F, t \tag{5.1c}$$

where $I^p_{j,t}$ is the inventory of material j at plant site p at time period t; $O^{s,p}_{j,t}$ is the purchase quantity of raw material j to plant site p from supplier s at time period t; $W^p_{ps,t}$ is the quantity of main raw material associated with production scheme ps at plant site p at time period t; $Q^{p,p'}_{j,t}$, $Q^{p',p}_{j,t}$, and $Q^{p,dc}_{j,t}$ are the quantities of material j shipped from plant site p to p', from plant site p' to p, and from plant site p to distribution site dc at time period t, respectively; $\delta^{s,p}$, δ^p_{ps}, and $\delta^{p',p}$ are the transportation delay from s to p, production delay of production scheme ps, and transportation delay from p' to p, respectively. O^p_j and I^p_j are the sets of production schemes that consume and output material j, respectively. $\mu^p_{ps,j}$ is the mass balance coefficient of j in ps at p, and β^p_{ps} is the yield parameters of ps at p. At time period t, if $W^p_{ps,t}$ units of main raw material are consumed for ps at p, then $\mu^p_{ps,j} W^p_{ps,t}$ units of raw material j is consumed; for a product j' of ps, $\mu^p_{ps,j'} \beta^p_{ps} W^p_{ps,t}$ units of j' is produced after a production delay δ^p_{ps}.

The mass balance for distribution sites is given by Equation (5.2):

$$I^{dc}_{j,t+1} = I^{dc}_{j,t} + \sum_{p \in S_P} Q^{p,dc}_{j,t-\delta^{p,dc}} - SA^{dc}_{j,t}, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.2a}$$

$$B^{dc}_{j,t+1} = B^{dc}_{j,t} + D^{dc}_{j,t} - SA^{dc}_{j,t}, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.2b}$$

where $I^{dc}_{j,t}$, $B^{dc}_{j,t}$, $D^{dc}_{j,t}$ and $SA^{dc}_{j,t}$ are the inventory, back orders, demand forecast and sales quantity of final product j at plant site p at time period t, respectively.

In cases where the distribution sites use a "best I can do" policy, the sales quantity $SA^{dc}_{j,t}$ is determined in this way: when sufficient stock is available, the demand and accumulated back orders are satisfied; otherwise the available stock is shipped to satisfy partial customer orders, and the unsatisfied orders will be recorded as back orders (Mastragostino et al., 2014). In other words, $SA^{dc}_{j,t}$ is determined based on the following rule:

$$SA^{dc}_{j,t} = \begin{cases} B^{dc}_{j,t} + D^{dc}_{j,t}, & \text{if} \quad I^{dc}_{j,t} \geq B^{dc}_{j,t} + D^{dc}_{j,t} \\ I^{dc}_{j,t}, & \text{if} \quad I^{dc}_{j,t} < B^{dc}_{j,t} + D^{dc}_{j,t} \end{cases} \quad \forall j \in S_F, dc \in S_{DC}, t \tag{5.3}$$

In order to model this nonlinear policy, a variable $IL^{dc}_{j,t}$ is defined:

$$IL^{dc}_{j,t} = I^{dc}_{j,t} - B^{dc}_{j,t}, \quad \forall j \in S_F, dc \in S_{DC}, t \tag{5.4}$$

$IL^{dc}_{j,t}$ is the inventory level defined in Axsäter (2015). By introducing $IL^{dc}_{j,t}$, the mass balance Equation (5.2) is transformed to Equation (5.5b), with inequality constraints (5.5c) and (5.5d) imposed (Mastragostino et al., 2014).

$$IL^{dc}_{j,t+1} = IL^{dc}_{j,t} + \sum_{p \in S_P} Q^{p,dc}_{j,t-\delta^{p,dc}} - D^{dc}_{j,t}, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.5a}$$

$$B^{dc}_{j,t} \geq -IL^{dc}_{j,t}, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.5b}$$

$$I^{dc}_{j,t} \geq IL^{dc}_{j,t} + B^{dc}_{j,t}, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.5c}$$

$$I^{dc}_{j,t} \geq 0, \quad B^{dc}_{j,t} \geq 0, \qquad \forall j \in S_F, dc \in S_{DC}, t \tag{5.5d}$$

This transformation renders the model linear. Note that $I^{dc}_{j,t}$ and $B^{dc}_{j,t}$ are minimized in the objective function of the MPC problem. Therefore, the inequality constraints Equations (5.5b), (5.5c), and (5.5d) ensure that when $IL^{dc}_{j,t} \geq 0$, $B^{dc}_{j,t} = 0$ and $I^{dc}_{j,t} = IL^{dc}_{j,t}$; and when $IL^{dc}_{j,t} < 0$, $B^{dc}_{j,t} = -IL^{dc}_{j,t}$ and $I^{dc}_{j,t} = 0$. In both cases, Equation (5.4) is satisfied.

The other constraints to be imposed are given in Equation (5.6). Equations (5.6a) and (5.6b) are the constraints on the binary production decisions and the minimum and maximum production batch sizes. Here, $u^p_{ps,t}$ is a binary variable associated with production scheme ps at plant site p at time period t. If $u^p_{ps,t} = 1$, then production scheme ps is started at time period t, and 0 otherwise. Equation (5.6b) ensures that for each production scheme, there is no more than one batch in process. Equations (5.6c), (5.6d), and (5.6e) are constraints on the transportation quantities. Equations (5.6f), (5.6g), and (5.6h) are the constraints on the inventories of the plant sites. Equation (5.6i) is the constraint on the inventories of the distribution sites. Equation (5.6j) specifies the nonnegative variables.

$$u^p_{ps,t} W^{p,lb}_{ps} \le W^p_{ps,t} \le u^p_{ps,t} W^{p,ub}_{ps}, \qquad \forall p \in S_P, ps \in S^p_{PS}, t \tag{5.6a}$$

$$\sum_{t'=t}^{t-\delta^p_{ps}+1} u^p_{ps,t'} \le 1, \quad u^p_{ps,t} \in \{0,1\}, \qquad \forall p \in S_P, ps \in S^p_{PS}, t \tag{5.6b}$$

$$\sum_{j \in S^p_R} O^{s,p}_{j,t} \le Q^{s,p}_{ub}, \qquad \forall p \in S_P, j \in S^p_R, s \in S_S, t \tag{5.6c}$$

$$\sum_{j \in S_I} Q^{p,p'}_{j,t} \le Q^{p,p',ub}, \qquad \forall p, p', t \tag{5.6d}$$

$$\sum_{j \in S_F} Q^{p,dc}_{j,t} \le Q^{p,dc,ub}, \qquad \forall p, dc, t \tag{5.6e}$$

$$\sum_{j \in S^p_R} I^p_{j,t} \le I^{p,ub}_R, \qquad \forall p \in S_P, t \tag{5.6f}$$

$$\sum_{j \in S^p_I} I^p_{j,t} \le I^{p,ub}_I, \qquad \forall p \in S_P, t \tag{5.6g}$$

$$I^p_{j,t} \le I^{p,ub}_j, \qquad \forall p \in S_P, j \in S_F, t \tag{5.6h}$$

$$IL^{dc}_{j,t} \le I^{dc,ub}_j, \qquad \forall dc \in S_{DC}, j \in S_F, t \tag{5.6i}$$

$$W^p_{ps,t}, O^{s,p}_{j,t}, Q^{p,p'}_{j,t}, Q^{p,dc}_{j,t}, I^p_{j,t} \ge 0, \qquad \forall s, p, p', ps, dc, j, t \tag{5.6j}$$

Two objectives are considered in the operational optimization of the supply chain, the economic performance and customer service. The economic performance is measured by the total operating cost over the planning horizon, which is the summation of the purchase

cost, production cost, transportation cost, and inventory cost. The costs are calculated by Equations (5.7a)–(5.7d), respectively.

$$C_{purchase} = \sum_{t=1}^{N} \sum_{s \in S_S} \sum_{p \in S_P} \sum_{j \in S_R^p} \alpha_j^s O_{j,t}^{s,p} \tag{5.7a}$$

$$C_{production} = \sum_{t=1}^{N} \sum_{p \in S_P} \sum_{ps \in S_{PS}^p} \rho_{ps}^p W_{ps,t}^p \tag{5.7b}$$

$$C_{transport} = \sum_{t=1}^{N} \sum_{p \in S_P} \sum_{dc \in S_{DC}} \sum_{j \in S_F} \tau^{p,dc} Q_{j,t}^{p,dc} + \sum_{t=1}^{N} \sum_{p \in S_P} \sum_{p' \in S_P : p' \neq p} \sum_{j \in S_I^p} \tau^{p',p} Q_{j,t}^{p',p} \tag{5.7c}$$

$$C_{inventory} = \sum_{t=1}^{N} \sum_{p \in S_P} \sum_{j \in S_R^p \cup S_I^p \cup S_F} \pi_j^p I_{j,t}^p + \sum_{t=1}^{N} \sum_{dc \in S_{DC}} \sum_{j \in S_F} \pi_j^{dc} I_{j,t}^{dc} \tag{5.7d}$$

where α_j^s is the unit purchase cost of material j at supplier s; ρ_{ps}^p is the unit production cost of production scheme ps at plant site p; $\tau^{p,dc}$ and $\tau^{p',p}$ are the unit shipping costs from plant site p to distribution site dc and from plant site p' to plant site p, respectively; π_j^p and π_j^{dc} are the unit inventory costs of material j at plant site p and distribution site dc, respectively.

The total operating cost is calculated as the summation of the above costs:

$$J_1 = C_{purchase} + C_{production} + C_{transport} + C_{inventory} \tag{5.8}$$

The customer service is measured by the total quantity of back orders over the planning horizon:

$$J_2 = \sum_{t=1}^{N} \sum_{dc \in S_{DC}} \sum_{j \in S_F} B_{j,t}^{dc} \tag{5.9}$$

This multi-objective problem is addressed using a weighted-sum method, i.e., the objective function is calculated as a weighted sum of J_1 and J_2, which is $J = \omega_1 J_1 + \omega_2 J_2$. Here, ω_1 and ω_2 are the weights assigned to J_1 and J_2, respectively. The objective function of the

MPC problem is thus to minimize J, and the MPC problem is formulated as:

$$\begin{aligned} &\min_{u} \omega_1 J_1 + \omega_2 J_2 \\ &\text{s. t.} \quad \text{Equations } (5.1), (5.5), (5.6) \end{aligned} \tag{5.10}$$

Due to the existence of demand uncertainty, the inventory of distribution sites may exceed the maximum capacity, which may lead to infeasibility in the closed-loop implementation of MPC. In order to address this issue, Equation (5.6i) is transformed from a hard to a soft constraint, which is given by Equation (5.11). By defining a variable $OS_{j,t}^{dc}$ for overstock of material j, Equation (5.11) allows the inventory of distribution sites to exceed the upper bound. This helps address the infeasibility under uncertain demand.

$$IL_{j,t}^{dc} \leq I_j^{dc,ub} + OS_{j,t}^{dc} \qquad \forall dc \in S_{DC}, j \in S_F, t \tag{5.11}$$

Meanwhile, the overstock should be penalized in the objective function. The cost for overstock is calculated as:

$$J_3 = C_{overstock} = \sum_{t=1}^{N} \sum_{dc \in S_{DC}} \sum_{j \in S_F} OS_{j,t}^{dc} \tag{5.12}$$

The MPC problem therefore becomes:

$$\begin{aligned} &\min_{u} \omega_1 J_1 + \omega_2 J_2 + \omega_3 J_3 \\ &\text{s. t.} \quad \text{Equations } (5.1), (5.5), (5.6a - 5.6h), (5.6j), (5.11) \end{aligned} \tag{5.13}$$

Here, ω_3 is the weight assigned to the objective function J_3.

5.2.2 Modelling delayed terms

The lagged terms in Equations (5.1) and (5.5) indicate that the past decisions should be taken into account when making current decisions. For more convenient implementation of MPC, additional auxiliary state variables are introduced to carry the past decisions in the system,

and transform the supply chain model into a state-space form. This approach is referred to as *lifting* (Li and Marlin, 2009; Subramanian et al., 2012; Mastragostino et al., 2014).

