Doctoral Dissertation
Doctoral Program in Italian National PhD in Artificial Intelligence for Industry 4.0
(37^{th} cycle)

Advancing Supply Chain Inventory Management through Mathematical and Deep Reinforcement Learning Approaches

Francesco Stranieri
* * * * *

Supervisors
Prof. Fabio Stella, Supervisor
Prof. Chaaben Kouki, Co-supervisor

Doctoral Examination Committee:
Prof. Paolo Brandimarte, President, Politecnico di Torino
Prof. Mohamed Zied Babai, Referee, KEDGE Business School
Prof. Robert Boute, Referee, KU Leuven
Prof. Camélia Dadouchi, Member, Polytechnique Montréal
Prof. Mirco Peron, Member, NEOMA Business School

Politecnico di Torino
2025

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Francesco Stranieri

Turin, 2025

Summary

The increasing complexity of modern supply chains, driven by industry 4.0 principles and advances in artificial intelligence (AI), has created a growing need for adaptive, data-driven supply chain management strategies. While simple and widely recognized, traditional inventory policies often struggle to address the multidimensional, uncertain, and dynamic challenges inherent in supply chain inventory management (SCIM). In this context, deep reinforcement learning (DRL) has emerged as a promising alternative capable of directly learning effective policies by interacting with simulated supply chain environments. However, its practical adoption remains constrained by several barriers, including computational complexity, scalability issues, limited real-world data, and interpretability concerns.

This thesis explores the potential of DRL to improve SCIM across diverse structural attributes and operational factors, focusing on three primary research objectives: 1) evaluating the performance of state-of-the-art DRL algorithms in two-echelon systems subjected to stochastic and seasonal demand, varying numbers of product types and warehouses, and different lead times, thus addressing scalability issues; 2) developing a hybrid heuristic that combines DRL with multi-stage (MS) stochastic programming to mitigate the computational complexity associated with the curse of dimensionality, while accounting for production limits; and 3) assessing the applicability and interpretability of DRL in a real-world pharmaceutical supply chain characterized by perishability, production yields, batch limits, lead times, and lost sales under non-stationary demand.

The first study demonstrates that DRL, particularly the proximal policy optimization (PPO) algorithm, consistently outperforms base-stock and (s, Q)-policies in capacitated two-echelon systems. By leveraging continuous action spaces and designing a balanced allocation rule, PPO effectively handles increasing complexity, scales across large state-action spaces, and achieves training times that are comparable to traditional inventory policies. The second study introduces a hybrid heuristic in which DRL determines long-horizon production planning while MS stochastic programming optimizes short-term logistics decisions. This combined approach results in robust, computationally tractable solutions that outperform both standalone methods. The third study focuses on a pharmaceutical supply chain, where we evaluate PPO with the order-up-to policy and one of its variants based on projected inventory levels, for which we derive and validate

bounds-based procedures to optimize their parameters. All three policies surpass human-driven policies in minimizing costs under complex conditions, providing managerial insights, and interpreting the impact of costs and lost sales in relation to ethical and legal constraints.

These findings collectively highlight the transformative potential of DRL, demonstrating its ability to tackle uncertain and dynamic SCIM problems, exploit high-dimensional state and action spaces, and outperform traditional inventory policies while also extending and identifying best practices for integrating DRL into practical SCIM decision-making. Furthermore, the open-source SCIMAI-Gym Python library, developed as part of this research, provides a flexible simulation environment that encourages further experimentation, collaboration, and practical implementation.

In conclusion, this thesis lays a solid foundation for future research. Potential directions include implementing action masking and reward shaping techniques, integrating large language models, leveraging advanced computational frameworks, and modeling even more complex supply chain configurations. Ultimately, these contributions advance the field toward more robust, efficient, and adaptive supply chains that align with the principles of AI and industry 4.0.

Acknowledgements

I would like to thank my supervisor, Prof. Stella, for accompanying me throughout this academic journey – from my Bachelor's to my Master's, and finally to the PhD – and for allowing me to conduct my research with complete freedom and autonomy. His constant guidance and unfailing support have been essential and have also helped me to see life with a little more positivity.

I would also like to thank my co-supervisor, Prof. Kouki, for his continuous availability, for his teachings on supply chain management, and for giving me the opportunity to discover and participate in some of the most important conferences in the field – from the ISIR Summer School in Cardiff to the POMS Conference in Paris and the ISIR Symposium in Budapest – as well as for hosting me in Angers during one of the most challenging, yet personally enriching, experiences of my life.

Both of my supervisors have taught me that hard work always pays off, sooner or later, and the very fact that I am here defending this PhD thesis is the clearest proof.

I would like to express my sincere gratitude to the members of the examination committee – Profs. Brandimarte, Babai, Boute, Dadouchi, and Peron – for their kind availability and for agreeing to be part of the committee for the defense of my doctoral dissertation. It is indeed both a pleasure and an honor for me to have the opportunity to discuss my research with such distinguished academics, whose papers I have studied and whose seminars I have attended as part of my professional growth. In particular, I would like to thank the reviewers for the time and effort they dedicated to evaluating my thesis.

I would also like to thank the University of Milano-Bicocca, together with the MUR, for funding my PhD scholarship, and the Rotary Clubs – with all the members I had the pleasure of meeting – along with Ms. Maria Montorfano, for awarding me the prestigious Rotary International "Dr. Gabriele Corbelli" scholarship, which gave me the opportunity to conduct a transformative and impactful research period abroad.

I would like to thank Enrico from Bristol Myers Squibb for the valuable collaboration we had the opportunity to establish, which gave me the chance to validate my academic research on a real industrial use case. His guidance has been extremely helpful and has significantly contributed to my growth as a researcher and as a professional.

I am grateful to the ISIR community and all of its members – in particular Prof. Syntetos, for making me feel included since my first summer school outside Italy – and

to all the special people I had the honor of meeting through the formative experiences made possible by this community. I am also deeply grateful for all the associations I had the privilege of being part of – including IEEE, AIxIA, AIRO, and ADI – and to all the professors, speakers, and organizers of the PhD courses, summer schools, workshops, symposia, and conferences I attended during my doctoral studies for expanding my horizons and enriching my research journey.

Still from a professional perspective, I am profoundly thankful to all of my co-authors, without whom I would never have been able to develop the work we did and achieve such quality, which allowed us to publish in prestigious journals and conferences. In particular, I would like to thank Prof. Fadda for always being available whenever I needed advice, and my colleague Alberto, with whom I have shared many memorable experiences – such as the AIxIA Conference in Rome. Behind each of these collaborations are genuinely inspiring people, whom I am glad to consider not only colleagues but also friends. The same goes for the students I had the pleasure of supervising over the years – such as Giovanni and Eleonora – to whom I hope I succeeded in teaching something meaningful and useful for life beyond purely theoretical concepts.

I would also like to thank all the professors and colleagues I have interacted with over the years at the university, who have made this journey significantly more memorable. Starting with the professors at Bicocca – such as Profs. Viviani, Sartori, Savi, and Dennunzio – those at Politecnico di Torino, like Prof. Di Carlo and Alessandra Calosso, and those I met at ESSCA, such as Prof. Cersosimo, as well as all my colleagues from both my Bachelor's and Master's degrees, with whom I am still in touch – in particular Davide, Francesco, Stefano, Ali, Simone, Alessandro, Andrea, and Jacopo. And, of course, my PhD colleagues – including Riccardo, for his hospitality every time I visit Bicocca – and especially those from the PhD-AI program – Leo, Mirko, Antonino, and Luca, with whom attending the Fall School and spending time in Turin every year, sharing hotel rooms and meals, has always been great fun.

Furthermore, I would like to thank all those who, over the years, have believed in me and supported my projects and ideas – in particular Arianna and Sofia, as well as all the artists, institutions, and sponsors who supported, participated in, and visited the art exhibition I conceived on the topic of genAI – those who have entrusted me with important roles, whether as a professor, tutor, reviewer, or speaker, and those who have recognized and rewarded my work.

I would also like to acknowledge those who, intentionally or not, tried to put obstacles in my path, exploit my efforts, or discourage me. In the end, they did not succeed but instead contributed to making me stronger.

On a personal level, it would be impossible not to thank my family. First and foremost, my parents – my father, Luigi, for introducing me to the world of computer science from the very first time I tried a computer, at around the age of six during the Formula 1 Grand Prix in Monza (which later became one of my passions), and for supporting this interest by always making sure I had electronic devices to experiment with, and my mother, Simona, for providing me with constant love and never letting me lack support,

even during the darkest moments. Given my path, reaching this point was far from guaranteed, but I am profoundly grateful for always having had the freedom to make mistakes in order to find my own way.

I am also grateful to always be able to count on my brothers – Michele, Gabriele, and Raffaele – my uncles and aunts – Maurizio and Maria Angela, Antonio and Teresa, Daniela and Paolo, and Anna and Carmelo – my cousins Claudia (aka Bonnie), Danilo, Marco, Simone, Alessandro, Andrea, and Davide, the little ones Mattia, Sofia, Carlotta, and Elisa, as well as my grandparents Pasqua and Michele, and my extended family, including Rebecca, Doris, and Vincenzo, together with Lilli and Chicca.

Many of them may not fully understand what a PhD is or what I have been working on over these past years, but they have always made me feel proud and supported.

A heartfelt thank you goes to all the family on the side of my mother, Simona – my grandparents Giulio and Francesca, and my uncles and aunts Maria, Giovanna, and Giancarlo. Hand in hand with my mother, they have all been authentic role models and a source of inspiration for the person I hope to become.