Consider modelling the delayed/lagged decision $O^{s,p}_{j,t-\delta^{s,p}}$ in Equation (5.1a), which has a delay of $\delta^{s,p}$ time periods. $\delta^{s,p}$ auxiliary state variables are introduced for each time period t, and the following Equation (5.14) is constructed:

$$\begin{bmatrix} O^{s,p}_{j,t+1,1} \\ O^{s,p}_{j,t+1,2} \\ \vdots \\ O^{s,p}_{j,t+1,\delta^{s,p}-1} \\ O^{s,p}_{j,t+1,\delta^{s,p}} \end{bmatrix} = \begin{bmatrix} O^{s,p}_{j,t,2} \\ O^{s,p}_{j,t,3} \\ \vdots \\ O^{s,p}_{j,t,\delta^{s,p}} \\ O^{s,p}_{j,t} \end{bmatrix}, \quad \forall p \in S_P, j \in S^p_R, t \tag{5.14}$$

By replacing t with $t - \delta^{s,p}$ in the last row of Equation (5.14), we have $O^{s,p}_{j,t-\delta^{s,p}+1,\delta^{s,p}} = O^{s,p}_{j,t-\delta^{s,p}}$. Also, based on Equation (5.14), we have:

$$O^{s,p}_{j,t-\delta^{s,p}+1,\delta^{s,p}} = O^{s,p}_{j,t-\delta^{s,p}+2,\delta^{s,p}-1} = \cdots = O^{s,p}_{j,t,1} \tag{5.15}$$

Hence,

$$O^{s,p}_{j,t-\delta^{s,p}} = O^{s,p}_{j,t-\delta^{s,p}+1,\delta^{s,p}} = O^{s,p}_{j,t,1} \tag{5.16}$$

By replacing the lagged term $O^{s,p}_{j,t-\delta^{s,p}}$ with the auxiliary state $O^{s,p}_{j,t,1}$, and incorporating Equation (5.14), Equation (5.1a) can be transformed into the state-space form as follows:

$$\begin{aligned} I^p_{j,t+1} &= I^p_{j,t} + \sum_{s \in S_S} O^{s,p}_{j,t,1} - \sum_{ps \in O^p_j} \mu^p_{ps,j} W^p_{ps,t}, && \forall p \in S_P, j \in S^p_R, t \\ O^{s,p}_{j,t+1,k} &= O^{s,p}_{j,t,k+1}, && \forall p \in S_P, j \in S^p_R, t, k = 1, 2, \ldots, \delta^{s,p} - 1 \\ O^{s,p}_{j,t+1,\delta^{s,p}} &= O^{s,p}_{j,t}, && \forall p \in S_P, j \in S^p_R, t \end{aligned} \tag{5.17}$$

The other lagged terms can be modelled in a similar way. Then, the supply chain system can

be represented as the following state-space form:

$$x_{t+1} = Ax_t + Bu_t + v_t \tag{5.18}$$

with the state variables x_t, input variables u_t, and disturbance v_t defined as follows:

$$\begin{aligned} x_t &= \Big[\Big(I^p_{j,t}\forall p,j\Big),\Big(IL^{dc}_{j,t}\forall dc,j\Big),\Big(O^{s,p}_{j,t,k}\forall s,p,j,k\Big),\Big(W^p_{ps,t,k}\forall p,ps,k\Big),\\ &\quad \Big(Q^{p,p'}_{j,t,k}\forall p,p',j,k\Big),\Big(Q^{p,dc}_{j,t,k}\forall p,dc,j,k\Big)\Big]^T\\ u_t &= \Big[\Big(O^{s,p}_{j,t}\forall s,p,j\Big),\Big(W^p_{ps,t}\forall p,ps\Big),\Big(Q^{p,p'}_{j,t}\forall p,p',j\Big),\Big(Q^{p,dc}_{j,t}\forall p,dc,j\Big)\Big]^T\\ v_t &= \Big[\Big(D^{dc}_{j,t}\forall dc,j\Big)\Big]^T \end{aligned} \tag{5.19}$$

In Equation (5.19), the state x_t is defined to comprise of the inventory of the supply chain members and the additional state variables for modelling past decisions. The input u_t is comprises of the variables of purchase, production, and shipment quantities. The customer demand is considered as the disturbance v_t to the supply chain system. The MPC problem is therefore expressed as:

$$\begin{aligned} \min_{u=[u_1^T,u_2^T,\ldots,u_N^T]^T} \quad & \sum_{t=1}^{N} C_x^T x_t + C_u^T u_t + C_a^T a_t \\ \text{s. t.} \quad & x_{t+1} = A_1 x_t + B_1 u_t + C_1 v_t, && \forall t = 1,\ldots,N-1\\ & A_2 x_t + A_0 a_t + B_2 u_t + C_2 v_t + D_2 b_t + c \le 0, && \forall t = 1,\ldots,N\\ & x_1 = x_0 \end{aligned} \tag{5.20}$$

where x_0 represents the initial values of the state variables, and the auxiliary variables a_t and the binary decision variables b_t are defined as follows:

$$\begin{aligned} a_t &= \Big[\Big(I^{dc}_{j,t}\forall dc,j\Big),\Big(B^{dc}_{j,t}\forall dc,j\Big)\Big]^T\\ b_t &= \Big[\Big(u^p_{ps,t}\forall p,ps\Big)\Big]^T \end{aligned} \tag{5.21}$$

In the MPC problem (5.20), a_t and b_t appear in the inequality constraints, while do not appear

in the state-space equation. The optimal solution of MPC is the decision over the planning horizon N, which is denoted as $u^* = [u_1^{*T}, u_2^{*T}, \ldots, u_N^{*T}]^T$. At the immediate succeeding time step, only the decision u_1^* is implemented. The MPC problem is solved again for updated state variables, and the process is repeated.

5.2.3 Deep learning-based supply chain MPC

If the problem (5.20) is an LP or QP, the optimal solution is a PWA function of the state: $u_1^* = \mathcal{PWA}(x_1)$. Karg and Lucia (2020) show that DNNs with rectified linear units (ReLU) as activation functions can exactly represent the piecewise affine explicit MPC law, and not only approximate it arbitrarily well. When binary variables are involved and the problem is mixed-integer MPC, the deep learning-based technique also applies (Karg and Lucia, 2018; Masti and Bemporad, 2019). When the problem is nonlinear, the deep learning-based technique also applies. This is because the DNN is able to be a universal approximator.

A feedforward NN is defined as a sequence of layers of neurons which determines a function (Karg and Lucia, 2020):

$$\mathcal{NN}(x;\theta,H,L) = f_{L+1} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1(x) \tag{5.22}$$

where $x \in \mathbb{R}^{n_x}$ is the input to the NN. The output of the NN is $y \in \mathbb{R}^{n_y}$. The input layer has the same dimension as $x \in \mathbb{R}^{n_x}$. The output layer has the same dimension as $y \in \mathbb{R}^{n_y}$. L is the number of hidden layers in the NN, and H is the number of neurons in each hidden layer. $\mathcal{NN}$ is called a deep NN if $L \geq 2$. Specifically, when $L = 1$, $\mathcal{NN}(x;\theta,H,L) = f_2 \circ g_1 \circ f_1(x)$ is called a shallow NN.

$\mathcal{NN}$ can be understood as a composition function that is computed layer by layer. For the l-th hidden layer, it takes the output of the $(l-1)$-th hidden layer h_{l-1} as the input, and compute a PWA function $f_l(h_{l-1}) = W_l h_{l-1} + b_l$. Then $f_l(h_{l-1})$ goes through a nonlinear activation function g_l to obtain $g_l[f_l(h_{l-1})]$. Activation functions that have been used for NN-based MPC include ReLUs (Karg and Lucia, 2020; Kumar et al., 2021; Chen et al., 2022),

radial basis functions (RBFs) (Csekő et al., 2015), etc. In this work, ReLUs are chosen as the activation functions of DNN, considering the relationship between ReLU NN and PWA function, as well as its good performance that has been shown by Karg and Lucia (2020), Kumar et al. (2021), and Chen et al. (2022). A ReLU activation function computes the element-wise maximum between 0 and the affine function of the current layer: $g_l(f_l) = \max(0, f_l)$. The set of all the weights and biases is denote as the parameter θ.

A diagram of the proposed deep learning-based explicit MPC for supply chain operation is shown in Figure 5.2. Implementation of this technique consists of two stages: the offline stage and the online stage.

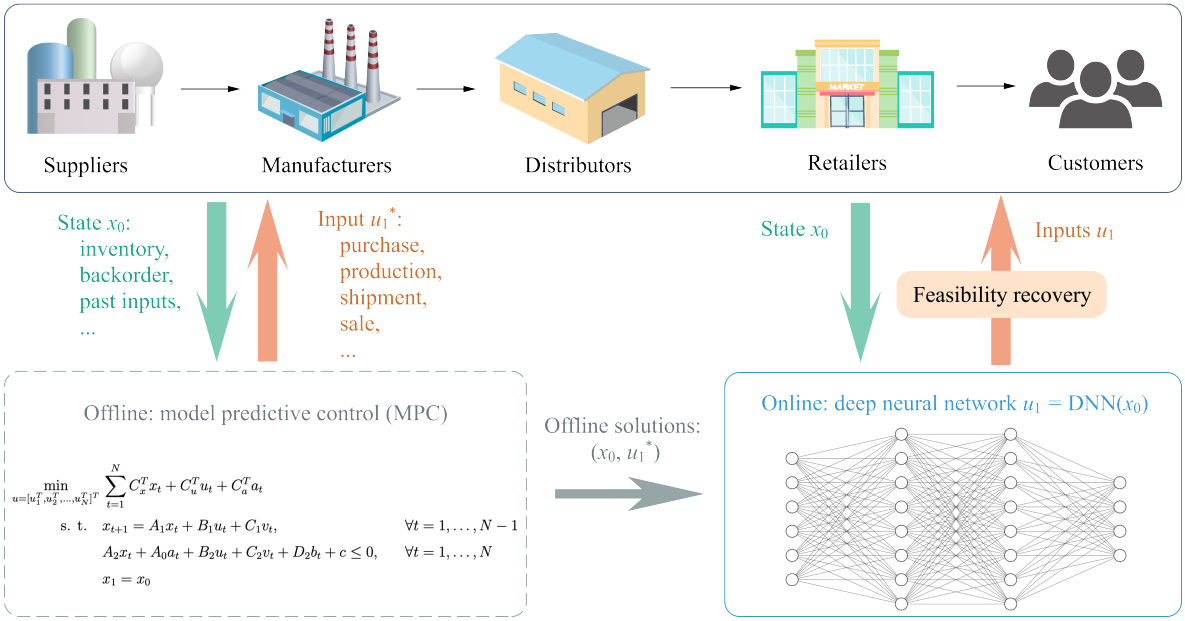


Figure 5.2. Deep learning-based explicit MPC for supply chain operation.

In the offline phase, the deep learning-controller is developed based on the optimal MPC solutions. The first step is to generate optimal state-input pairs and use them to train the DNN. This involves formulating the MPC problem for supply chain operation, i.e., the problem (5.20) in Section 5.2.2; and solving the problem offline for a set of initial states x_0 to obtain corresponding optimal inputs u_1^*. These state-inputs pairs (x_0, u_1^*) are collected and stored as the training data for the NN controller. Then, a NN $\mathcal{NN}(x; \theta, H, L) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ is built. The NN is trained using the offline data to approximate the MPC law $u_1^* = \mathcal{MPC}(x_0)$.

It is worth mentioning that the constraints in the MPC problem involve the past values of the binary decision variables b_t. In order to take into account the potential impact of these variables on the solution, the initial state x_0 is augmented with the past values of the binary variables at time step 0, which is denoted as $\bar{u}_0 = \left[\left(u^p_{ps,k} \forall p, ps, k = -1, \ldots, 1 - \delta^p_{ps}\right)\right]^T$. Here, a $u^p_{ps,k}$ with a negative k represents the past decision before the current time period. Moreover, the initial auxiliary variables a_0 are incorporated, and the state x_0 is therefore augmented into the following *augmented state* $\bar{x}_0$:

$$\begin{aligned}\bar{x}_0 = \Big[&\left(I^p_{j,0} \forall p, j\right), \left(I^{dc}_{c,0} \forall dc, c\right), \left(B^{dc}_{c,0} \forall dc, c\right), \left(O^{s,p}_{j,0,k} \forall s, p, j, k\right), \left(W^p_{ps,0,k} \forall p, ps, k\right), \\ &\left(Q^{p,p'}_{j,0,k} \forall p, p', j, k\right), \left(Q^{p,dc}_{j,0,k} \forall p, dc, j, k\right), \left(u^p_{ps,k} \forall p, ps, k = -1, \ldots, 1 - \delta^p_{ps}\right)\Big]^T\end{aligned} \tag{5.23}$$

Therefore, in this work, the DNN is approximating the map from $\bar{x}_0$ to u^*_1. With well-tuned hyperparameters and appropriate training, the NN is supposed to achieve high approximation accuracy, i.e., $u_1 = \mathcal{NN}(x_0)$ should be a good approximation of $u^*_1 = \mathcal{MPC}(x_0)$. Data generation and preprocessing are discussed in Section 5.2.4.

In the online phase, the NN is used for real-time control, thus the optimization solver is no longer needed. The current state of supply chain x_0 is fed to the NN, and the NN computes the decisions u_1. Since the output of NN is an approximation, it is possible u_1 is an infeasible solution, and thus feasibility recovery is needed. Feasibility recovery is discussed in Section 5.2.5.

5.2.4 Data generation and preprocessing

Generating data requires sampling x_1 and solving for optimal u^*_1. It is worth mentioning that it could be difficult to cover the entire state space $\mathcal{S}$ when sampling x_1, especially when there are delayed decisions in the system, and auxiliary state variables are introduced. It could be challenging to ensure that the constraints involving the auxiliary state variables to be satisfied, otherwise the x_1 could be an infeasible state. Therefore, generating optimal state-input pairs through closed-loop simulation is a better option in some cases. Closed-loop

data are used by (Karg and Lucia, 2020; Kumar et al., 2021).

The process of closed-loop sampling in this work is shown by Figure 5.3. At the beginning, an initial state x_1 is randomly sampled from the state space $\mathcal{S}$, with all the auxiliary states set to 0. This means the past decisions before time step 1 are not considered. Then, the MPC problem is solved to obtain an optimal solution u_1^*. After that, the simulator accepts the input, does a simulation with uncertainty realization, and returns the new state x_2 to the controller. In general, at time step t, MPC computes the solution u_t^* based on the current state x_t, and the state is then updated by the simulator. In this way, assuming the simulation horizon is N_h, for each simulation run, a total of N_h state-input pairs $\{(x_t, u_t^*)\}_{t=1,2,\ldots,N_h}$ are obtained. The simulation procedure is run for N_s times, from which a total of $N_s N_h$ state-input pairs are obtained.

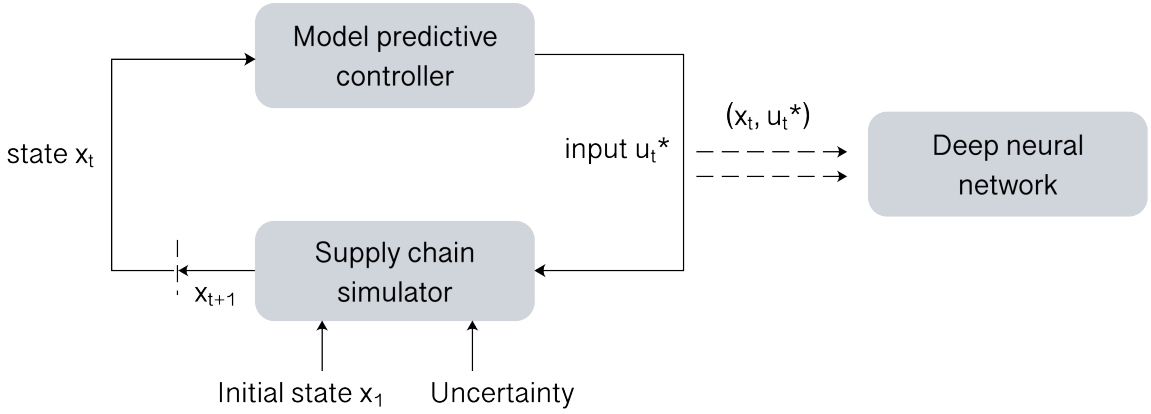


Figure 5.3. Sampling of optimal state-input pairs based on closed-loop simulation.

It should be noted that in closed-loop simulation, MPC tends to determine inputs that result in, e.g., low inventory of raw materials at plants, low inventory of final products at distribution sites of the supply chain. This is because MPC minimizes the inventory cost. Therefore, the state enter a small region ("sweet spot") of the state space (denoted as $x \in \mathcal{S}_s \subset \mathcal{S}$) soon after the simulation starts, even it starts from a random initial state. In the collected data, samples in $\mathcal{S}_s$ may make up a large portion. However, this is a small region in the entire state-space, and cannot cover the entire state space $\mathcal{S}$. This means the DNN is trained mainly for $\mathcal{S}_s$ and ignores the rest of the state space. Thus, the approximation for $\mathcal{S}_s$ may be accurate, but overfitting may happen, and affect the overall performance. For an initial

state that falls outside the sweet spot, the approximation error could be large. In order to avoid overfitting into the subset of state space and obtain more accurate approximation, besides the closed-loop data from simulation, the DNN is supplied with more random initial states in this work. This means a hybrid of closed-loop data and random optimal state-input pairs sampled from the state space are used. This approach improves the overall accuracy of approximation.

The state and input data are organized as matrices, with each row represents a sample, and each column corresponds to a variable. Before using the data for training the DNN controller, the data need to be preprocessed. In this chapter, data preprocessing is implemented as follows (Karg and Lucia, 2021; Kumar et al., 2021):

(1) Remove the columns with constant values in the state and input data. These columns are corresponding to the state and input variables that have constant values for all samples.

(2) Split the data into a training set and a validation set. The training set accounts for 80% of the data, which is used for fitting the DNN. The validation set accounts for the remaining 20% of the data, which is used for selecting hyperparameters in the training step.

(3) Scale each state and input variable to the range of $[0, 1]$. A variable x is scaled by using:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.24}$$

where x_{min} and x_{max} are the minimum and maximum of the samples of x, respectively; and x_{scaled} is the scaled variable.

The scaling parameters x_{min} and x_{max} are calculated based on the training set. The scaling is also implemented on the validation set and test set using the same parameters.

The test set for closed-loop testing is generated separately such that MPC and DLBMPC can be compared under the same initial state and demand realization. The DNN is blind to the test set before online testing.

5.2.5 Feasibility recovery

In terms of dealing with mixed-integer linear or mixed-integer quadratic MPC, basically there are two paradigms: (1) Use a NN to approximate and predict the binary decision variables, then in the online phase solve a LP or QP with the binary variables fixed to their predicted values (Masti and Bemporad, 2019; Masti et al., 2020; Srinivasan et al., 2021); (2) Use a NN to approximate the binary variables and continuous variables (Karg and Lucia, 2018). Masti and Bemporad (2019) use a NN to approximate the binary decisions of multiparametric MIQP problems. In the online phase, a QP is solved with all the binary variables set to the predicted values. If the QP is infeasible, a sequence of MIQPs is solved with the binary variable unlocked one by one from their predicted values to {0, 1}. Masti et al. (2020) use a machine learning approach to predict the binary decisions of a multiparametric MILP, so that only a LP is solved online. Infeasibility is avoided through a simple rule-based engine that modifies the binary configuration when necessary. Srinivasan et al. (2021) fix the values of most or all integer variables in MILPs based on the predicted trajectories, and then the resulting simpler MILP is solved. A further improvement is solving the full MILP initialized at the above solution with a timeout for real-time feasibility.

Since the output of the NN is an approximation of the optimal solution and yields floating point numbers, infeasibility may exist, especially when there are integer decision variables. Therefore, feasibility recovery after obtaining the prediction of NN is an important step. In this work, the binary and continuous decision variables are approximated by the same DNN. The following heuristic method is used to deal with the binary variables in the mixed-integer linear MPC and recover the feasibility of the decisions obtained from the NN:

(1) For the binary decision variables $\{u^p_{ps,t}\}_{p\in S_P, ps\in S^p_{PS}}$, a threshold of 0.5 is used to adjust the prediction of the NN, $\hat{u}^p_{ps,t}$:

$$\hat{u}^p_{ps,t} = \begin{cases} 0, & \text{if} \quad \hat{u}^p_{ps,t} \leq 0.5 \\ 1, & \text{if} \quad \hat{u}^p_{ps,t} > 0.5 \end{cases} \tag{5.25}$$

Equation (5.25) means if the prediction of NN for a binary variable $\hat{u}^p_{ps,t}$ is less or equal to 0.5, then it is set to 0; otherwise it is set to 1.

(2) Adjust $\hat{u}^p_{ps,t}$ according to past production decisions based on Equation (5.6b):

If $\sum_{t'=t-1}^{t-\delta^p_{ps}+1} \hat{u}^p_{ps,t'} = 1$ (for $\delta^p_{ps} \geq 2$), then set $\hat{u}^p_{ps,t} \leftarrow 0$.

This adjustment aims to mitigate the potential error in the prediction $\hat{u}^p_{ps,t}$ in step (1), thus ensuring that the constraint on the binary variables, i.e., $\sum_{t'=t-1}^{t-\delta^p_{ps}+1} \hat{u}^p_{ps,t'} + \hat{u}^p_{ps,t} \leq 1$ required by Equation (5.6b), is satisfied.

(3) Adjust the production quantity $\hat{W}^p_{ps,t}$ based on the associated binary production decision $\hat{u}^p_{ps,t}$ and the minimum and maximum batch sizes (Equation (5.6a)):

$$\hat{W}^p_{ps,t} = \begin{cases} 0, & \text{if} \quad \hat{u}^p_{ps,t} = 0 \\ \max\{W^{p,lb}_{ps}, \min\{W^{p,ub}_{ps}, \hat{W}^p_{ps,t}\}\}, & \text{if} \quad \hat{u}^p_{ps,t} = 1 \end{cases} \tag{5.26}$$

Equation (5.26) ensures that the production quantity $\hat{W}^p_{ps,t}$ is 0 when $\hat{u}^p_{ps,t} = 0$, and is within the lower and upper bounds when $\hat{u}^p_{ps,t} = 1$.

(4) Update the current inventory state variables with the past decisions based on Equation (5.1). For example:

$I^{p,tmp}_{j,t+1} \leftarrow I^p_{j,t} + \sum_{ps \in O^p_j} \mu^p_{ps,j} \beta^p_{ps} W^p_{ps,t-\delta^p_{ps}}, \forall p \in S_P, j \in S^p_F$.

Here, $I^{p,tmp}_{j,t+1}$ is the maximum available inventory of product j at plant site p that can be shipped to the distribution centers. It can be used to adjust the shipment quantities $\hat{Q}^{p,dc}_{j,t}$, as shown in the next step.

(5) Adjust the input variables according to the updated state variables calculated in the last step. For example, $\hat{Q}^{p,dc}_{j,t}$ is adjusted based on Equation (5.1c) as follows:

If $\Delta = \sum_{dc \in S_{DC}} \hat{Q}^{p,dc}_{j,t} - I^{p,tmp}_{j,t+1} > 0$, then set $\hat{Q}^{p,dc}_{j,t} \leftarrow \max\{0, \hat{Q}^{p,dc}_{j,t} - \Delta\}$.

This adjustment ensures that the total shipment quantities do not exceed the current available inventory. The production quantity $\hat{W}^p_{ps,t}$ and shipment quantity $\hat{Q}^{p,p'}_{j,t}$ are adjusted in a similar way based on Equations (5.1a) and (5.1b), respectively.

It is worth mentioning that the approximation of DNN is (supposed to be) generally accurate, therefore the adjustment should not be significant.

5.3 Case Studies

In this section, two case studies are carried out to validate the deep learning-based MPC in supply chain optimization. The first case is an illustrative example based on the beer game supply chain, and the second case involves a more complicated supply chain network. The two case studies are presented in Section 5.3.1 and Section 5.3.2, respectively.

5.3.1 Case study I

The Beer Distribution Game is a classical roleplay simulation of an industrial production-distribution system in the operations management area (Sterman, 1989; Chen and Samroengraja, 2000). As shown in Figure 5.4, the beer game involves a 4-echelon serial supply chain that consists of a factory, a distributor, a wholesaler, and a retailer. It deals with the production and distribution of a beer product to satisfy customer orders. The beer game has been widely used in operations management as an example to illustrate the dynamics in supply chain system and to investigate supply chain control (Oroojlooyjadid et al., 2022).

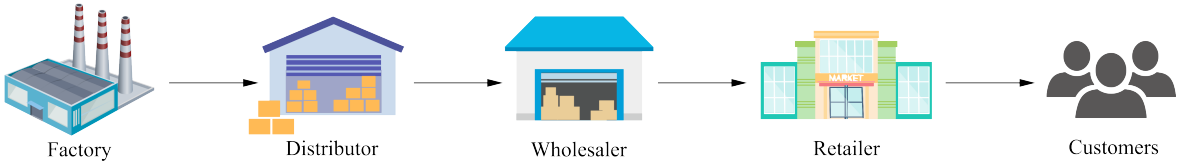


Figure 5.4. Case study I: the beer game supply chain.

Based on Sterman (1989), the following assumptions are made for the supply chain control problem in this work:

(1) Each supply chain member holds an inventory of the beer product.

(2) Each supply chain member receives orders from and ships beer to its downstream.

(3) Each supply chain member (except for the factory) orders beer from its upstream component and receives beer after a shipping delay. The factory places production requests and receives beer after a production delay.

(4) There is no information delay.

(5) The retailer uses a "best I can do" policy and allows backorders. If the retailer's inventory is insufficient to fill incoming orders plus backlog, it fills as many orders as it can and add the remaining orders to the backlog.