I would also like to thank my friends – lifelong ones from the neighborhood and school, and those I have met more recently – including Davide S., Alessandro, frtm Arthur, Oury, Giovanni from Keti Perdy, Michele, Andrea N., Mirco, Manuel, Luigi, Stefano P., Eleonora, Federico (with his daily dose of reels), Andrea V., Matteo, Davide G., Nicolò, Giuseppe, Luisa, Jacopo, Carlo, Stefano O., Miriam, and Angela. I am also grateful to my friends in Turin – especially my bro Antonio, Vito, and Roberta – and to all those I met through the amazing student association IEEE-HKN, as well as the friends from Civico 144, including Aldo, Giulia, Andrea, Alessandro (the Maestro), and Laura, and to those from the U.C. 19 group and from Bax and his bar – especially Christian and his family, Tania, Silvana, Fabrizio, Emanuele, Simone, Fabiana and Daniela with Sofia and Rachele, Stefano, Salvatore, Alessandro, the two Bennys, Michael, Francesco, and Steve – and many others whose names I cannot list here without making this too long. The experiences I have shared with my friends have helped me grow as a person, and I will never be able to thank them enough for all the moments we have lived together.

A special thought also goes to the professors of the evening classes at Hensemberger, who reignited my passion for learning, as well as to my favorite teachers, Bianca and Daniela.

A special mention also goes to my colleagues at Technoprobe – from the machine learning, data engineering, robotic process automation, and metrology teams – for the warm welcome they gave me and for their support during these months in which I was working on my thesis at the same time – Marco M., Daniele, Camilla, Mohammed (my desk mate on the left, known as Mao), Satwinder (my desk mate on the right and partner in fun, known as Sat), Diego, Andrea (Tamixxx), Tommaso, Vincent (the top player), Roberta, Lucia, Marco P., Matteo (with Waze), Nicola, Beatrice, Riccardo, Mehdi, Giulia, Krizia, and the whole team.

A thought also goes to all the people who have been part of my path, and to all those who, even if they are no longer with me physically, are still present in spirit. Thank you.

Last but not least, I would like to sincerely thank my wonderful girlfriend, Marta. She has shown me what real love is, supporting me through every moment – even the hardest ones – by being a constant point of reference, giving me strength and inspiration, and understanding, accepting, and encouraging me to push beyond every obstacle. None of this is simple or can be taken for granted, and I thank her for always being by my side. The moments together and the shared adventures we have experienced have been nothing short of magical, and I could not have entrusted the key to my heart to anyone else. I would also like to thank her family – Elena, Vito, Federico, Newton, Silvia, Amos, Ermelina, and Piero – for making me feel at home from the very first day.

And, of course, I want to thank myself – for never giving up, for always believing, and for coming this far.

With the completion of my PhD, an important chapter of my life comes to a close. Beyond my professional achievements, I am deeply grateful for all the people I have met along the way and who are still by my side. I am now curious to see where my next steps will bring me. Audentes fortuna iuvat. Always.

To my mom, my warrior.
I love you always forever!

Contents

List of Tables xiii

List of Figures xv

1 Introduction 1
- 1.1 Background and Motivation . . . 1
  - 1.1.1 Artificial Intelligence and Industry 4.0 in Supply Chains . . . 1
  - 1.1.2 Supply Chain Management and Inventory Management . . . 2
  - 1.1.3 Advancements through Deep Reinforcement Learning . . . 3
  - 1.1.4 Challenges in Supply Chain Inventory Management . . . 4
  - 1.1.5 Evolution of Inventory Policies . . . 5
  - 1.1.6 Barriers to Implementing Deep Reinforcement Learning in Supply Chain Inventory Management . . . 7
  - 1.1.7 Opportunities for Advanced Solutions . . . 8
- 1.2 Research Questions . . . 9
  - 1.2.1 SCIMAI-Gym Python Library . . . 11

2 Performance of Deep Reinforcement Learning Algorithms in Two-Echelon Inventory Control Systems 17
- Abstract . . . 18
- 2.1 Introduction . . . 19
- 2.2 Literature Review . . . 20
- 2.3 Problem Definition and Modelling . . . 25
  - 2.3.1 Markov Decision Process Formulation . . . 27
  - 2.3.2 Deep Reinforcement Learning Formulation . . . 32
- 2.4 Numerical Experiments . . . 35
  - 2.4.1 Environment Parameters . . . 36
  - 2.4.2 Implemented Algorithms . . . 37
  - 2.4.3 Results . . . 38
- 2.5 Discussion and Insights . . . 40
  - 2.5.1 Conclusion and Future Research . . . 41

3 Combining Deep Reinforcement Learning and Multi-Stage Stochastic Programming to address the Supply Chain Inventory Management problem **49**
Abstract . . . 50
3.1 Introduction . . . 51
3.2 Literature Review . . . 52
3.3 Problem Formulation . . . 54
3.4 Solution Methods . . . 57
3.4.1 Deep Reinforcement Learning . . . 57
3.4.2 Multi-Stage Stochastic Programming . . . 59
3.4.3 Deep Reinforcement Learning-Based Decomposition . . . 61
3.5 Numerical Experiments . . . 63
3.5.1 Small Settings . . . 65
3.5.2 Large Settings . . . 69
3.5.3 Sensitivity Analysis . . . 72
3.6 Conclusions . . . 74

4 Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity **81**
Abstract . . . 82
4.1 Introduction . . . 83
4.2 Related Work . . . 85
4.2.1 Classical Policies . . . 85
4.2.2 Deep Reinforcement Learning . . . 86
4.3 Case Study Description . . . 87
4.3.1 Order of Events . . . 89
4.3.2 Cost Transformation . . . 90
4.3.3 Dynamic Programming Formulation . . . 91
4.4 Inventory Policies . . . 92
4.4.1 OUT Policy . . . 92
4.4.2 BMS Baseline Policy . . . 95
4.4.3 PIL Policy . . . 96
4.4.4 PPO Algorithm . . . 101
4.5 Numerical Experiments . . . 102
4.5.1 First Scenario . . . 103
4.5.2 Second Scenario . . . 108
4.6 Conclusions . . . 112

5 Conclusions **119**
5.1 Collective Findings . . . 124
5.2 Future Research . . . 125

A Supplementary Material of Chapter 2 **131**
A.1 Hyperparameters Selection 131
A.2 Training Times 134

B Supplementary Material of Chapter 3 **137**
B.1 SCIM Environment Parameters 137
B.2 Hyperparameters Selection 138

C Supplementary Material of Chapter 4 **141**
C.1 Proof of Lemma 1 141
C.2 Proof of Lemma 2 142
C.3 Bounds Value Analysis 142

List of Tables

2.1 The adopted SCIM notation accompanied by a relative explanation (and units of measure). 27

2.2 The environment parameters associated with the experiments conducted. For episode duration, the first value pertains to the experiments with a lead time of one, while the second is for the experiments with lead times of three. Concerning backorder costs, the two values represent experiments with one and two product types, respectively. Finally, for maximum demand value and variation, the first value corresponds to the first product type, while the second value relates to the second product type. 36

3.1 The considered SCIM notation with relative explanation (and units of measure). 56

3.2 Average opt-gap (expressed as a percentage) over 250 episodes with respect to the multi-stage programming model. The standard deviation is provided in round brackets. The lower the value, the better the algorithm. 67

3.3 Average training and testing times for the two experiments conducted under small settings (i.e., with $\epsilon \sim \mathscr{B}(0.5)$ and ϵ as in Equation (3.16), respectively) with their standard deviation (in brackets). 70

3.4 Average EVPI-gap (expressed as a percentage) over 250 episodes concerning the expected value of perfect information model. The standard deviation is provided in round brackets. The lower the value, the better the algorithm. 71

3.5 Average training and testing times for the two experiments conducted under large settings (i.e., with $J = 5$ and $J = 10$ distribution warehouses, respectively) with their standard deviation (in brackets). 73

3.6 Average EVPI gap (as a percentage) by varying demand parameters $P = \{3, 6, 9\}$ and $D = \{1, 2, 3\}$, and considering $J = 5$ distribution warehouses. Adjusting these parameters allows us to alter the number of demand peaks and the balance between deterministic and stochastic demand components. The standard deviation is provided in round brackets. The lower the value, the better the algorithm. 74

4.1 Notation for the supply chain environment with their explanations (and units of measure). 91
4.2 Average total cost for OUT, PIL, PPO, and the human-driven policy over 2000 simulated episodes in the second scenario derived from real-world data, for values of $\hat{z} = \{0\%, 5\%, 10\%\}$ and $b = \{100, 1000, 10000\}$. Lower cost values indicate better performance. 109
4.3 Percentage gap in performance between OUT vs. PPO and PIL vs. PPO for various values of $w = \{3, 6\}$, $m = \{3, 6, 12, 24\}$, and $b = \{10, 50, 100,$
$1000, 10000\}$, at lead times $L = \{3, 6, 12\}$. A positive percentage value indicates better performance by PPO. 111
A.1 The selected hyperparameters for the DRL algorithms. For hyperparameters not included in the table, we used the default values provided by Ray. 132
A.2 The average training times (in minutes) for DRL algorithms and static inventory policies across the two values of lead times considered for one product type (Table A.2a) and two product types (Table A.2b) and for each number of local warehouses. 135
B.1 Parameters defining the SCIM environment for small and large settings. In small settings, when two values are listed, the first is associated with the experiment with $\epsilon \sim \mathcal{B}(0.5)$, while the second refers to ϵ as in Equation (3.16). For the large settings, the provided two values correspond to experiments with $J = 5$ and $J = 10$ distribution warehouses, respectively. 138
B.2 Hyperparameters of the PPO algorithm selected for tuning, along with their corresponding values. Through a grid search, we search for the best combination of hyperparameters for each experiment. 139
C.1 Lower (LB), upper (UB), and optimal (OPT) values for the OUT and PIL policies, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. The results are shown for parameter values $m = \{2, 3, 4\}$, $w = \{2, 4\}$, and $b = \{10, 50, 100, 1000\}$. 143