The MPC problem for the operation of this supply chain is formulated as follows:

$$
\begin{aligned}
&\min_{u} \sum_{t=1}^{N} \omega_1 \left(\pi^p I_t^p + \pi^{dc} I_t^{dc} + \pi^w I_t^w + \pi^r I_t^r \right) + \omega_2 B_t^r \\
\text{s. t.} \quad & I_{t+1}^p = I_t^p - Q_t^{p,dc} + W_{t-\delta_p}^p, && \forall t \in T \\
& I_{t+1}^{dc} = I_t^{dc} - Q_t^{dc,w} + Q_{t-\delta_{p,dc}}^{p,dc}, && \forall t \in T \\
& I_{t+1}^w = I_t^w - Q_t^{w,r} + Q_{t-\delta_{dc,w}}^{dc,w}, && \forall t \in T \\
& IL_{t+1}^r = IL_t^r + Q_{t-\delta_{w,r}}^{w,r} - D_t^r, && \forall t \in T \\
& B_t^r \geq -IL_t^r, && \forall t \in T \\
& I_t^r \geq IL_t^r + B_t^r, && \forall t \in T \\
& I_t^p, I_t^{dc}, I_t^w, I_t^r, B_t^r, W_t^p, Q_t^{dc,w}, Q_t^{w,r} \geq 0, && \forall t \in T \\
& I_1^p = I_0^p, I_1^{dc} = I_0^{dc}, I_1^w = I_0^w, IL_1^r = IL_0^r
\end{aligned}
\tag{5.27}
$$

where the superscripts p, dc, w, and r represent the factory, distributor, wholesaler, and retailer, respectively. As has been presented in Section 5.2, I_t^p, I_t^{dc}, I_t^w, I_t^r, and IL_t^r are the inventory of the corresponding agent at time period t; B_t^r represents the back orders of the retailer. I_0^p, I_0^{dc}, I_0^w, and IL_0^r are the initial inventory. W_t^p, $Q_t^{p,dc}$, $Q_t^{dc,w}$, and $Q_t^{w,r}$ are the production and shipment quantities; δ_p, $\delta_{p,dc}$, $\delta_{dc,w}$, and $\delta_{w,r}$ are the production and shipment delays. D_t^r is the demand forecast at the retailer.

The objective function is a weighted sum of the total inventory holding cost and the penalty for back orders. The unit inventory holding costs of the 4 agents, π^p, π^{dc}, π^w, and π^r are

set to 0.25, 0.5, 0.75, and 1 \$/unit/period, respectively; the weights ω_1 and ω_2 are set to 1 and 10, respectively (Chen and Samroengraja, 2000).

The state variables for time period t, including the additional states for modelling past decisions, are:

$$\begin{aligned} x_t = [&I_t^p, I_t^{dc}, I_t^w, IL_t^r, W_{t,1}^p, \ldots, W_{t,\delta_p}^p, \\ &Q_{t,1}^{p,dc}, \ldots, Q_{t,\delta_{p,dc}}^{p,dc}, Q_{t,1}^{dc,w}, \ldots, Q_{t,\delta_{dc,w}}^{dc,w}, Q_{t,1}^{w,r}, \ldots, Q_{t,\delta_{w,r}}^{w,r}]^T \end{aligned} \tag{5.28}$$

The auxiliary state variables for time period t are $a_t = [I_t^r, B_t^r]^T$. The input variables for time period t are $u_t = [W_t^p, Q_t^{p,dc}, Q_t^{dc,w}, Q_t^{w,r}]^T$, and the decision variables of the MPC problem are $u = [u_1^T, u_2^T, \ldots, u_N^T]^T$.

In this illustrative example, the delays δ_p, $\delta_{p,dc}$, $\delta_{dc,w}$, and $\delta_{w,r}$ are all set to 1. Therefore, the state is a 8-dimensional vector $x_t = [I_t^p, I_t^{dc}, I_t^w, IL_t^r, W_{t,1}^p, Q_{t,1}^{p,dc}, Q_{t,1}^{dc,w}, Q_{t,1}^{w,r}]^T$. The initial state is $x_0 = [I_0^p, I_0^{dc}, I_0^w, IL_0^r, W_{0,1}^p, Q_{0,1}^{p,dc}, Q_{0,1}^{dc,w}, Q_{0,1}^{w,r}]^T$, where $IL_0^r = I_0^r - B_0^r$. For each time period t, the MPC problem is solved for given updated state x_0 to obtain the optimal decision u^* for the current period, and $u_1^* = [W_1^{p*}, Q_1^{p,dc*}, Q_1^{dc,w*}, Q_1^{w,r*}]^T$ are implemented at the next period.

Table 5.1. Case study I: supply chain parameters.

Parameters	Value
Factory unit inventory cost π^p	0.25
Distributor unit inventory cost π^{dc}	0.5
Wholesaler unit inventory cost π^w	0.75
Retailer unit inventory cost π^r	1
Weight ω_1	1
Weight ω_2	10
Planning horizon N	7
Demand	$\mathcal{N}(20, 4^2)$
Demand forecast D_t	20

For simplification, it is assumed that there are no capacity constraints on the storage. The customer demand is assumed to follow a normal distribution $\mathcal{N}(20, 4^2)$. In the deterministic MPC formulation, the demand forecasts over the planning horizon are set to the mean 20.

The planning horizon is selected to be $N = 7$. The parameters are summarized in Table 5.1.

It is worth mentioning that in the classical beer game, the beer product is measured in cases (Sterman, 1989) or kegs (Chen and Samroengraja, 2000), thus the decision variables are integers. In this work, the supply chain control problem is adapted to allow continuous decision variables, thus it is more general. The MPC optimization problem is formulated as a linear program.

Implementation of the DLBMPC technique follows an offline-online procedure. In the offline phase, a DNN $u_1 = \mathcal{DNN}(x_0)$ is developed to construct a map from the state x_0 to the control input u_1. This DNN is fed with optimal state-input pairs to approximate the explicit MPC law $u_1^* = \mathcal{MPC}(x_0)$. The optimization problem for the supply chain MPC is modelled using the `Python` optimization package `Pyomo` (Hart et al., 2011; Bynum et al., 2021) and solved using the `IBM CPLEX 12.10` solver. The problem is solved offline for many initial system states x_0 to obtain corresponding optimal inputs u_1^*. These state-inputs pairs (x_0, u_1^*) are collected and stored as the training data for the DNN controller. Since the demand forecast in the MPC problem is a constant, it is not considered as an input to the DNN. For the case where the demand forecast is varying, the DNN should map from the initial state and demand forecast to the control inputs. The state variables used as the input to the DNN are $\bar{x}_0 = [I_0^p, I_0^{dc}, I_0^w, I_0^r, B_0^r, W_{0,1}^p, Q_{0,1}^{p,dc}, Q_{0,1}^{dc,w}, Q_{0,1}^{w,r}]^T \in \mathbb{R}^9$. The decisions used as the output of the DNN are $u_1 = [W_1^p, Q_1^{p,dc}, Q_1^{dc,w}, Q_1^{w,r}]^T \in \mathbb{R}^4$.

The DNN is built by using the `tensorflow.keras.Sequential()` API provided by the machine learning platform `TensorFlow` (Martín Abadi et al., 2015). The hyperparameters of DNN and relevant information are summarized in Table 5.2. In this case, the linear MPC problem has a state dimension of 9. Therefore, the input layer of the DNN has 9 neurons. The output layer has 4 neurons, which is the number of decision variables. The DNN has 5 hidden layers, with 50 neurons in each hidden layer. The DNN maps from the state to the decisions, which is $u_1 = \mathcal{DNN}(x_0) : \mathbb{R}^9 \to \mathbb{R}^4$.

In order to train the DNN, 5000 optimal state-input pairs are generated through closed-loop simulation, and another 5000 optimal state-input pairs are generated for states randomly sampled from the state space. The supply chain simulation model is developed using the

programming language `Python` to generate closed-loop samples and validate the method. For the closed-loop simulation, the simulation horizon is set to 50 time periods, and the simulation procedure is run for 100 replications, starting from different random initial states. 80% of the data are used as training set, i.e., 4000 closed-loop samples plus 4000 random samples. The remaining 2000 closed-loop samples and 2000 random samples are used to construct the validation set. For the random state samples, each state variable is sampled from a uniform distribution $\mathcal{U}(0, 50)$, except for the backorder, which is set to 0. The DNN is trained using the Adam optimizer for 100 epochs with a learning rate of 0.001. For the scaled data where each variable is scaled to $[0, 1]$, the DNN achieved a root-mean-squared-error (RMSE) of 0.003 on the training set, and a RMSE of 0.004 on the validation set. In the original scale, the RMSEs are 0.14 and 0.17, respectively.

Table 5.2. Case study I: DNN hyperparameters and relevant information

Hyperparameters	Value
Number of neurons in input layer	9
Number of hidden layers	5
Number of neurons per hidden layer	50
Number of neurons in output layer	4
Activation function	ReLUs
Batch size	32
Epochs	100
Optimizer	Adam
Learning rate	0.001
Number of training samples	8,000
Number of validation samples	2,000

In the online implementation phase, a total of $N_s^{test} = 200$ random scenarios are generated as the test set for closed-loop validation. Each scenario is a combination of a random initial state and a demand sample. The MPC and DLBMPC are tested and compared under the same scenarios. The simulation horizon is set to 30. The comparison of the state and input over the simulation horizon using the two control methods under a scenario is shown in Figure 5.5. The column on the left shows the state variables including inventory of the agents, the backorders of the retailer and the customer demand. The column on the right shows the input variables including the production and shipment quantities. The simulation starts

from an initial state where the inventories and past decisions are sampled randomly. It is observed that soon after the simulation starts, the agents tend to maintain low inventories to avoid inventory holding costs. This is the "sweet spot" mentioned in the previous section. It can be seen that the state trajectory and the input obtained from the DLNMPC is almost identical to the MPC, and can hardly be distinguished. This implies that the DNN controller can approximate the model predictive controller well. It is worth mentioning that the DNN controller is blind to (not trained for) the test samples in the offline training phase. Therefore, the closed-loop performance is excellent.

Similar to Kumar et al. (2021), a performance loss metric is defined in this work to assess the DLBMPC method in closed-loop implementation. Under a scenario s, the total cost over the simulation horizon, denoted as C_{mpc} and C_{dlbmpc} for MPC and DLBMPC, respectively, are calculated as $\sum_{t=1}^{N} \omega_1 \left(\pi^p I_t^p + \pi^{dc} I_t^{dc} + \pi^w I_t^w + \pi^r I_t^r\right) + \omega_2 B_t^r$, which is the objective function in the MPC problem. The performance loss using DLBMPC under a scenario s is then defined as:

$$\Delta_{loss}^{s} = \frac{C_{dlbmpc} - C_{mpc}}{C_{mpc}} \times 100\% \tag{5.29}$$

The average performance loss over the N_s^{test} scenarios is calculated as:

$$\Delta_{loss}^{avg} = \frac{1}{N_s^{test}} \sum_{s} \Delta_{loss}^{s} \times 100\% \tag{5.30}$$

Under different scenarios, the total cost is different. The results over the 200 test scenarios are presented in Figure 5.6. The distributions of the total cost using the two methods are shown in the box plot on the left of Figure 5.6. A box plot is a graphical summary of a dataset based on the median, the minimum and maximum (excluding any outliers), and the first and third quartiles (Frigge et al., 1989). The small circles in the plot represent the outliers, which are the samples beyond the boundary of the whiskers. It can be seen that two box plots are very similar, in terms of the five-number summary. The plot on the right of Figure 5.6 shows the spread of performance loss (Δ_{loss}^{s}) over the 200 scenarios. It is observed that the median is greater than 0. This means for at least a half of the scenarios, the DLBMPC causes performance loss. For most scenarios, the loss is less than 2%. However, it is worth