List of Figures

2.1 Agent-environment interaction in an MDP (taken from Sutton and Barto [1]). 22
2.2 A divergent two-echelon supply chain consisting of a factory and its central warehouse (first echelon) and three local warehouses (second echelon). Shopping carts represent customer demands. 26
2.3 Representation of the dynamics of the inventory control system considered. 31
2.4 Different demands behaviors generated according to Section 2.3.2 fixing $d^1_{max} = 10$, $d^1_{var} = 5$ and $d^2_{max} = 5$, $d^2_{var} = 10$ for distinct typologies and configurations of the SCIM problem. Specifically, a represents one product type and two local warehouses with lead times of three; b shows two product types and one local warehouse with a lead time of one; and c depicts two product types and two local warehouses with lead times of three. 33
2.5 EVPI gap, grouped by experiment, for the experimental setup with one product type. For each number of local warehouses and length of lead times, the figure compares the performance of DRL algorithms, static inventory policies, and EVP. The lower the gap, the better the agent. . 39
2.6 EVPI gap, grouped by experiment, for the experimental setup with two product types. For each number of local warehouses and length of lead times, the figure compares the performance of DRL algorithms, static inventory policies, and EVP. The lower the gap, the better the agent. . 40
3.1 A two-echelon supply chain consisting of a factory (first echelon) and three distribution warehouses (second echelon). 55
3.2 Representation of the dynamics of the SCIM system considered. 57
3.3 Agent-environment interface in an MDP [1]. 58
3.4 Scenario tree representation where each node represents a possible demand realization. 59
3.5 95% confidence interval computed over 250 realizations of the seasonal demand with two distribution warehouses (i.e., WH1 and WH2), $D_j = 5$, $P_j = 5$, $T = 7$, $\phi_j = 0$, and ϵ as in Equation (3.16). 64

3.6 Cost per timestep over 250 episodes, as computed by the multi-stage programming model. The X-axis represents the timesteps, while the Y-axis displays the per-step cost. From both Figures 3.6a and 3.6b, which refers to the experiments of the small settings (but with $T = 21$), it is possible to observe the periodic behavior of the costs. 66
3.7 Average opt-gap (expressed as a frequency distribution) over 250 episodes in comparison to the multi-stage programming model. 69
3.8 Total costs histograms over 250 episodes in relation to PPO and DRLBD techniques. 72
4.1 Representation of the supply chain environment. 88
4.2 Order of events in the supply chain environment. 89
4.3 A 95% confidence interval based on 2000 simulated episodes for the demand in the second scenario derived from real-world data over an episode horizon of $T = 240$ timesteps. 104
4.4 Average total cost for the PPO algorithm, with lower (LB), upper (UB), and optimal (OPT) values for the OUT and PIL policies. Demand noise is modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. Each subplot shows the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$. 106
4.5 Bar plots representing the average total cost over 2000 simulated episodes, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. In each subplot, the bars show the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$. 107
4.6 Bar plots representing the average total cost over 2000 simulated episodes, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, d_t \times 15\%)$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. In each subplot, the bars show the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$. 108
A.1 Convergence trajectories during training of sQ policy (Figure A.1a), BS policy (Figure A.1b), PPO (Figure A.1c), PG (Figure A.1d), and A3C (Figure A.1e) in the most challenging experiment, which involves two product types, three local warehouses, and lead times of three. Plots related to other experiments conducted can be found in the GitHub repository associated with our open-source library (available at https://github.com/frenkowski/SCIMAI-Gym). 134

Chapter 1

Introduction

1.1 Background and Motivation

The integration of artificial intelligence (AI) and industry 4.0 principles has *transformed* supply chain inventory management (SCIM), addressing challenges like multi-echelon systems, demand variability, and operational constraints. Traditional inventory policies often lack adaptability to complex, dynamic, and high-dimensional environments under uncertainty. Deep reinforcement learning (DRL) offers a promising alternative, enabling data-driven, adaptive decision-making with the potential to outperform traditional optimization methods. However, barriers such as computational complexity, real-world data limitations, scalability issues, and interpretability concerns limit its broader adoption. In the following sections, we provide background and motivation by exploring these topics, examining the evolution of inventory policies, the opportunities in applying DRL to SCIM, and the path toward innovative solutions with the potential to enhance the efficiency and robustness of modern supply chains.

1.1.1 Artificial Intelligence and Industry 4.0 in Supply Chains

AI is a transformative domain of computer science dedicated to developing systems capable of performing tasks that typically require *human intelligence*, such as reasoning, learning, and decision-making [22]. Machine learning (ML), a branch of AI, focuses on algorithms that enable systems to identify patterns and make data-driven decisions for specific tasks without *explicit programming* [17].

The importance of AI and ML lies in their ability to process vast amounts of data efficiently, generating *actionable insights* that drive operational efficiency, reduce costs, and improve customer satisfaction. Their integration into modern industries has accelerated innovation by optimizing processes and minimizing the need for human intervention in repetitive tasks. These technologies are central to solving complex and dynamic challenges across various domains, including supply chains, where they help manage uncertainty, intricate interdependencies, and critical issues related to structural attributes

and operational factors.

As industries increasingly adopt AI and ML, their transformative power has become indispensable for maintaining competitiveness in a rapidly evolving technological landscape, enabling organizations to capitalize on emerging opportunities. Central to this impact is deep learning (DL), a specialized branch of ML that employs artificial neural networks (ANNs) to learn *non-linear data relationships* directly from raw inputs, enhancing pattern recognition and decision-making processes [8]. DL not only addresses problems previously considered intractable but also redefines the potential of automated systems, offering a significant competitive advantage in an interconnected digital economy.

This potential aligns closely with the principles of industry 4.0, also known as the *fourth industrial revolution*. Industry 4.0 integrates advanced digital technologies into manufacturing and industrial practices, including AI, the internet of things, and big data. By emphasizing the development of smart, connected, and autonomous systems, it promotes data-driven decision-making and automation, fundamentally revolutionizing traditional operational paradigms [11]. The principles of industry 4.0 are transforming supply chains by integrating advanced analytics and predictive tools that enhance efficiency in logistics, demand forecasting, and other critical operations [20].

1.1.2 Supply Chain Management and Inventory Management

A supply chain is defined as an *interconnected network* of entities, resources, activities, and processes involved in the production and delivery of products or services, from raw material suppliers to end consumers. It includes key stages such as procurement, manufacturing, transportation, warehousing, and distribution, forming a complex system of interdependent functions. Complementing this, supply chain management (SCM) refers to the *strategic coordination* of these activities to optimize efficiency, minimize costs, and improve customer satisfaction [12]. The importance of SCM has grown significantly in response to increasing globalization, digitization, and sustainability concerns, extending its influence beyond individual organizations to serve as the backbone of global trade and commerce.

Within SCM, SCIM focuses on coordinating *inventory levels* across the entire supply chain. At its core, SCIM addresses critical questions related to how much stock to hold, where to store it, and how to distribute it effectively [24]. Efficient SCIM ensures timely product delivery while optimizing resource utilization. Conversely, inefficiencies in SCIM can result in substantial financial losses, customer dissatisfaction, and operational disruptions, highlighting its indispensable role in the success of modern industries.

The *core components* of SCIM include inventory control, production planning, and logistics. Inventory control focuses on maintaining optimal inventory levels to prevent stockouts or overstocking, balancing the need to meet customer demands without incurring excessive costs. Production planning aligns manufacturing schedules with demand forecasts, optimizing resource utilization and mitigating the risk of production

delays. Logistics plays a crucial role in transporting and storing products, ensuring timely delivery to meet customer requirements. Collectively, these functions form an interconnected system that supports seamless operations across the supply chain. Advanced computational approaches are increasingly being applied to address the complexities inherent in SCIM and enhance operational efficiency.

1.1.3 Advancements through Deep Reinforcement Learning

Reinforcement learning (RL), a specialized branch of ML, has emerged as a powerful tool in this context. RL focuses on sequential decision-making in dynamic and uncertain environments, where algorithms are designed to optimize cumulative rewards by learning the optimal action to take in a given state through *continuous interaction* with the environment [19]. RL problems are typically formulated using the mathematical framework of Markov decision processes (MDPs), which define the environment in terms of states, actions, and rewards. Unlike supervised learning, which trains models using labeled data for classification and regression tasks, or unsupervised learning, which discovers hidden patterns and clusters within data, RL relies on a trial-and-error process. Through this process, agents develop policies by receiving feedback from their actions, enabling them to adapt and optimize their strategies over time [30].

DRL extends the capabilities of RL by integrating ANNs to approximate *value functions*–quantitative measures of expected future rewards–and *policy mappings*, which determine the optimal action for a given state. In environments with high-dimensional state and action spaces, where states represent the system's conditions and actions denote possible decisions, DRL excels in addressing the complexities of real-world applications, such as managing large-scale supply chains. State-of-the-art DRL algorithms include value function-based methods, such as the deep Q-network; policy mapping approaches, like the vanilla policy gradient (VPG); and hybrid techniques that combine both methodologies, such as asynchronous advantage actor-critic (A3C) and proximal policy optimization (PPO) [6].

The adoption of AI in SCM has led to transformative advancements in inventory control, production planning, and logistics, as reported in the *academic literature*. Review studies have consolidated key insights from significant papers, demonstrating how ML techniques have enhanced SCIM practices. For instance, Panzer and Bender [18] discussed the application of ML in real-time monitoring and optimization of production systems, which has become an increasingly critical aspect in the context of industry 4.0. Similarly, Khedr and S [14] emphasized the role of ML in supplier selection and demand forecasting, highlighting its potential to reduce costs and improve supply chain coordination. These findings underscore that DL techniques, such as ANNs, enable precise predictions and adaptive responses to market fluctuations, thereby contributing to developing robust and efficient supply chains.

A comprehensive review by Yan et al. [33] further highlighted the increasing adoption of RL to address dynamic decision-making challenges, particularly in inventory control

and supply chain coordination. Additionally, Boute et al. [5] proposed a roadmap outlining key design choices for applying DRL in inventory control, emphasizing its capacity to expand the scope of inventory research by providing structural policy insights. These contributions illustrate the growing relevance of RL and DRL, which have the potential to outperform traditional optimization methods in addressing complex, data-driven problems under uncertainty in modern supply chain operations.

1.1.4 Challenges in Supply Chain Inventory Management

Supply chains operate within environments characterized by *structural attributes*, such as the number of echelons and product types, and *operational factors*, including but not limited to perishability, lead times, demand variability, and capacity constraints. Together, these parameters pose significant challenges for decision-making and performance optimization [15]. By unifying these components, effective SCIM serves as the backbone of robust supply chain operations, directly influencing an organization's ability to ensure consistent product availability, reduce costs, and maintain competitiveness.