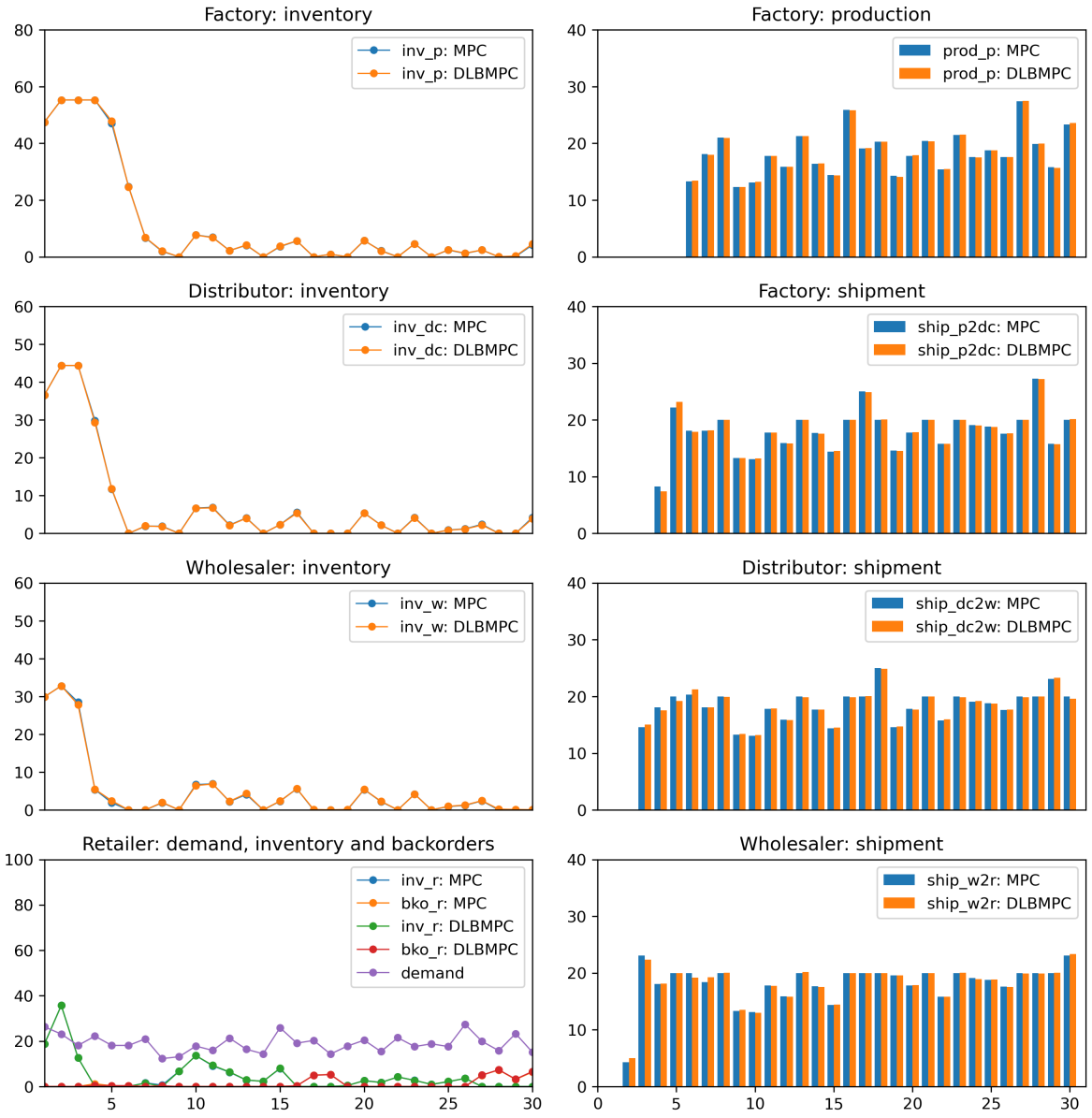


Figure 5.5. Case study I: closed-loop simulation results for a test scenario.

mentioning that for some scenarios, the DLBMPC achieved a lower total cost than MPC. This is because uncertainty exists in the demand, and MPC cannot guarantee an optimal decision in real implementation. The highest outlier in the plot on the right indicates that the worst-case performance loss of DLBMPC over the test scenarios is approximately 4.5%. The lowest outlier indicates that in the best case, DLBMPC improves the performance by approximately 4.5%. The average performance loss of using the DLBMPC controller is $\Delta_{loss}^{avg} = 0.43\%$. The loss is very small, which indicates that the DLBMPC achieved good performance in the closed-loop implementation. The DNN is able to approximate the MPC solutions to a high accuracy.

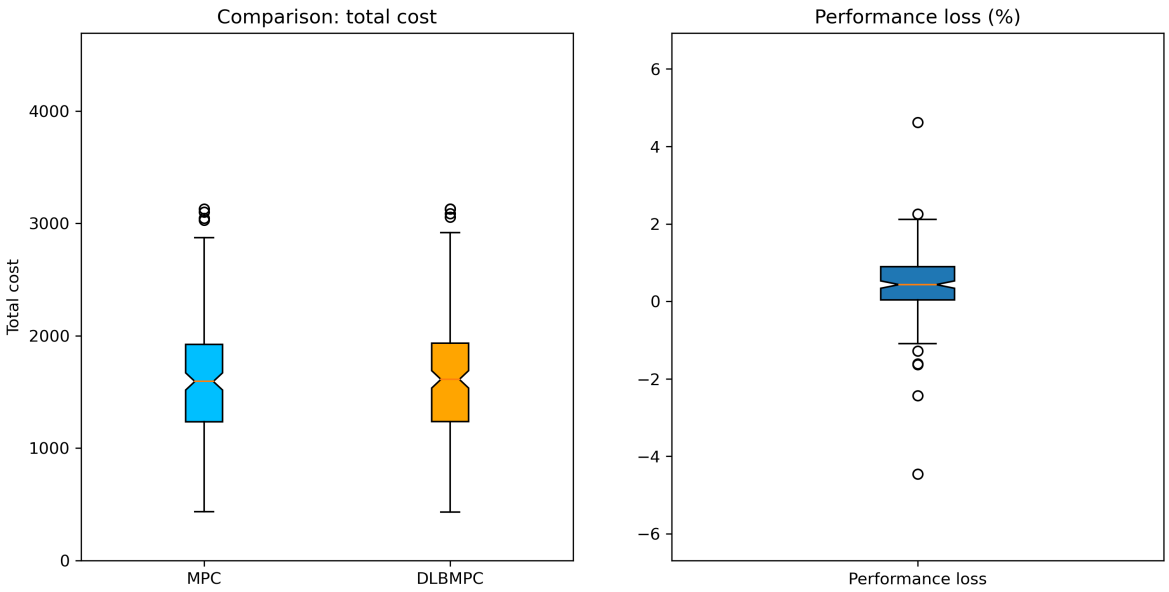


Figure 5.6. Case study I: comparison of closed-loop performance.

5.3.2 Case study II

In the second case study, a multi-echelon and multi-product supply chain is investigated. This supply chain network has been investigated by Mastragostino et al. (2014), and is more complicated than the serial supply chain in case study I. In this section, the structure of the supply chain and the MPC problem are described. The development and implementation of the DNN controller is discussed. The performance of the deep learning-based MPC is

validated through closed-loop simulation and compared with the nominal MPC.

The structure of this supply chain is shown in Figure 5.7 (Mastragostino et al., 2014). This supply chain involves 2 suppliers ($S_S = \{S_1, S_2\}$), 2 plant sites ($S_P = \{P_1, P_2\}$), and 2 distribution sites ($S_{DC} = \{DC_1, DC_2\}$). Plant site P_1 has two production schemes PS_1 and PS_2 installed ($S_{PS}^{P_1} = \{PS_1, PS_2\}$). Plant site P_2 has two production schemes PS_3 and PS_4 installed ($S_{PS}^{P_2} = \{PS_3, PS_4\}$). The mass balances of the 4 production schemes are given as follows:

$$
\begin{aligned}
PS_1: &\quad A + B \rightarrow C + D \\
PS_2: &\quad C \rightarrow 0.5E + 0.5G \\
PS_3: &\quad A + D \rightarrow E + F \\
PS_4: &\quad F \rightarrow G
\end{aligned}
$$

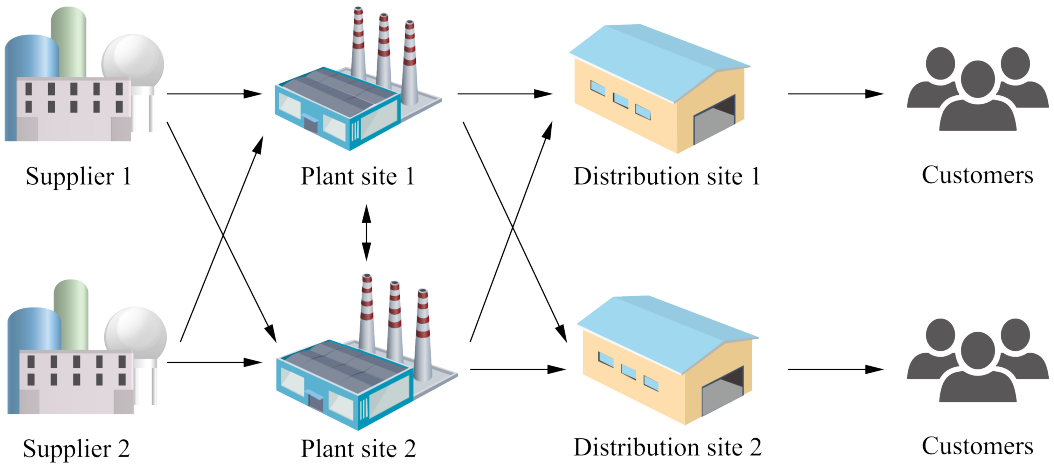


Figure 5.7. Case study II: supply chain network.

The supply chain deals with the production and distribution of products to satisfy customers' demand. The plant sites purchase raw materials from the suppliers. For plant P_1, A and B are the raw materials, C and D are the intermediate products, and E and G are the final products. For plant P_2, A and D are the raw materials, F is the intermediate product, and E and G are the final products. A is purchased from suppliers and D is shipped from P_1. The final products are shipped to the distribution sites, where customer orders are fulfilled. The plant sites hold inventory of the raw materials and products. The distribution sites DC_1 and DC_2 use a "best I can do" policy.

For the supply chain configuration in this case study, the mass balances for plant P_1 and P_2 are expressed in detail as Equation (5.31):

$$I^{P_1}_{j,t+1} = I^{P_1}_{j,t} + \sum_{s \in S_S} O^{s,P_1}_{j,t-\delta^{s,P_1}} - \mu^{P_1}_{PS_1,j} W^{P_1}_{PS_1,t}, \qquad \forall j \in \{A,B\}, t \tag{5.31a}$$

$$I^{P_1}_{C,t+1} = I^{P_1}_{C,t} + \mu^{P_1}_{PS_1,C} \beta^{P_1}_{PS_1} W^{P_1}_{PS_1,t-\delta^{P_1}_{PS_1}} - \mu^{P_1}_{PS_2,C} W^{P_1}_{PS_2,t}, \qquad \forall t \tag{5.31b}$$

$$I^{P_1}_{D,t+1} = I^{P_1}_{D,t} + \mu^{P_1}_{PS_1,D} \beta^{P_1}_{PS_1} W^{P_1}_{PS_1,t-\delta^{P_1}_{PS_1}} - Q^{P_1,P_2}_{D,t}, \qquad \forall t \tag{5.31c}$$

$$I^{P_1}_{j,t+1} = I^{P_1}_{j,t} + \mu^{P_1}_{PS_2,j} \beta^{P_1}_{PS_2} W^{P_1}_{PS_2,t-\delta^{P_1}_{PS_2}} - \sum_{dc \in S_{DC}} Q^{P_1,dc}_{j,t}, \qquad \forall j \in \{E,G\}, t \tag{5.31d}$$

$$I^{P_2}_{A,t+1} = I^{P_2}_{A,t} + \sum_{s \in S_S} O^{s,P_2}_{A,t-\delta^{s,P_2}_{A}} - \mu^{P_2}_{PS_3,A} W^{P_2}_{PS_3,t}, \qquad \forall t \tag{5.31e}$$

$$I^{P_2}_{D,t+1} = I^{P_2}_{D,t} + Q^{P_1,P_2}_{D,t-\delta^{P_1,P_2}} - \mu^{P_2}_{PS_3,D} W^{P_2}_{PS_3,t}, \qquad \forall t \tag{5.31f}$$

$$I^{P_2}_{E,t+1} = I^{P_2}_{E,t} + \mu^{P_2}_{PS_3,D} \beta^{P_2}_{PS_3} W^{P_2}_{PS_3,t-\delta^{P_2}_{PS_3}} - \sum_{dc \in S_{DC}} Q^{M_2,dc}_{E,t}, \qquad \forall t \tag{5.31g}$$

$$I^{P_2}_{F,t+1} = I^{P_2}_{F,t} + \mu^{P_2}_{PS_3,D} \beta^{P_2}_{PS_3} W^{P_2}_{PS_3,t-\delta^{P_2}_{PS_3}} - \mu^{P_2}_{PS_4,F} W^{P_2}_{PS_4,t}, \qquad \forall t \tag{5.31h}$$

$$I^{P_2}_{G,t+1} = I^{P_2}_{G,t} + \mu^{P_2}_{PS_4,G} \beta^{P_2}_{PS_4} W^{P_2}_{PS_4,t-\delta^{P_2}_{PS_4}} - \sum_{dc \in S_{DC}} Q^{P_2,dc}_{G,t}, \qquad \forall t \tag{5.31i}$$

The MPC problem is formulated based on Section 5.2.1. Since the customer demand is uncertain, overstock is allowed at the distribution sites and soft constraints are imposed to avoid potential infeasibility in closed-loop implementation. The objective function of the MPC problem is to minimize the total weighted sum of the economic performance, the customer service (measured by backorders), and a penalty for overstock. The weights assigned to the 3 objectives are chosen as $\omega_1 = 1$, $\omega_2 = 10$, and $\omega_3 = 5$, respectively. The economic performance is measured by the total operating cost over the planning horizon, which consists of the purchase cost, production cost, transportation cost, and inventory cost.