One key contributor to complexity is the number of echelons, where decisions at one stage affect other stages in a multi-echelon system. For instance, balancing inventory levels between a central factory and regional local warehouses requires careful production planning and logistics coordination in a two-echelon system [26]. Interdependencies among echelons amplify the difficulty of maintaining optimal inventory levels and service performance, especially as the number of warehouses increases.

The presence of diverse product types, each with distinct demand patterns and capacity constraints, further complicates the problem. This complexity is particularly evident in industries such as pharmaceuticals, where perishable products add another layer of complexity [29]. Perishability necessitates tracking the age of each product to ensure timely production and delivery, minimizing stockouts and overstocking while reducing waste. Efficiently managing stock turnover, prioritizing older stock, and maintaining proper storage conditions are essential to prevent spoilage and support sustainability efforts.

Lead times, defined as the delays between order placement and product receipt, are another factor contributing to the overall difficulty. Effective management of lead times requires proactive planning across the supply chain to ensure materials and products are available where and when needed. This approach is crucial for maintaining coordination, preventing disruptions, and ensuring a smooth flow of products that meets customer demands efficiently.

Demand variability represents another significant source of complexity arising from features like stochastic, seasonal, and non-stationary demand. For example, stochastic demand fluctuates unpredictably, making accurate forecasting challenging. Seasonal patterns introduce cyclical variations, such as increased sales during holidays or specific times of the year, necessitating adjustments to inventory levels to align with anticipated peaks. Non-stationary demand, driven by long-term shifts due to evolving market

fluctuations or external disruptions, requires dynamic and adaptive policies to remain aligned with changing conditions.

Finally, capacity constraints–including production and batch limits, storage availability, and transportation capabilities–represent critical factors that require proactive planning to prevent bottlenecks and ensure efficient operations. These constraints become especially important during demand surges, such as peak seasons or unexpected disruptions, necessitating careful balancing of supply chain activities while maintaining adequate inventory levels to meet customer demands.

Another crucial challenge is managing unsatisfied demand when products are unavailable. This situation can lead to either *lost sales*, resulting in permanent revenue loss and potential damage to customer relationships, or *backorders*, where customers wait for product availability, delaying revenue but incurring additional order management costs. Addressing these scenarios requires precise coordination to minimize delays and associated costs.

Ultimately, the goal in SCIM is to *maximize profit* or *minimize total costs*, including components such as ordering, production, transportation, holding, expiration, and costs associated with lost sales or backorders. Traditional approaches often struggle to account for the multifaceted nature of these challenges, resulting in suboptimal policies. The structural attributes and operational factors outlined above highlight the need for advanced tools and sophisticated methodologies. Modern advancements, including DRL, offer promising solutions by enabling dynamic, data-driven decision-making that adapts to the complexities and uncertainties of real-world supply chains.

1.1.5 Evolution of Inventory Policies

An inventory policy is a systematic framework designed to *guide decisions* regarding inventory levels, order quantities, and replenishment schedules within a supply chain. The primary objective of inventory policies is to balance customer satisfaction with cost minimization while addressing uncertainties in demand and other operational factors. Traditional inventory policies, such as *rule-based approaches*, provide straightforward frameworks for decision-making but rely on fixed parameters optimized under specific assumptions. These policies are often supported or refined by optimization methods that identify optimal or near-optimal parameters [24, 12].

Rule-based approaches, such as the (s, S)-policy and the (s, Q)-policy, have been widely adopted due to their simplicity. The (s, S)-policy maintains a fixed target inventory level S and triggers an order whenever the inventory position falls below the threshold s. Here, the inventory position is defined as the on-hand inventory plus in-transit products minus backorders (if any). For this reason, it is also known as the order-up-to (OUT) policy. Among its variations are the base-stock policy, where $s = S - 1$ [23], and the projected inventory level (PIL) policy, which estimates the expected inventory level in the future to determine the order quantity [13]. In contrast, the (s, Q)-policy specifies a reorder point s and a fixed order quantity Q. When the

inventory position drops below s, an order of size Q is placed [9]. While policies like the (s, S)-policy and the (s, Q)-policy are effective in environments with stable demand and minimal variability, their reliance on *static parameters* limits their adaptability to uncertain conditions. Optimizing parameters, such as target inventory levels and reorder points, often requires advanced mathematical techniques to ensure these policies remain effective in practice.

Optimization methods have played a critical role in developing more sophisticated inventory policies. Early models, such as the economic order quantity (EOQ), used *exact mathematical formulations* derived from operations research to calculate the optimal order quantity that minimizes total costs by balancing ordering and holding costs under deterministic demand and fixed lead times [10]. Similarly, the economic production quantity (EPQ) model extends EOQ principles by determining the optimal production lot size that minimizes total costs under deterministic conditions. While models like EOQ and EPQ provided foundational insights, they fail to capture the complexities of real-world supply chains, which often involve stochastic demands, lead times, multi-echelon systems, and other operational factors.

More sophisticated optimization methods have been developed to address these limitations [31]. Linear programming is effective for *deterministic problems*, offering solutions to optimize decisions within linear constraints [3]. Stochastic programming extends this framework by incorporating *probabilistic scenarios*, enabling decision-makers to optimize under uncertain conditions [4]. However, this technique is computationally intensive and faces scalability challenges due to the *curse of dimensionality* [2]. As the number of variables, scenarios, or interdependencies in a supply chain increases, state and action spaces grow exponentially, making exact solutions infeasible, particularly in multi-echelon systems.

Approximate methods, such as *heuristics*, have emerged as practical alternatives when exact solutions are computationally prohibitive. For example, the Silver-Meal algorithm [1] and the Wagner-Whitin method [32] are widely used in dynamic lot-sizing problems to determine optimal order quantities and replenishment schedules. These methods minimize the average cost per period by balancing ordering and holding costs. Heuristics prioritize computational efficiency and feasibility over exact precision, making them suitable for diverse operational contexts.

Optimization via *simulation* has also emerged as a complementary approach for determining optimal parameters in inventory policies, particularly for rule-based approaches [7]. This data-driven method involves simulating the performance of different parameter configurations to identify the most effective combination. Standard techniques include brute-force searches and Bayesian optimization. Despite their potential, these methods face significant scalability challenges as the computational effort required to evaluate parameter spaces grows exponentially, limiting their applicability to large-scale supply chains.

In response to these challenges, DRL offers a transformative, data-driven method that overcomes the limitations of traditional optimization methods, which often rely on static

parameters, simplifying assumptions, or predefined mathematical formulations that constrain their practical applicability. In contrast, DRL learns optimal policies directly through interaction with the environment. By leveraging ANNs to approximate value functions and policy mappings, DRL can effectively navigate high-dimensional state and action spaces, making it particularly effective for multi-echelon systems [5]. For example, while stochastic programming requires explicit modeling of probabilistic scenarios, DRL dynamically *adapts* to evolving conditions–such as stochastic demands and lead times–without the need for exhaustive scenario enumeration. This adaptability enables DRL to outperform traditional inventory policies and heuristics in environments characterized by dynamic challenges, including uncertainty and non-linear data relationships.

Moreover, a key advantage of DRL lies in its ability to *incorporate* structural attributes and operational factors, such as the number of echelons, demand variability, and capacity constraints, directly into its decision-making process. Traditional optimization methods often rely on separate models or heuristics to address these components, increasing computational complexity and reducing applicability. In contrast, DRL inherently learns policies that account for these interdependencies, optimizing order quantities at each decision period based on the *actual state* of the environment, including on-hand inventory, in-transit products, and demand forecasts, among other elements [5]. By continuously interacting with the environment and refining decisions over time, DRL achieves unparalleled flexibility and adaptability. These characteristics position DRL as a powerful tool for modern SCIM, enabling organizations to navigate the complexities of real-world supply chains effectively.

1.1.6 Barriers to Implementing Deep Reinforcement Learning in Supply Chain Inventory Management

Despite the growing interest in applying DRL to SCIM, significant barriers persist that affect its practical implementation in real-world applications, as documented in the academic literature [5, 33, 21, 16, 14]. Among the most critical issues is *computational complexity*. The intricate nature of supply chains, particularly in multi-echelon systems, requires modeling high-dimensional state and action spaces. This complexity is further exacerbated by demand variability, capacity constraints, and other operational factors, substantially increasing the computational effort required. Additionally, hyperparameter tuning represents a time-consuming, computationally intensive process that adds further challenges. As a result, extensive computational resources and advanced strategies are necessary to train DRL algorithms effectively.

Challenges related to *scalability* also represent a critical barrier. Although DRL has demonstrated potential in controlled or small-scale environments, scaling its application to large, complex supply chains–with increased numbers of echelons, warehouses, and product types–often reduces algorithmic performance. The inclusion of lead times and perishability introduces additional layers of complexity. The resulting increase in computational requirements frequently leads to longer convergence times or even algorithmic

divergence, limiting the practical applicability of DRL for large-scale problems.

Another major obstacle lies in the availability of *real-world data*. Successfully implementing DRL algorithms depends on access to comprehensive, high-quality datasets, which are typically scarce. Many organizations face limitations due to fragmented and inconsistent data sources, complicating algorithm training and compromising decision-making processes. Furthermore, the scarcity of real-world data makes evaluating DRL performance in practical applications challenging, as assumed environments may fail to capture the actual complexities of realistic supply chains. This reliance on robust data infrastructure poses a critical barrier to the adoption of DRL in real-world applications.

Finally, the *interpretability* of DRL algorithms remains a persistent barrier. Unlike traditional inventory policies, DRL algorithms can be perceived as black-box systems, where the rationale behind their decision-making is not easily interpretable. This issue is particularly problematic in highly regulated industries, such as pharmaceuticals, where decision-makers require clear explanations to ensure compliance with ethical and legal constraints. Consequently, it is essential to analyze the actions chosen by DRL algorithms and closely examine inventory trends to reveal patterns and managerial insights that help interpret the results. Such analyses can build trust in DRL-based solutions and support their broader acceptance across industries.