The supply chain parameters are given in Tables 5.3–5.9 consistent with Mastragostino et al. (2014). Table 5.3 presents the unit purchase costs. Table 5.4 presents the production parameters, including production cost, batch size, production delay and process yield. Table 5.5, Table 5.6, and Table 5.7 present the transportation cost, delay, and maximum capacity,

respectively. Table 5.8 presents the inventory cost and capacity. The demand of final products at the distribution sites are assumed to follow normal distributions. The mean and standard deviation of the normal distributions are presented in Table 5.9.

Table 5.3. Case study II: raw material purchase cost parameters α_j^s.

Raw material	Suppliers	
	S_1	S_2
A	1	1.2
B	1.4	1.7

Table 5.4. Case study II: production parameters.

Parameters	Production schemes			
	PS_1	PS_2	PS_3	PS_4
Cost ρ_{ps}^p	1.25	1	1.3	1.5
Maximum batch size $W_{ps}^{p,ub}$	120	300	120	150
Minimum batch size $W_{ps}^{p,lb}$	25	60	25	25
Production delay δ_{ps}^p	1	2	2	1
Process yield parameter β_{ps}^p	0.8	0.7	0.8	0.7

Table 5.5. Case study II: shipment cost parameters $\tau^{p,dc}$, $\tau^{p,p'}$.

Plant sites	Plant sites		Distribution sites	
	P_1	P_2	D_1	D_2
P_1	–	4	2.7	2.8
P_2	4	–	2.8	2.5

The state variables for time period t, x_t in Equation (5.19) (plus past binary variables) is a 71-dimensional vector. The auxiliary state variables for time period t are $a_t = [I_t^r, B_t^r]^T$. The input variables for time period t, u_t in Equation (5.19) is a 23-dimensional vector. The decision variables of the MPC problem are $u = [u_1^T, u_2^T, \ldots, u_N^T]^T$. For each time period t, the MPC problem is solved for updated state x_0 to obtain the optimal decision u^* for the current period, and u_1^* are implemented at the next period.

In the offline development of the deep learning-based controller, the mixed-integer linear MPC problem is modelled using `Pyomo` and solved using the `IBM CPLEX 12.10` solver

Table 5.6. Case study II: transportation delays $\delta_{p,dc}$, $\delta_{p,p'}$.

Plant sites	Suppliers		Distribution sites		Plant sites	
	S_1	S_2	D_1	D_2	P_1	P_2
P_1	3	3	3	4	0	2
P_2	2	2	4	2	2	0

Table 5.7. Case study II: maximum transportation quantity $Q^{s,p,ub}$, $Q^{p,p',ub}$, $Q^{p,dc,ub}$.

Plant sites	Suppliers		Distribution sites		Plant sites	
	S_1	S_2	D_1	D_2	P_1	P_2
P_1	120	120	100	100	0	100
P_2	120	120	100	100	100	0

Table 5.8. Case study II: inventory parameters.

Parameters	Plant sites		Distribution sites	
	P_1	P_2	D_1	D_2
Inventory cost of S_R^p	0.8	0.7	–	–
Inventory cost of S_I^p	0.9	1.1	–	–
Inventory cost of S_E^p	1.4	1.1	–	–
Inventory cost of S_F	–	–	1.5	1.25
Storage capacity $I_R^{p,ub}/I_I^{p,ub}/I_E^{p,ub}/I_G^{p,ub}$	500	500	–	–
Storage capacity $I_E^{dc,ub}/I_G^{dc,ub}$	–	–	500	500

Table 5.9. Case study II: demand parameters.

Parameters	Distribution site DC_1		Distribution site DC_2	
	E	G	E	G
mean	20	20	40	20
std.	4	4	8	4

to obtain optimal state-input data. The state x_0 is augmented with past binary decision and auxiliary states to obtain $\bar{x}_0$. Then a DNN $u_1 = \mathcal{DNN}(\bar{x}_0)$ is built and trained to construct a map from the augmented state $\bar{x}_0$ to the control input u_1. This DNN is fed with optimal state-input pairs to approximate the MPC law $u_1^* = \mathcal{MPC}(x_0)$. The problem is solved offline for many initial system states x_0 to obtain corresponding optimal inputs u_1^*. These state-inputs pairs $(\bar{x}_0, u_1^*)$ are collected and stored as the training data for the DNN controller.

In order to train the DNN, 10^5 optimal state-input pairs are generated through closed-loop simulation, and another 10^5 optimal state-input pairs are generated for states randomly sampled from the state space. The supply chain simulation model is developed using `Python` to generate closed-loop samples and validate the method. For the closed-loop simulation, the simulation horizon is set to 40 time periods, and the simulation procedure is run for 2500 replications, starting from different random initial states. 80% of the data are used as training set, i.e., 8×10^4 closed-loop samples plus 8×10^4 random samples. The remaining 2×10^4 closed-loop samples and 2×10^4 random samples are used to construct the validation set. For the random state samples, the inventory variables are sampled from a uniform distribution $\mathcal{U}(0, 200)$, and the backorders and past decision variables are set to 0. The generation of 2×10^5 samples takes around 11 hours.

For the mixed-integer MPC problem in this case study, $\bar{x}_0 \in \mathbb{R}^{71}$, and $u_1^* \in \mathbb{R}^{23}$. In the preprocessing step, it is observed that in u_1^* of the MPC solution, three decision variables $Q_{A,t}^{S_2,P_1}$, $Q_{B,t}^{S_2,P_1}$, and $Q_{A,t}^{S_2,P_2}$ are always zeros. Therefore, the corresponding components of the state and input vectors are removed to obtain $\hat{x}_0$ and $\hat{u}_1^*$. By removing $Q_{A,0,k}^{S_2,P_1}, k = 1, 2, 3$, $Q_{B,0,k}^{S_2,P_1}, k = 1, 2, 3$, and $Q_{A,0,k}^{S_2,P_2}, k = 1, 2$ from $\bar{x}_0$, the state variables used as the input to the DNN are reduced from $\bar{x}_0 \in \mathbb{R}^{71}$ to $\hat{x}_0 \in \mathbb{R}^{63}$. Also, by removing $Q_{A,1}^{S_2,P_1}$, $Q_{B,1}^{S_2,P_1}$, and $Q_{A,1}^{S_2,P_2}$ from u_1^*, the decisions used as the output of the DNN are reduced from $u_1^* \in \mathbb{R}^{23}$ to $\hat{u}_1^* \in \mathbb{R}^{20}$. Since the demand forecast in the MPC problem is a constant, it is not considered as an input to the DNN. Each state and input variable is scaled to the range of $[0, 1]$.

The DNN is built by using the `tensorflow.keras.Sequential()` API. The hyperparameters of DNN and relevant information are summarized in Table 5.10. The input layer and

output layer of the DNN has 63 and 20 neurons, respectively. The DNN has 5 hidden layers, with 500 neurons in each hidden layer. The DNN maps from the state to the decisions, which is $\hat{u}_1 = \mathcal{DNN}(\hat{x}_0) : \mathbb{R}^{63} \rightarrow \mathbb{R}^{20}$. Here, $\hat{u}_1$ is an approximation of the reduced optimal decisions $\hat{u}_1^*$. Once $\hat{u}_1$ is computed by the DNN controller, the approximated MPC solution $u_1 \in \mathbb{R}^{23}$ is obtained from augmenting $\hat{u}_1$ with $Q_{A,1}^{S_2,P_1}, Q_{B,1}^{S_2,P_1}, Q_{A,1}^{S_2,P_2} = 0$.

Table 5.10. Case study II: DNN hyperparameters and relevant information.

Hyperparameters	Value
Number of neurons in input layer	63
Number of hidden layers	5
Number of neurons per hidden layer	400
Number of neurons in output layer	20
Activation function	ReLUs
Batch size	2048
Epochs	400
Optimizer	Adam
Learning rate	0.001
Number of training samples	1.6×10^5
Number of validation samples	4×10^4

The DNN is trained using the Adam optimizer for 400 epochs with a learning rate of 0.001. For the scaled data where each variable is scaled to $[0, 1]$, the DNN achieved a RMSE of 0.024 on the training set, and a RMSE of 0.076 on the validation set. In the original scale, the RMSEs are 2.71 and 6.45, respectively. For the prediction of binary variables, the DNN achieved an error rate of $\approx 0.05\%$ for the training set, and 1.77% for the validation set.

For the online closed-loop validation, a total of $N_s^{test} = 100$ random scenarios are generated as the test set. Each scenario consists of an initial state randomly sampled from the state space, and the demand of final products over the simulation horizon. The MPC and DLBMPC are tested and compared under the same initial state and demand realization. The simulation horizon is set to 40 time periods. Figure 5.8 shows the comparison of some state and input variables using the two control methods under a test scenario, with the state variables such as inventory, backorder, and demand shown by the column on the left, and the input variables such as the production and shipment quantities shown by the column on the right. Similarly to case study I, in this case it is also observed that the supply chain system enters a "sweet

spot" soon after the simulation starts. This means for most of the time, the inventories of most materials are low such that high inventory holding costs are avoided. The trajectories of the state and input variables obtained from the DLNMPC are very close to the MPC, and are barely distinguishable for some variables. This implies that the decisions obtained from the DNN controller are a good approximation of the MPC solutions, and the closed-loop performance of the DNN controller is excellent.

The results over the 100 test scenarios are presented in Figure 5.9. The distributions of the total cost using the two methods are shown in the box plot on the left of Figure 5.9. It can be seen that two box plots are very similar, in terms of the median, mean and quantiles. The plot on the right of Figure 5.9 shows the spread of performance loss (Δ^{s}_{loss}) over the 100 scenarios. It is observed that the median is approximately 1%. For 75% of the scenarios, the loss is less than 2.5%. Similar to case study I, for some scenarios, the DLBMPC achieved a lower total cost than MPC. This is because uncertainty exists in the demand, and MPC cannot guarantee an optimal decision in real implementation. The average performance loss of using the DLBMPC controller is $\Delta^{avg}_{loss} \approx 1.8\%$. The loss is small, which indicates that the DLBMPC achieved good performance in the closed-loop implementation. The DNN is able to be a good approximation of the MPC law.

The comparison of average computation time for MPC and DLBMPC is shown in Figure 5.10. It is observed that to obtain a solution, DLBMPC takes significantly less computation time than MPC. The average solution time of using MPC is approximately 0.16 CPU seconds, while using DLBMPC only takes approximately 0.05 CPU seconds on average. DLBMPC saves 68.8% computation time. In the context of supply chain optimization, a solution time of 0.16 seconds may be not long. However, it should be noted that in this work, a deterministic MPC formulation is investigated, and solving it is not time-consuming. In case where more sophisticated MPC problem is formulated to obtain better control performance, e.g., robust MPC strategies (Mastragostino et al., 2014), the solution time could increase by a large amount. By contrast, the DLBMPC only requires the evaluation of a DNN to obtain decisions, thus it would still be fast in the real-time implementation.

Figure 5.8. Case study II: closed-loop simulation results for a test scenario.

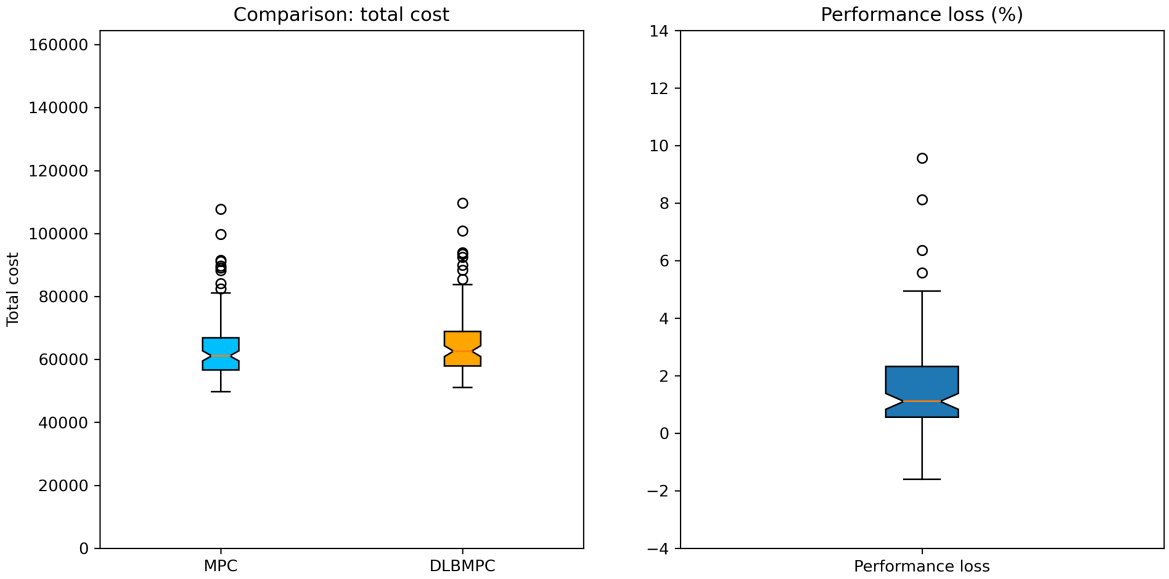


Figure 5.9. Case study II: comparison of closed-loop performance.