Addressing these significant barriers requires continued innovation and dedicated research in both methodologies and implementation strategies. Emphasizing the importance of reducing computational complexity, validating DRL effectiveness using real-world data, and enhancing the scalability and interpretability of DRL algorithms is thus essential. Such advancements are crucial for unlocking the full potential of DRL and driving its *practical application* in SCIM.

1.1.7 Opportunities for Advanced Solutions

In addition to these barriers, the studies presented in this thesis reveal further challenges facing DRL adoption. Research on two-echelon systems, for instance, highlights insufficient testing across various *combinations* of the number of warehouses, product types, and length of lead times. Moreover, a lack of comprehensive *benchmarking* persists when comparing DRL to traditional inventory policies, such as base-stock or (s, Q)-policies, and also among different state-of-the-art DRL algorithms themselves. Additionally, existing DRL algorithms often struggle with high-dimensional state and action spaces in multi-product, multi-echelon systems, leading to infeasible actions and suboptimal policies. To address these issues, exploiting *continuous action spaces*, which scale linearly with the size of the problem, in contrast to discrete action spaces that grow combinatorially, can improve effectiveness by enabling faster training convergence and more efficient decision-making.

Multi-stage (MS) stochastic programming models are commonly used for SCIM but are computationally intractable due to the curse of dimensionality, particularly under demand variability. While DRL offers an alternative method, it presents its own

challenges, including computational complexity and hyperparameter tuning. Incorporating capacity constraints, production limits, and transportation capabilities adds further complications, even for DRL. *Hybrid approaches* that integrate DRL with MS stochastic programming hold the potential to combine the optimization strengths of MS stochastic programming with the adaptability and decision-making efficiency of DRL. By leveraging these complementary strengths, such approaches can mitigate computational bottlenecks, improve solution quality, and enhance scalability. However, despite their promise, hybrid approaches have yet to be thoroughly explored.

Pharmaceutical supply chains introduce unique challenges, including perishable products, non-stationary demand, and strict ethical and legal constraints. Existing studies lack rigorous comparisons of DRL with traditional inventory policies in this context. Notably, there is a scarcity of evaluations using real-world demand data and realistic supply chain models to benchmark DRL performance against human-driven policies currently employed by industries. Addressing these gaps is essential to validate the practical applicability and effectiveness of DRL in real-world domains.

1.2 Research Questions

The research presented in this thesis aims to address the challenges and limitations associated with SCIM, particularly through the application of DRL and mathematical approaches, as discussed in the previous sections. Each paper contributes to advancing the field by tackling specific research questions and proposing *innovative methodologies* and *implementation strategies* to enhance DRL's performance and applicability in diverse supply chain contexts.

Performance of Deep Reinforcement Learning Algorithms in Two-Echelon Inventory Control Systems [27]

The first paper, discussed in Chapter 2, investigates how DRL algorithms can advance SCIM within a two-echelon system. The study focuses on the following key questions:

1. *How can DRL improve performance in multi-echelon systems under seasonal and stochastic demand scenarios, varying the number of local warehouses, product types, and the length of lead times?*
2. *Can we design an allocation rule for selecting feasible actions and preventing backorders in the first echelon?*
3. *How does the performance of DRL compare to traditional inventory policies in terms of cost minimization and training time?*

These research questions are addressed by mathematically formulating the problem as an MDP and evaluating the performance of three state-of-the-art DRL algorithms–PPO, VPG, and A3C–against traditional inventory policies, namely the base-stock and

(s, Q)-policies, through comprehensive numerical experiments assessing both costs and training time.

Key findings reveal that the PPO algorithm consistently outperforms the other algorithms across diverse experimental conditions, including variations in the length of lead times, number of product types, and number of local warehouses. While computationally efficient, the (s, Q)-policy offers a balanced trade-off between performance and implementation simplicity. These results underscore DRL's ability to handle large state-action spaces and complex demand patterns by exploiting the balanced allocation rule designed for continuous action spaces, thereby addressing critical challenges related to computational complexity and scalability.

Combining Deep Reinforcement Learning and Multi-Stage Stochastic Programming to Address the Supply Chain Inventory Management Problem [25]

The second paper, described in Chapter 3, examines the combination of DRL with MS stochastic programming and aims to answer the following research questions:

1. *How can combining DRL with MS stochastic programming optimize production and logistics decisions in two-echelon systems?*
2. *How can the hybrid heuristic adapt to changes in supply chain parameters, such as seasonal and stochastic demand patterns, production limits, warehouse capacity constraints, and fixed and variable logistics costs, while mitigating computational complexity?*
3. *Can the hybrid heuristic improve performance compared to standalone DRL and MS stochastic programming methods?*

This study introduces a hybrid heuristic combining the PPO algorithm with MS stochastic programming to enhance state-of-the-art methodologies. The PPO algorithm determines batch production quantities, leveraging its adaptive learning capabilities to manage long-horizon production processes. Simultaneously, MS stochastic programming is applied for short-horizon logistics decisions, providing a mathematical formulation to address operational constraints effectively.

Key contributions of this research include addressing the curse of dimensionality commonly associated with MS stochastic programming. This is achieved by leveraging DRL's capacity and decomposing complex decision spaces into more manageable components. The hybrid heuristic demonstrates scalability and computational efficiency, consistently outperforming standalone DRL algorithms in minimizing costs and successfully solving computationally intractable problems using standalone MS stochastic programming. Comprehensive numerical experiments show that performance remains robust under various demand scenarios, operational constraints, and costs, including production limits, warehouse capacity constraints, and fixed and variable logistics costs.

Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity [28]

The third paper, presented in Chapter 4, focuses on SCIM for perishable products in a real-world pharmaceutical supply chain. It highlights two critical challenges for DRL implementation, specifically the scarcity of real-world data for training and performance evaluation and the need for behavioral analysis to enhance interpretability. This research investigates the following questions:

1. *How should DRL and traditional inventory policies be adapted to provide robust solutions for managing non-stationary demand while considering product perishability, production yields, batch limits, lead times, and lost sales?*
2. *How does DRL compare to human-driven policies and traditional inventory policies in managing a real-world pharmaceutical product?*
3. *How do human-driven policies compare to data-driven methods, and what managerial insights can be drawn from their performance across varying experimental conditions?*

The study employs synthetic data based on information provided by a multinational pharmaceutical company to evaluate the PPO algorithm against OUT and PIL policies while modeling the problem as an MDP. Numerical experiments simulate the lifecycle of a real pharmaceutical product under varying levels of demand noise, planning horizons, and diverse supply chain parameters.

Key findings indicate that DRL offers strong performance in minimizing costs, particularly in high-risk environments characterized by significant lead times, lost sales costs, and production yields. However, it does not consistently outperform traditional inventory policies across all experimental conditions. Compared to human-driven policies, which mitigate lost sales through higher holding costs, DRL and rule-based approaches achieve lower average costs but exhibit greater cost variability. This trade-off underscores the need for decision-makers to balance cost efficiency with the variability inherent in more sophisticated data-driven methods while also considering ethical and legal constraints.

1.2.1 SCIMAI-Gym Python Library

The collective contributions of the three papers presented in this thesis represent notable advancements in the field of SCIM, particularly through the implementation of DRL to address multifaceted challenges. By combining state-of-the-art DRL algorithms with domain-specific mathematical formulations, this work advances both the theoretical understanding and the practical implementation of innovative solutions in complex, dynamic, high-dimensional environments under uncertainty. Beyond addressing specific

research questions, this work also introduces a comprehensive and flexible *open-source Python library*, called SCIMAI-Gym (which stands for SCIM–through–AI-Gym), which facilitated the resolution of these challenges.

This library enables researchers and practitioners to tackle SCIM barriers using a wide range of DRL algorithms, traditional optimization methods, and inventory policies while simultaneously promoting reproducible research through standardized and customizable experimental conditions. Its simulation environment, designed for both single- and two-echelon systems, serves as a central platform for advancing research. It offers unparalleled flexibility and adaptability in adjusting key parameters related to non-stationary, seasonal, or real-world stochastic demand patterns; lengths of lead times; numbers of product types and local warehouses; production limits and warehouse capacity constraints; perishability and production yields; and handling unsatisfied demand through backorders or lost sales. It also accommodates various cost components, including ordering, transportation, holding, and expiration costs.

In addition, the library allows extensive customization of the state vector, including information such as available and in-transit stock or age-specific inventory levels for perishable products. It also supports the integration of historical demand data or demand forecasts, enhancing the realism and applicability of simulations. Moreover, employing continuous action spaces helps address scalability and computational complexity by leveraging features such as the balanced allocation rule. It can also adjust the sequence of events within the supply chain modeled as an MDP to reflect specific operational contexts. Furthermore, the library provides tools to visualize and analyze the behavior of algorithms through detailed plots, enabling researchers and practitioners to explore the interpretability of learned policies thoroughly. By offering such an adaptable platform, the library facilitates rigorous experimentation, fosters actionable insights, and bridges the gap between methodological research and real-world applications. The open-source library is publicly available on GitHub at the following link: https://github.com/frenkowski/SCIMAI-Gym.

Collectively, the advancements presented in this thesis represent a significant step forward in applying DRL to SCIM. By demonstrating the potential of DRL to optimize complex, multidimensional decision-making processes, this work highlights the growing relevance and transformative power of these methodologies. Through its combination of theoretical mathematical rigor and practical adaptability, this thesis lays a *solid foundation* for future research and innovation in SCIM, mapping a clear path toward more robust and efficient supply chains.

References

[1] Kenneth R. Baker et al. “An Algorithm for the Dynamic Lot-Size Problem with Time-Varying Production Capacity Constraints.” In: *Management Science* 24.16

(Dec. 1978), pp. 1710–1720. ISSN: 1526-5501. DOI: 10.1287/mnsc.24.16.1710. URL: http://dx.doi.org/10.1287/mnsc.24.16.1710.

[2] Richard Bellman. "Dynamic Programming." In: *Science* 153.3731 (July 1966), pp. 34–37. ISSN: 1095-9203. DOI: 10.1126/science.153.3731.34. URL: http://dx.doi.org/10.1126/science.153.3731.34.

[3] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific books. Athena Scientific, 1997. ISBN: 9781886529199.

[4] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer New York, 2011. ISBN: 9781461402374. DOI: 10.1007/978-1-4614-0237-4. URL: http://dx.doi.org/10.1007/978-1-4614-0237-4.

[5] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[6] Vincent François-Lavet et al. "An Introduction to Deep Reinforcement Learning." In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. DOI: 10.1561/2200000071. URL: https://doi.org/10.1561/2200000071.

[7] Michael C. Fu. "Optimization for simulation: Theory vs. Practice." In: *INFORMS Journal on Computing* 14.3 (Aug. 2002), pp. 192–215. ISSN: 1526-5528. DOI: 10.1287/ijoc.14.3.192.113. URL: http://dx.doi.org/10.1287/ijoc.14.3.192.113.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262035613.

[9] G. Hadley and T.M. Whitin. *Analysis of Inventory Systems*. International series in management). Prentice-Hall, 1963. ISBN: 9780130329530.

[10] Ford W. Harris. "How Many Parts to Make at Once." In: *Operations Research* 38.6 (Dec. 1990), pp. 947–950. ISSN: 1526-5463. DOI: 10.1287/opre.38.6.947. URL: http://dx.doi.org/10.1287/opre.38.6.947.

[11] Mario Hermann, Tobias Pentek, and Boris Otto. "Design Principles for Industrie 4.0 Scenarios." In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, Jan. 2016, pp. 3928–3937. DOI: 10.1109/hicss.2016.488. URL: http://dx.doi.org/10.1109/HICSS.2016.488.

[12] Michael Hugos. *Essentials of Supply Chain Management*. Wiley, Feb. 2018. ISBN: 9781119464495. DOI: 10.1002/9781119464495. URL: http://dx.doi.org/10.1002/9781119464495.

[13] Willem van Jaarsveld and Joachim Arts. "Projected Inventory-Level Policies for Lost Sales Inventory Systems: Asymptotic Optimality in Two Regimes." In: *Operations Research* (Apr. 2024). ISSN: 1526-5463. DOI: 10.1287/opre.2021.0032. URL: http://dx.doi.org/10.1287/opre.2021.0032.

[14] Ahmed M. Khedr and Sheeja Rani S. "Enhancing supply chain management with deep learning and machine learning techniques: A review." In: *Journal of Open Innovation: Technology, Market, and Complexity* 10.4 (Dec. 2024), p. 100379. ISSN: 2199-8531. DOI: 10.1016/j.joitmc.2024.100379. URL: http://dx.doi.org/10.1016/j.joitmc.2024.100379.

[15] Ton de Kok et al. "A typology and literature review on stochastic multi-echelon inventory models." In: *European Journal of Operational Research* 269.3 (Sept. 2018), pp. 955–983. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.02.047. URL: http://dx.doi.org/10.1016/j.ejor.2018.02.047.

[16] Chengxi Li et al. "Deep reinforcement learning in smart manufacturing: A review and prospects." In: *CIRP Journal of Manufacturing Science and Technology* 40 (Feb. 2023), pp. 75–101. ISSN: 1755-5817. DOI: 10.1016/j.cirpj.2022.11.003. URL: http://dx.doi.org/10.1016/j.cirpj.2022.11.003.

[17] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN: 9780071154673.

[18] Marcel Panzer and Benedict Bender. "Deep reinforcement learning in production systems: a systematic literature review." In: *International Journal of Production Research* 60.13 (Sept. 2021), pp. 4316–4341. ISSN: 1366-588X. DOI: 10.1080/00207543.2021.1973138. URL: http://dx.doi.org/10.1080/00207543.2021.1973138.

[19] Warren B. Powell. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. Wiley, Apr. 2022. ISBN: 9781119815068. DOI: 10.1002/9781119815068. URL: http://dx.doi.org/10.1002/9781119815068.

[20] Maciel M. Queiroz et al. "Industry 4.0 and digital supply chain capabilities: A framework for understanding digitalisation challenges and opportunities." In: *Benchmarking: An International Journal* 28.5 (Dec. 2019), pp. 1761–1782. ISSN: 1463-5771. DOI: 10.1108/bij-12-2018-0435. URL: http://dx.doi.org/10.1108/BIJ-12-2018-0435.

[21] Benjamin Rolf et al. "A review on reinforcement learning algorithms and applications in supply chain management." In: *International Journal of Production Research* 61.20 (Nov. 2022), pp. 7151–7179. ISSN: 1366-588X. DOI: 10.1080/00207543.2022.2140221. URL: http://dx.doi.org/10.1080/00207543.2022.2140221.

[22] S.J. Russell, P. Norvig, and E. Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. ISBN: 9780136042594.

[23] Herbert Scarf. "The Optimality of (S, s) Policies in the Dynamic Inventory Problem." In: *Herbert Scarf's Contributions to Economics, Game Theory and Operations Research*. Palgrave Macmillan UK, 2005, pp. 1–7. ISBN: 9781137024381. DOI: 10.1057/9781137024381_1. URL: http://dx.doi.org/10.1057/9781137024381_1.

[24] Edward A. Silver, David F. Pyke, and Douglas J. Thomas. *Inventory and Production Management in Supply Chains*. CRC Press, Dec. 2016. ISBN: 9781466558625. DOI: 10.1201/9781315374406. URL: http://dx.doi.org/10.1201/9781315374406.

[25] Francesco Stranieri, Edoardo Fadda, and Fabio Stella. "Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem." In: *International Journal of Production Economics* 268 (Feb. 2024), p. 109099. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109099. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109099.

[26] Francesco Stranieri and Fabio Stella. "Comparing Deep Reinforcement Learning Algorithms in Two-Echelon Supply Chains." In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer Nature Switzerland, 2025, pp. 454–469. ISBN: 9783031746406. DOI: 10.1007/978-3-031-74640-6_37. URL: http://dx.doi.org/10.1007/978-3-031-74640-6_37.

[27] Francesco Stranieri, Fabio Stella, and Chaaben Kouki. "Performance of deep reinforcement learning algorithms in two-echelon inventory control systems." In: *International Journal of Production Research* 62.17 (Mar. 2024), pp. 6211–6226. ISSN: 1366-588X. DOI: 10.1080/00207543.2024.2311180. URL: http://dx.doi.org/10.1080/00207543.2024.2311180.

[28] Francesco Stranieri et al. *Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity*. 2025. DOI: 10.48550/ARXIV.2501.10895. URL: https://arxiv.org/abs/2501.10895.

[29] Francesco Stranieri et al. "Drug Inventory Control: Human Decisions Versus Deep Reinforcement Learning." In: *CEUR workshop proceedings* 3650.3 (Mar. 2024). ISSN: 1613-0073. URL: https://ceur-ws.org/Vol-3650/paper3.pdf.

[30] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039246.

[31] Germán Herrera Vidal. "Deterministic and Stochastic İnventory Models in Production Systems: a Review of the Literature." In: *Process Integration and Optimization for Sustainability* 7.1–2 (Dec. 2022), pp. 29–50. ISSN: 2509-4246. DOI: 10.1007/s41660-022-00299-3. URL: http://dx.doi.org/10.1007/s41660-022-00299-3.

[32] Harvey M. Wagner and Thomson M. Whitin. "Dynamic Version of the Economic Lot Size Model." In: *Management Science* 5.1 (Oct. 1958), pp. 89–96. ISSN: 1526-5501. DOI: 10.1287/mnsc.5.1.89. URL: http://dx.doi.org/10.1287/mnsc.5.1.89.

[33] Yimo Yan et al. "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities." In: *Transportation Research Part E: Logistics and Transportation Review* 162 (June 2022), p. 102712. ISSN: 1366-5545. DOI: 10.1016/j.tre.2022.102712. URL: http://dx.doi.org/10.1016/j.tre.2022.102712.

Chapter 2

Performance of Deep Reinforcement Learning Algorithms in Two-Echelon Inventory Control Systems

Francesco Stranieri[a,b], Fabio Stella[a], and Chaaben Kouki[c]

[a]Department of Informatics, Systems, and Communication (DISCo), University of Milano-Bicocca, Viale Sarca, 336, Milan, 20126, Italy
[b]Department of Control and Computer Engineering (DAUIN), Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy
[c]Department of Operations Management and Decision Science (OMDS), ESSCA School of Management, Rue Joseph Lakanal, 1, Angers, 49100, France

This is the original manuscript of an article published by Taylor & Francis in the International Journal of Production Research on 1 March 2024, available at: https://doi.org/10.1080/00207543.2024.2311180.

Abstract

This study conducts a comprehensive analysis of deep reinforcement learning (DRL) algorithms applied to supply chain inventory management (SCIM), which consists of a sequential decision-making problem focused on determining the optimal quantity of products to produce and ship across multiple capacitated local warehouses over a specific time horizon. In detail, we formulate the problem as a Markov decision process for a divergent two-echelon inventory control system characterized by stochastic and seasonal demand, also presenting a balanced allocation rule designed to prevent backorders in the first echelon. Through numerical experiments, we evaluate the performance of state-of-the-art DRL algorithms and static inventory policies in terms of both cost minimization and training time while varying the number of local warehouses and product types and the length of replenishment lead times. Our results reveal that the Proximal Policy Optimization algorithm consistently outperforms other algorithms across all experiments, proving to be a robust method for tackling the SCIM problem. Furthermore, the (s, Q)-policy stands as a solid alternative, offering a compromise in performance and computational efficiency. Lastly, this study presents an open-source software library that provides a customizable simulation environment for addressing the SCIM problem, utilizing a wide range of DRL algorithms and static inventory policies.

2.1 Introduction

In this study, we address an *optimization problem* that consists of a two-echelon inventory control system with limited capacity at each echelon. The system includes a capacitated source (central warehouse) that receives products from the upstream level (factory). The downstream level consists of multiple retailers (local warehouses) that face stochastic and seasonal demands for a diverse range of product types. Excess demand for each type of product that cannot be directly satisfied from stocks is backordered. Local warehouses place orders at the upstream level, and these orders are delivered after a constant and positive replenishment lead time. The objective is to minimize the total cost of the system under consideration.