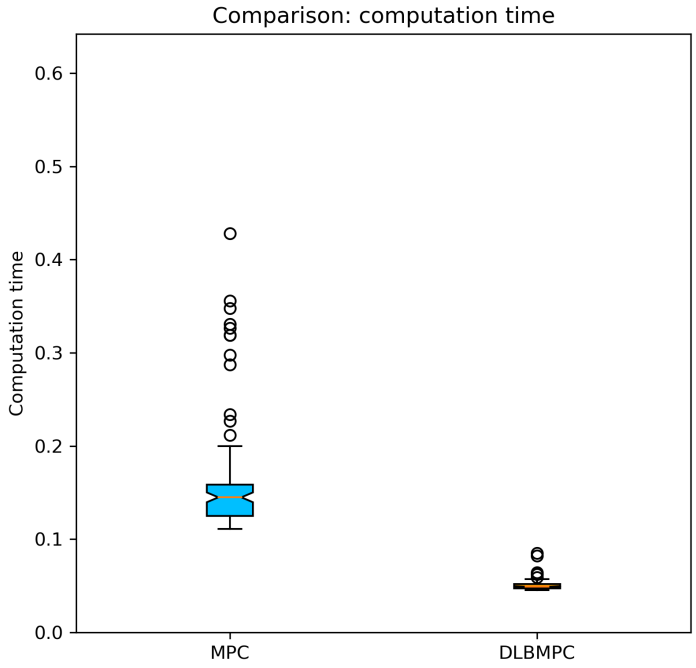


Figure 5.10. Case study II: comparison of computation time.

5.4 Conclusion

This work investigates the performance of deep learning-based MPC in the context of supply chain optimization. An approach is proposed to implement the method when there are delayed decisions in the system. Moreover, a heuristic method is proposed to deal with the binary decision variables for deep learning-based mixed-integer MPC. For data generation, a combination of closed-loop simulation samples and random samples are generated to train the DNN. The DLBMPC is validated on two case studies through closed-loop simulation. The first case study involves a linear MPC, and the second case study involves a more complicated mixed-integer linear MPC. Results show that DLBMPC has good closed-loop performance and achieves high accuracy in approximating the MPC decisions. The average performance loss in the two cases are 0.43% and 1.8%, respectively. Compared with MPC, DLBMPC reduces the online computation time in the second case study.

A future research direction is deep learning-based robust MPC of supply chain systems. Deep learning-based robust MPC approximates the optimal solution of a robust MPC problem by using a DNN (Karg and Lucia, 2021). This technique helps address real-time supply chain decision-making under uncertainty. Another potential direction is deep learning-based Pareto solution of multi-objective MPC. As shown by Bemporad and Muñoz de la Peña (2009), the Pareto optimal solution of a multi-parametric multi-objective linear or quadratic MPC can be computed as a PWA function of the state vector and the weights assigned to the objective functions. Therefore, it is worth investigating using a DNN to approximate the Pareto optimal solution of multi-objective MPC. This technique would help improve the computational efficiency of real-time multi-objective supply chain optimization.

Nomenclature

Indices/Sets

t	time period
j	material
$s \in S_S$	supplier
$p \in S_P$	plant site
$dc \in S_{DC}$	distribution site
$ps \in S_{PS}^p$	production scheme at plant site p
S_R^p	set of raw material at plant site p
S_I^p	set of intermediate product at plant site p
S_F^p	set of final product at plant site p
$I(j)$	set of production schemes that consumes material j
$O(j)$	set of production schemes that produces material j

Binary variables

$u_{ps,t}^p$	binary variable associated with production scheme ps at plant p at time period t; $u_{ps,t}^p = 1$ if ps begins at time period t, and 0 otherwise.

Continuous variables

$B_{j,t}^{dc}$	quantity of back orders of product j at distribution site dc at time period t
$I_{j,t}^p$	inventory of material j at plant p at time period t
$I_{j,t}^{dc}$	stock of product j at distribution site dc at time period t
$IL_{j,t}^{dc}$	inventory level of product j at distribution site dc at time period t
$O_{j,t}^{s,p}$	quantity of raw material j purchased from supplier s by plant site p at time period t
$Q_{j,t}^{p,p'}$	quantity of material j from plant site p to plant site p' at time period t
$Q_{j,t}^{p,dc}$	quantity of material j from plant site p to distribution site dc at time period t
$SA_{j,t}^{dc}$	sale of product j at distribution site dc at time period t
$W_{ps,t}^p$	quantity of main raw material associated with production scheme ps at plant site p at time period t

Parameters

N	planning horizon
α_j^s	unit purchase cost of material j at supplier s
β_{ps}^p	process yield parameter of product in production scheme ps at plant site p
$\delta^{p,dc}$	transportation delay from plant site p to distribution site dc
$\delta^{p,p'}$	transportation delay from plant site p to plant site p'
δ_{ps}^p	production delay of production scheme ps at plant site p
$\mu_{ps,j}^p$	mass balance coefficient of material j in production scheme ps at plant site p
π_j^p	unit inventory cost of material j at plant site p
π_j^{dc}	unit inventory cost of material j at distribution site dc
ρ_{ps}^p	unit production cost of production scheme ps at plant site p
$\tau^{p,dc}$	unit shipping cost from plant site p to distribution site dc
$\tau^{p',p}$	unit shipping cost from plant site p' to plant site p
$D_{j,t}^{dc}$	demand of final product j distribution site dc at time period t
$I_R^{p,ub}$	maximum storage capacity of raw materials at plant site p
$I_I^{p,ub}$	maximum storage capacity of intermediate products at plant site p
$I_j^{p,ub}$	maximum storage capacity of final product j at plant site p
$I_j^{dc,ub}$	maximum storage capacity of final product j at distribution site dc
$Q^{s,p,ub}$	maximum transportation capacity from supplier s to plant site p
$Q^{p,dc,ub}$	maximum transportation capacity from plant site p to distribution site dc
$Q^{p,p',ub}$	maximum transportation capacity from plant site p to plant site p'
$W_{ps}^{p,lb}$	minimum batch size for production scheme ps at plant site p
$W_{ps}^{p,ub}$	maximum batch size for production scheme ps at plant site p

References

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

Abiodun, O. I., A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad (2018). "State-of-the-art in artificial neural network applications: A survey". In: *Heliyon* 4.11, e00938.

Adhau, S., V. V. Naik, and S. Skogestad (2021). "Constrained neural networks for approximate nonlinear model predictive control". In: *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 295–300.

Ahmed, M. S. and M. A. Al-Dajani (1998). "Neural regulator design". In: *Neural Networks* 11.9, pp. 1695–1709.

Åkesson, B. M. and H. T. Toivonen (2006). "A neural network model predictive controller". In: *Journal of Process Control* 16.9, pp. 937–946.

Åkesson, B. M., H. T. Toivonen, J. B. Waller, and R. H. Nyström (2005). "Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process". In: *Computers & Chemical Engineering* 29.2, pp. 323–335.

Axsäter, S. (2015). *Inventory Control*. 3rd. Springer, Cham.

Bemporad, A., M. Morari, V. Dua, and E. N. Pistikopoulos (2002). "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1, pp. 3–20.

Bemporad, A. and D. Muñoz de la Peña (2009). "Multiobjective model predictive control". In: *Automatica* 45.12, pp. 2823–2830.

Bieker, K., S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz (2020). "Deep model predictive flow control with limited sensor data and online learning". In: *Theoretical and Computational Fluid Dynamics* 34, pp. 577–591.

Bonassi, F., M. Farina, J. Xie, and R. Scattolini (2022). "On Recurrent Neural Networks for learning-based control: Recent results and ideas for future developments". In: *Journal of Process Control* 114, pp. 92–104.

Bonzanini, A. D., J. A. Paulson, G. Makrygiorgos, and A. Mesbah (2021). "Fast approximate learning-based multistage nonlinear model predictive control using Gaussian processes and deep neural networks". In: *Computers & Chemical Engineering* 145, p. 107174.

Bynum, M. L., G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff (2021). *Pyomo – Optimization Modeling in Python.* Third. Vol. 67. Springer Science & Business Media.

Cao, Y. and R. B. Gopaluni (2020). "Deep neural network approximation of nonlinear model predictive control". In: *IFAC-PapersOnLine* 53.2, pp. 11319–11324.

Cavagnari, L., L. Magni, and R. Scattolini (1999). "Neural network implementation of nonlinear receding-horizon control". In: *Neural Computing & Applications* 8.1, pp. 86–92.

Chang-Yun Seong and B. Widrow (2001a). "Neural dynamic optimization for control systems. I. Background". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.4, pp. 482–489.

— (2001b). "Neural dynamic optimization for control systems. II. Theory". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.4, pp. 490–501.

— (2001c). "Neural dynamic optimization for control systems. III. Applications". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.4, pp. 502–513.

Chen, F. and R. Samroengraja (2000). "The stationary beer game". In: *Production and Operations Management* 9.1, pp. 19–30.

Chen, S., K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari (2018). "Approximating explicit model predictive control using constrained neural networks". In: *2018 Annual American Control Conference (ACC)*, pp. 1520–1527.

Chen, S. W., T. Wang, N. Atanasov, V. Kumar, and M. Morari (2022). "Large scale model predictive control with neural networks and primal active sets". In: *Automatica* 135, p. 109947.

Csekő, L. H., M. Kvasnica, and B. Lantos (2015). "Explicit MPC-based RBF neural network controller design with discrete-time actual Kalman filter for semiactive suspension". In: *IEEE Transactions on Control Systems Technology* 23.5, pp. 1736–1753.

Deepa, S. and I. Baranilingesan (2018). "Optimized deep learning neural network predictive controller for continuous stirred tank reactor". In: *Computers & Electrical Engineering* 71, pp. 782–797.

Drgoňa, J., D. Picard, M. Kvasnica, and L. Helsen (2018). "Approximate model predictive building control via machine learning". In: *Applied Energy* 218, pp. 199–216.

Faísca, N. P., V. Dua, and E. N. Pistikopoulos (2007). "Multiparametric linear and quadratic programming". In: *Multi-Parametric Programming*. Ed. by E. N. Pistikopoulos, M. C. Georgiadis, and V. Dua. Weinheim: WILEY-VCH Verlag GmbH & Co. KGaA, pp. 3–23.

Frigge, M., D. C. Hoaglin, and B. Iglewicz (1989). "Some implementations of the boxplot". In: *The American Statistician* 43.1, pp. 50–54.

Furuta, D., K. Kutsuzawa, T. Okamoto, S. Sakaino, and T. Tsuji (2017). "Model predictive control based deep neural network for dynamic manipulation". In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 5215–5220.

Grossmann, I. E., R. M. Apap, B. A. Calfa, P. García-Herreros, and Q. Zhang (2016). "Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty". In: *Computers & Chemical Engineering* 91, pp. 3–14.

Hart, W. E., J.-P. Watson, and D. L. Woodruff (2011). "Pyomo: modeling and solving mathematical programs in Python". In: *Mathematical Programming Computation* 3.3, pp. 219–260.

Hassanpour, H., B. Corbett, and P. Mhaskar (2022). "Artificial neural network-based model predictive control using correlated data". In: *Industrial & Engineering Chemistry Research* 61.8, pp. 3075–3090.

Hertneck, M., J. Köhler, S. Trimpe, and F. Allgöwer (2018). "Learning an approximate model predictive controller with guarantees". In: *IEEE Control Systems Letters* 2.3, pp. 543–548.

Hirose, N., R. Tajima, and K. Sukigara (2018). "MPC policy learning using DNN for human following control without collision". In: *Advanced Robotics* 32.3, pp. 148–159.

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366.

Karg, B. and S. Lucia (2018). "Deep learning-based embedded mixed-integer model predictive control". In: *2018 European Control Conference (ECC)*, pp. 2075–2080.

Karg, B. and S. Lucia (2019). "Learning-based approximation of robust nonlinear predictive control with state estimation applied to a towing kite". In: *2019 18th European Control Conference (ECC)*, pp. 16–22.