This problem can be viewed as a nonlinear stochastic problem whose optimal solution is challenging to find due to the *curse of dimensionality*, which emerges from the number of product types, the number of warehouses, the presence of lead times, and the stochastic nature of the demand [28]. The simplest solutions available for one-warehouse-one-product-type inventory control systems or employed in serial supply chains cannot be extended to systems involving multiple warehouses and multiple product types in a multi-echelon structure as the complexity grows with the problem size. By relaxing the assumption of stochastic demand and replacing it with a deterministic value (e.g., with its mean), the system can be modeled as a linear programming problem, which can be technically solved for a reasonable number of warehouses and product types and length of lead times. Nevertheless, for supply chain systems with large-scale instances, even when considering deterministic demand, the problem remains arduous to solve.

Consequently, the goal is to identify an *approximate solution* that provides robust performance in terms of cost minimization or profit maximization, all while maintaining limited computation time. A common approximate approach for dealing with the supply chain inventory management (SCIM) problem involves formulating the system as a Markov decision process (MDP) and solve it approximately. For further information on approximate techniques for solving MDPs, the interested reader can refer to the work of Gordon [17].

In this context, artificial intelligence (AI) enhances the decision-making capabilities and operational efficiency in supply chains, leading to its *expanding role* in the SCIM field [41]. For example, Jackson, Jesus Saenz, and Ivanov [22] demonstrated how recent advances in AI and natural language processing can be leveraged to enable human experts to automatically build simulation models of inventory control systems using verbal descriptions. Additionally, artificial neural networks (ANNs) have been effectively employed to propose a deep learning method for demand forecasting, which has proven robust compared to other benchmarks used in the study [35], and to develop a predictive model that determines the likelihood of product backorders in complex scenarios, particularly those characterized by unbalanced training datasets [40]. According to Rolf et al. [36], the most common application of reinforcement learning (RL) in supply chains lies in addressing the SCIM problem, with Q-learning emerging as the

most utilized algorithm. However, to overcome the limitations of traditional tabular RL algorithms such as Q-learning, deep reinforcement learning (DRL) has recently emerged as a promising approach for efficiently tackling MDP problems.

Given the recent advancements and the increasing application of deep learning in SCIM, this study aims to investigate this trend by comparing different state-of-the-art DRL algorithms in two-echelon inventory control systems. In detail, we focus on evaluating the effectiveness of DRL algorithms by comparing their performance in terms of cost minimization and computation time. It is worth noting that the considered problem remains challenging to solve, even with the implementation of DRL algorithms. Consequently, we assess their performance by simplyfing certain assumptions of the original system. Specifically, we limit the set of experiments by considering one and two product types, a number of local warehouses ranging from one to three, and replenishment lead times of length one and three. These constraints allow all the algorithms used in the benchmark to converge within a reasonable amount of time. The primary objective of this study is thus to evaluate the effectiveness of DRL algorithms and compare their performance with static inventory policies. Through this effort, we aim to provide insights into the relative strengths of each approach and contribute valuable information for decision-makers in selecting appropriate SCIM strategies that balance performance and computational efficiency.

In what follows, Section 2.2 will offer a review of recent literature on SCIM, specifically focusing on the use of DRL algorithms in this domain. Then, in Section 2.3, we will outline the general MDP framework of DRL algorithms and detail their application within our assumed inventory control system. In Section 2.4, we will conduct numerical experiments to assess the performance of the different DRL algorithms and static inventory policies. Finally, Section 2.5 will conclude the paper, summarizing our findings and discussing their implications for both researchers and practitioners in the field of SCIM.

2.2 Literature Review

The SCIM problem is a *sequential decision-making problem* that involves determining both the optimal production quantities at the factory and the optimal shipment quantities from the central warehouse to local warehouses over a specified time horizon (either finite or infinite). As evidenced by the roadmap provided by Boute et al. [4], DRL algorithms are rarely applied to the SCIM field. Nevertheless, they hold the potential to develop near-optimal policies that are challenging, if not impossible, to achieve using traditional methods. Indeed, the uncertain and stochastic nature of product demand, combined with lead times, presents significant challenges for mathematical programming approaches to be effective. Furthermore, when the inventory control system accounts for limited storage capacity, identifying effective SCIM strategies becomes even more complex.

As observed by Kok et al. [25], the *optimal policy* for capacitated two-echelon

inventory control systems is multidimensional and infeasible for real-world scenarios. In this context, installation stock and echelon stock policies represent two primary strategies for defining ordering decisions. Under installation stock policies, decisions are based solely on the local inventory position at each warehouse, whereas under echelon stock policies, orders are made based on the inventory position across downstream levels of the supply chain [3].

Classical static inventory policies, such as the base-stock policy [10] and the (s, Q)-policy [37], can be implemented under either installation or echelon control logic. Specifically, the base-stock policy has been extensively investigated in multi-echelon structures. De Bodt and Graves [11] analyze it under stochastic demand in serial supply chains, considering backorders at the final level. Chen [9] derive an optimal base-stock policy for similar structures, extending them with batch ordering constraints. Kouki, Arts, and Babai [26] evaluate the performance of the base-stock policy in a two-echelon supply chain with lost sales, offering exact and approximate solutions under stochastic demand conditions and lead times. Similarly, the (s, Q)-policy has been applied across different two-echelon inventory control systems with stochastic demand. Forsberg [13] evaluate the (s, Q)-policy, providing exact methods for cost analysis under Poisson demand and constant lead times. Lee and Wu [27] study it and propose strategies to mitigate the bullwhip effect, which refers to the amplification of order fluctuations at upstream levels. Ntio et al. [32] investigate the (s, Q)-policy within divergent two-echelon supply chains, considering stochastic lead times and particularly emphasizing the challenges associated with lost sales.

The relative advantage of installation stock versus echelon stock policies depends on the specific structure of the supply chain [2]. In line with this observation, we adopted *installation stock policies* in our numerical experiments as benchmark strategies for evaluating the performance of DRL algorithms. Although decisions are based on local inventory positions, the policy parameters were optimized globally to minimize the total supply chain cost, as will be explained in Section 2.4.2.

Interested readers can refer to Bouti, Boukallal, and El Khoukhi [5] for an extended overview of classical inventory policies. In what follows, we review studies that investigate solutions to inventory control systems through RL and DRL algorithms.

Reinforcement Learning Algorithms

RL algorithms have recently achieved remarkable results in the field of AI. Essentially, RL adopts the MDP framework to represent the interactions between a learning agent and an environment [45], as depicted in Figure 2.1. To define this interaction, at each time step t, the agent observes the current state of the environment, s_t, chooses an action, a_t, and receives a reward, $r_{t+1} \in \mathbb{R}$; then, the environment transitions into a new state, s_{t+1}. The primary objective of RL is to find an optimal policy that maximizes the sum of *expected discounted rewards*, $\sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$, where $0 \leq \gamma \leq 1$ represents a hyperparameter known as the discount rate.

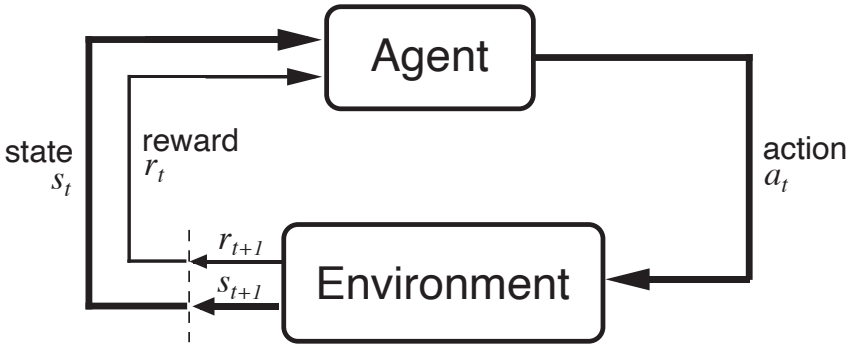


Figure 2.1: Agent-environment interaction in an MDP (taken from Sutton and Barto [45]).

One of the most common approaches to addressing the SCIM problem through RL algorithms is Q-learning [50, 36]. This approach is rooted in a tabular and temporal-difference algorithm that learns to determine the *value* of an action a_t in a state s_t, referred to as the Q-value and denoted $Q\left(s_t, a_t\right)$. The Q-values for each state-action pair are stored in a table known as the Q-table, where each state corresponds to a row and each action to a column. Through the Q-learning algorithm, the Q-values associated with each state-action pair are estimated according to an updating rule. Once convergence has been achieved – which is guaranteed under certain conditions [21] – an optimal policy can be derived by identifying the action with the highest Q-value for each state.

Chaharsooghi, Heydari, and Zegordi [7] proposed a method based on Q-learning to tackle the beer game problem, which consists of a serial supply chain structure with four participants and specific assumptions. In particular, they defined the current system state using a vector that consists of the current four stock levels. However, since the stock levels thus defined could potentially take infinite values, directly applying this strategy appears unfeasible since the Q-table would become infinitely large. Consequently, the authors *discretized* the state space into nine intervals, resulting in 9^4 possible state values. Regarding actions, their method determines the number of products to order using the $d{+}x$ policy (precisely, if a participant in the previous time step received a request for d units from the downstream level, this policy requires ordering $d{+}x$ units to the upstream level in the current time step). The primary goal of the learning process is to determine the optimal value of the unknown variable x based on the given system state. To keep the Q-table's size manageable, the authors *constrained* x to values within the range [0, 3], resulting in 4^4 possible actions.

By defining limited state and action spaces, the resulting Q-table becomes more manageable. However, it becomes evident that the Q-tables implemented are typically huge and, thus, *unscalable*. For example, the Q-table adopted by Chaharsooghi, Heydari, and Zegordi [7] contains ($9^4 \cdot 4^4 =$) 1679616 cells, equivalent to the number of states multiplied by the number of actions. Consequently, expanding the size of the state or

action spaces might not be feasible, as the Q-tables can no longer be handled. Geevers, Hezewijk, and Mes [15] also considered the beer game problem and demonstrated that the method proposed by Chaharsooghi, Heydari, and Zegordi [7] is unable to learn an effective policy since the impact of the variable x is so limited that even *random values* yield basically the same outcomes.