Karg, B., T. Alamo, and S. Lucia (2021). "Probabilistic performance validation of deep learning-based robust NMPC controllers". In: *International Journal of Robust and Nonlinear Control* 31.18, pp. 8855–8876.

Karg, B. and S. Lucia (2020). "Efficient representation and approximation of model predictive control laws via deep learning". In: *IEEE Transactions on Cybernetics* 50.9, pp. 3866–3878.

— (2021). "Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning". In: *Computers & Chemical Engineering* 148, p. 107266.

Krishnamoorthy, D., A. Mesbah, and J. A. Paulson (2021). "An adaptive correction scheme for offset-free asymptotic performance in deep learning-based economic MPC". In: *IFAC-PapersOnLine* 54.3, pp. 584–589.

Krishnamoorthy, D. and S. Skogestad (2019). "Real-time optimization strategies using surrogate optimizers". In: *Proceedings of the 2019 Foundations in Process Analytics and Machine Learning*.

Kumar, P., J. B. Rawlings, and S. J. Wright (2021). "Industrial, large-scale model predictive control with structured neural networks". In: *Computers & Chemical Engineering* 150, p. 107291.

Li, X. and T. E. Marlin (2009). "Robust supply chain performance via Model Predictive Control". In: *Computers & Chemical Engineering* 33.12, pp. 2134–2143.

Lopez-Martin, M., B. Carro, A. Sanchez-Esguevillas, and J. Lloret (2019). "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks". In: *Expert Systems with Applications* 124, pp. 196–208.

Lucia, S. and B. Karg (2018). "A deep learning-based approach to robust nonlinear model predictive control". In: *IFAC-PapersOnLine* 51.20. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018, pp. 511–516.

Lucia, S., D. Navarro, B. Karg, H. Sarnago, and Ó. Lucía (2021). "Deep learning-based model predictive control for resonant power converters". In: *IEEE Transactions on Industrial Informatics* 17.1, pp. 409–420.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Masti, D. and A. Bemporad (2019). "Learning binary warm starts for multiparametric mixed-integer quadratic programming". In: *2019 18th European Control Conference (ECC)*, pp. 1494–1499.

Masti, D., T. Pippia, A. Bemporad, and B. D. Schutter (2020). "Learning approximate semi-explicit hybrid MPC with an application to microgrids". In: *IFAC-PapersOnLine* 53.2, pp. 5207–5212.

Mastragostino, R., S. Patel, and C. L. Swartz (2014). "Robust decision making for hybrid process supply chain systems via model predictive control". In: *Computers & Chemical Engineering* 62, pp. 37–55.

Mestan, E., M. Türkay, and Y. Arkun (2006). "Optimization of operations in supply chain systems using hybrid systems approach and model predictive control". In: *Industrial & Engineering Chemistry Research* 45.19, pp. 6493–6503.

Nayeri, M. D., A. Alasty, and K. Daneshjou (2004). "Neural optimal control of flexible spacecraft slew maneuver". In: *Acta Astronautica* 55.10, pp. 817–827.

Nubert, J., J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe (2020). "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control". In: *IEEE Robotics and Automation Letters* 5.2, pp. 3050–3057.

Oroojlooyjadid, A., M. Nazari, L. V. Snyder, and M. Takáč (2022). "A deep Q-network for the beer game: Deep reinforcement learning for inventory optimization". In: *Manufacturing & Service Operations Management* 24.1, pp. 285–304.

Ortega, J. G. and E. Camacho (1996). "Mobile robot navigation in a partially structured static environment, using neural predictive control". In: *Control Engineering Practice* 4.12, pp. 1669–1679.

Papageorgiou, L. G. (2009). "Supply chain optimisation for the process industries: Advances and opportunities". In: *Computers & Chemical Engineering* 33.12, pp. 1931–1938.

Parisini, T. and R. Zoppoli (1995). "A receding-horizon regulator for nonlinear systems and a neural approximation". In: *Automatica* 31.10, pp. 1443–1451.

Paulson, J. A. and A. Mesbah (2020). "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction". In: *IEEE Control Systems Letters* 4.3, pp. 719–724.

Perea-López, E., B. Ydstie, and I. E. Grossmann (2003). "A model predictive control strategy for supply chain optimization". In: *Computers & Chemical Engineering* 27.8, pp. 1201–1218.

Pin, G., M. Filippo, F. Pellegrino, G. Fenu, and T. Parisini (2013). "Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and offline design". In: *International Journal of Control* 86.5, pp. 804–820.

Pon Kumar, S. S., A. Tulsyan, B. Gopaluni, and P. Loewen (2018). "A deep learning architecture for predictive control". In: *IFAC-PapersOnLine* 51.18, pp. 512–517.

Punjani, A. and P. Abbeel (2015). "Deep learning helicopter dynamics models". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3223–3230.

Qin, S. J. (2003). "Statistical process monitoring: basics and beyond". In: *Journal of Chemometrics* 17.8-9, pp. 480–502.

Saikia, P., R. D. Baruah, S. K. Singh, and P. K. Chaudhuri (2020). "Artificial Neural Networks in the domain of reservoir characterization: A review from shallow to deep models". In: *Computers & Geosciences* 135, p. 104357.

Schildbach, G. and M. Morari (2016). "Scenario-based model predictive control for multi-echelon supply chain management". In: *European Journal of Operational Research* 252.2, pp. 540–549.

Schulz, E., M. Diaz, and J. Bandoni (2005). "Supply chain optimization of large-scale continuous processes". In: *Computers & Chemical Engineering* 29.6, pp. 1305–1316.

Srinivasan, M., A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. D. Cairano (2021). "Fast multi-robot motion planning via imitation learning of mixed-integer programs". In: *IFAC-PapersOnLine* 54.20, pp. 598–604.

Sterman, J. D. (1989). "Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment". In: *Management Science* 35.3, pp. 321–339.

Subramanian, K., C. T. Maravelias, and J. B. Rawlings (2012). "A state-space model for chemical production scheduling". In: *Computers & Chemical Engineering* 47, pp. 97–110.

Subramanian, K., J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan (2013). "Integration of control theory and scheduling methods for supply chain management". In: *Computers & Chemical Engineering* 51, pp. 4–20.

Torres, J. J., C. Li, R. M. Apap, and I. E. Grossmann (2022). "A review on the performance of linear and mixed integer two-stage stochastic programming software". In: *Algorithms* 15.4.

Wu, Q. H., B. W. Hogg, and G. W. Irwin (1992). "A neural network regulator for turbogenerators". In: *IEEE Transactions on Neural Networks* 3.1, pp. 95–100.

Wu, Z., D. Rincon, and P. D. Christofides (2020). "Real-time adaptive machine-learning-based predictive control of nonlinear processes". In: *Industrial & Engineering Chemistry Research* 59.6, pp. 2275–2290.

Zhang, X., M. Bujarbaruah, and F. Borrelli (2021). "Near-optimal rapid MPC using neural networks: A primal-dual policy learning framework". In: *IEEE Transactions on Control Systems Technology* 29.5, pp. 2102–2114.

Chapter 6

Conclusions and Future Research Directions

This chapter presents the conclusions of this work and the future research directions.

6.1 Conclusions

This thesis investigates the use of data-driven methods to improve supply chain management and optimization, specifically, to address operational supply chain monitoring and control problems.

In Chapter 2, a data-driven SCMo method based on PCA is proposed for the fault detection

and diagnosis of supply chain systems. In order to build the SCMo model, supply chain data such as inventory levels, demands, and material in transit are collected. PCA is employed to model the NOC of the supply chain. Two monitoring statistics, the Hotelling's T^2 and SPE, are used to detect abnormal behaviour of the supply chain. Contribution plots are adopted to identify the fault-related variables when abnormality is detected. A Python-based supply chain simulator is developed, and the proposed SCMo method is validated on two simulation case studies. The results show that dynamic PCA is able to detect and diagnose the abnormal behaviour of the supply chain, such as transportation delay, low production rate and supply shortage. Moreover, the SPE by dynamic PCA is more reliable than the other monitoring statistics investigated in this chapter, while the sensitivity of PCA is lower.

In Chapter 3, we further extended the data-driven SCMo framework by using the CVA method. A state-space model of supply chain is developed, such that the CVA-based SCMo method can be implemented. This method is validated on two simulation case studies. Results show that CVA is able to detect different types of operational supply chain faults, and the CVA-based contribution plot is able to identify the fault-related variables and thus indicating the cause of the fault. Moreover, it is observed that for both CVA- and PCA-based SCMo, the monitoring statistics in the residual space are more reliable. A comparative study shows that CVA identifies a smaller system order than PCA and achieves comparable performance to PCA in a lower-dimensional space. This study indicates the advantage of applying state-space models to supply chain systems.

In Chapter 4, we developed a risk-averse supply chain control method using safe RL. Two safe RL algorithms, $\hat{Q}$-learning and β-pessimistic Q-learning, are examined and compared with the classical OUT policy and Q-learning. Compared with Q-learning which maximizes the expected return, the safe RL algorithms incorporates the worst-case return in the optimization criterion. A Python-based supply chain simulator is developed to conduct case studies. Results show that in a normal environment, Q-learning has the best performance in terms of the total cost. However, when there is an unexpected change in the environment, $\hat{Q}$-learning and β-pessimistic Q-learning with appropriate β perform better,

and thus they are more robust to uncertainty. Moreover, we extend the use of RL-based control method to CLSCs. It is found that RL polices can reduce the bullwhip effect in a CLSC.

In Chapter 5, a deep learning-based MPC method for real-time supply chain optimization is developed. We developed an approach to implementing the method in the case where there are delayed decisions in the system, and a heuristic method to deal with the binary decision variables for deep learning-based mixed-integer MPC. The deep learning-based MPC is validated on two case studies through closed-loop simulation. Results show that this technique has good closed-loop performance with a small average performance loss, and achieves high accuracy in approximating the MPC decisions. Compared with MPC, this technique can reduce the online computation time.

6.2 Future Research Directions

Based on a survey of the current literature, several research gaps are identified and the following potential research directions are proposed for further exploration.

The first research avenue that is worth further exploration is data-driven SCMo based on different SPM methods, besides PCA and CVA. In this work, although we focused on the use of PCA and CVA, this implies that other SPM methods may also be applied to supply chain systems. The area of process monitoring has been well-established, and various data-driven methods have been developed for industrial process monitoring, for example, PLS, Fisher discriminant analysis (FDA), independent component analysis (ICA) (Yin et al., 2014; Severson et al., 2016; Ge, 2017). However, most of them have not explored in the context of SCMo. It would be beneficial to SCMo to explore the use of other data-driven methods.

The second potential research direction is safe RL with function approximation. In the current literature, function approximation has been mainly investigated for conventional RL, for example, the *deep RL* method (Hubbs et al., 2020). It would help extend the scope of RL to investigate safe RL with function approximation, e.g., deep RL with a risk-averse optimization criterion. This method can be named *deep safe RL*. It is applicable to supply

chain control problems with continuous state and action spaces.

For the research avenue of deep learning-based supply chain MPC, it would help bridge the research gaps to explore deep learning-based robust MPC for supply chain control. This technique uses a DNN to approximate the optimal solution of a robust MPC problem (Karg and Lucia, 2021). It can be applied to deal with real-time supply chain decision-making under uncertainty. Another potential direction is deep learning-based Pareto solution of multi-objective MPC. Bemporad and Muñoz de la Peña (2009) showed that for a multi-parametric multi-objective linear or quadratic MPC addressed by the weighting method, the Pareto optimal solution can be computed as a piecewise affine function of the state vector and the weights assigned to the objectives. Therefore, it is possible to approximate the Pareto optimal solution of multi-objective MPC by using a DNN. This technique would help improve the computational efficiency for multi-objective supply chain optimization in real time.

References

Bemporad, A. and D. Muñoz de la Peña (2009). "Multiobjective model predictive control". In: *Automatica* 45.12, pp. 2823–2830.

Ge, Z. (2017). "Review on data-driven modeling and monitoring for plant-wide industrial processes". In: *Chemometrics and Intelligent Laboratory Systems* 171, pp. 16–25.

Hubbs, C. D., C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick (2020). "A deep reinforcement learning approach for chemical production scheduling". In: *Computers & Chemical Engineering* 141, p. 106982.

Karg, B. and S. Lucia (2021). "Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning". In: *Computers & Chemical Engineering* 148, p. 107266.

Severson, K., P. Chaiwatanodom, and R. D. Braatz (2016). "Perspectives on process monitoring of industrial systems". In: *Annual Reviews in Control* 42, pp. 190–200.

Yin, S., S. X. Ding, X. Xie, and H. Luo (2014). "A review on basic data-driven approaches for industrial process monitoring". In: *IEEE Transactions on Industrial Electronics* 61.11, pp. 6418–6428.