In conclusion, tabular RL approaches are feasible only when state and action spaces are discretized or constrained. However, such limitations can lead to a *loss of critical information*, in addition to being inappropriate for real-world scenarios. As a result, enhanced RL methods are necessary for effectively addressing multi-echelon inventory control systems.

Deep Reinforcement Learning Algorithms

DRL combines RL with *deep learning*, enabling it to tackle decision-making problems previously considered intractable. DRL is particularly well-suited for environments with high-dimensional state and action spaces, such as video games and gaming in a general sense [30, 42, 46]. Deep learning is rooted in ANNs, which are capable of providing optimal approximations for highly nonlinear functions. Basically, function parameters are adjusted during the learning process to minimize or maximize a specific objective function.

The DRL algorithms we used belong to a class called *policy-based methods*. These methods can learn a parameterized and stochastic policy to select actions directly, in contrast to value-based methods like Q-learning. Under this class of algorithms, policy gradient methods offer a significant theoretical advantage due to the *policy gradient theorem*, and the vanilla policy gradient (PG) algorithm [48] is a natural result of this theorem. However, the high variance of gradient estimates can lead to policy update instabilities [49]. To mitigate this issue, Schulman et al. [39] introduced the trust region policy optimization (TRPO) algorithm, an actor-critic method that simultaneously learns a policy and a value function while bounding the difference between the new and the old policies in a trust region. Proximal policy optimization (PPO) [38] shares the same background as TRPO but is simpler to implement and tune while demonstrating comparable or superior performance. Mnih et al. [29] proposed the asynchronous advantage actor-critic (A3C) algorithm, which involves multiple agents interacting with different instances of the environment, each with its parameters. These agents periodically and asynchronously update a global ANN that incorporates shared parameters. For a more in-depth and rigorous discussion of various DRL algorithms, interested readers can refer to François-Lavet et al. [14].

The implementation of DRL algorithms in inventory control systems has been limited, but some promising results have been reported in recent studies. Oroojlooyjadid et al. [33] used an extension of the deep Q-network (DQN) proposed by Mnih et al. [30] for the beer game problem. Their DQN agent, which uses an ANN instead of a Q-table to return Q-values for state-action pairs, learned a near-optimal policy while other supply

chain participants followed a base-stock policy. Due to the DQN's requirement for a limited action space cardinality, the authors conducted numerical experiments using a $d+x$ policy imposing different constraints on the variable x.

Peng et al. [34] proposed the PG algorithm to address a two-echelon inventory control system characterized by stochastic and seasonal demand while considering storage capacity constraints. Their experiments demonstrated that the PG agent outperformed the (s, Q)-policy used as a baseline in all three experiments. Gijsbrechts et al. [16] applied and tuned the A3C algorithm to a similar supply chain structure, showing that A3C achieved performance comparable to state-of-the-art heuristics and approximate dynamic programming algorithms despite its initial tuning being computationally intensive. Alternatively, Hezewijk et al. [18] investigated the capacitated single-echelon lot-sizing problem – a type of production planning problem – proposing modifications to the standard PPO algorithm, which matched state-of-the-art benchmarks. These modifications included the use of a feasibility mask and the scaling of states and rewards, aimed at reducing the action space size in large-scale instances.

For more extended inventory control systems, Hubbs et al. [19] studied a four-echelon structure and employed the PPO algorithm in two different environments (i.e., without and with backorders), demonstrating that PPO outperformed the base-stock policy in both cases. Finally, Alves and Mateus [1] considered a similar structure and proposed the PPO algorithm to deal with the problem, while a deterministic linear programming model (i.e., considering deterministic demand) is employed as a baseline. Results of numerical experiments showed that PPO still achieved satisfactory performance.

While these studies highlight the potential of DRL algorithms in multi-echelon inventory control systems, more research is needed to further explore their abilities in this domain.

Contributions to Literature

The current state of DRL algorithms applied in multi-echelon inventory control systems presents several *limitations*. These include insufficient testing across various supply chain typologies (e.g., by varying the number of local warehouses), lack of extensive experiments with different configurations (e.g., number of product types, length of lead times, and associated costs), and an absence of comprehensive comparisons between different state-of-the-art DRL algorithms.

Moreover, relevant aspects of the SCIM problem have not yet been efficiently addressed. Notable gaps include the lack of a well-established open-source library for reproducing and validating simulation models, insufficient performance benchmarking against an oracle, and an absence of multiple product types in multi-echelon inventory control systems, which results in a high-dimensional action space that necessitates a more interconnected and complex set of constraints. In this context, our work contributes to the existing literature by avoiding *infeasible actions* through specific constraints and employing an allocation rule designed to prevent backorders in the first echelon. This

contrasts with approaches that combine RL with operations research methods [47] or rely on knowledge of the optimal solution in small-scale instances to model the action space in larger-scale ones [18].

In an attempt to address these shortcomings, this paper offers the following *contributions to the literature* on multi-product types and multi-echelon inventory control systems:

- Design and formulation of a simulation environment representing a divergent two-echelon inventory control system under stochastic and seasonal demand, avoiding central warehouse backorders and allowing for an arbitrary number of warehouses and product types to be managed.
- Comparison of a set of state-of-the-art DRL algorithms in terms of their ability to find an optimal policy, i.e., a policy that minimizes the supply chain costs as determined by an oracle.
- Evaluation of performance achieved by DRL algorithms and comparison with selected static inventory policies, such as the base-stock policy and the (s, Q)-policy, with their optimal parameters determined through a data-driven approach.
- Design and execution of a rich experimental setup involving different system configurations (i.e., considering different lead time lengths and varying numbers of local warehouses and product types) to assess the performance of DRL algorithms and static inventory policies in a wide range of experiments.
- Design and development of an open-source library [1] for addressing the presented SCIM problem, thus embracing open science principles and guaranteeing reproducible results.

Through addressing these gaps and offering these contributions, this paper aims to advance the understanding and application of DRL algorithms in multi-echelon inventory control systems and provides a foundation for future research and advancements in this domain.

2.3 Problem Definition and Modelling

We consider a finite-horizon, periodic-review, two-echelon inventory control system composed of a factory and its central warehouse (first echelon), denoted by the index zero, along with J local warehouses (second echelon). The structure of the system under consideration is depicted in Figure 2.2 and is defined as *divergent* because, while the first level consists of a single central warehouse, the second level can include multiple

[1] Our open-source library is available at https://github.com/frenkowski/SCIMAI-Gym.

local warehouses. Each local warehouse $j, j \in \{1, \dots, J\}$, faces, for each time step $t, t \in \{1, \dots, T\}$, and each product type $i, i \in \{1, \dots, I\}$, a stochastic and seasonal demand. Any unsatisfied unit of demand that exceeds the on-hand stocks is backordered. Contextually, the central warehouse produces and ships products to local warehouses throughout the finite episode horizon of length T. Products produced by the factory are available to the central warehouse at the same time step, while products shipped to the local warehouse j are received after a positive and constant lead time L_j.

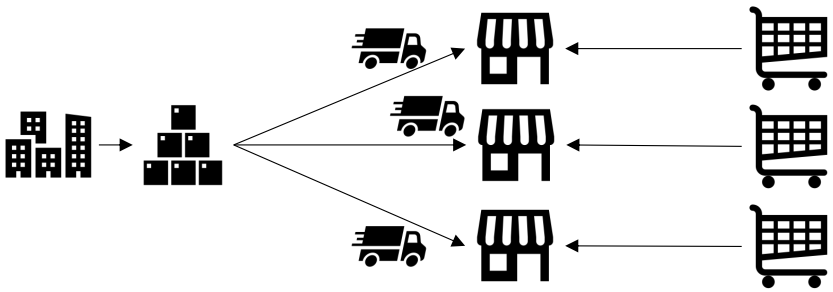


Figure 2.2: A divergent two-echelon supply chain consisting of a factory and its central warehouse (first echelon) and three local warehouses (second echelon). Shopping carts represent customer demands.

The inventory control system proposed in this study primarily draws inspiration from the models presented in Kemmer et al. [24], Peng et al. [34], and Stranieri and Stella [44]. However, we introduce two critical distinctions to more accurately characterize complex inventory control systems. First, we extended the model to consider a *multi-product type* case in a multi-echelon structure rather than focusing exclusively on a single product type. This extension not only prevents central warehouse backorders but also enables an assessment of the performance in challenging and heterogeneous configurations where multiple product types are managed simultaneously. Secondly, we incorporated *positive lead times* between the central and local warehouses. By accounting for lead times, we capture the inherent uncertainty and delay in the replenishment strategy, which can significantly influence SCIM decisions and the overall system process, allowing for a more in-depth exploration of how lead times impact performance. These distinctions to the original models aim to offer a more comprehensive understanding of the effectiveness and applicability of the DRL algorithms and static inventory policies in managing complex inventory control systems.

For every time step t and each product type i, the factory determines its production level $a^i_{0,t}$, which represents the number of units to produce, considering a fixed production cost of p^i_0 per unit (as mentioned before, we assume $j = 0$ for the central warehouse and $1 \leq j \leq J$ for local warehouses). The central warehouse can store a maximum of c^i_0 units for product type i, implying that the overall capacity is $\sum_{i=1}^{I} c^i_0 = c_0$. The cost of storing one unit of product type i at the central warehouse is h^i_0 per time step, while the

corresponding stocks on hand, also referred to as *stock level*, at time step t equals $q^i_{0,t}$.

At every time step t, $a^i_{j,t}$ units of product type i are shipped from the central to local warehouse j, with an associated transportation cost of z^i_j per unit. For each product type i, each local warehouse j has a maximum storage capacity of c^i_j (resulting in an overall capacity of $\sum_{i=1}^{I} c^i_j = c_j$), a storage cost of h^i_j per unit and time step, and a stock level at time t equal to $q^i_{j,t}$. The demand for product type i at local warehouse j during time step t is indicated by $d^i_{j,t}$.

Products in the considered inventory control system are assumed to be non-perishable and provided in discrete quantities. In addition, we assume that each local warehouse is legally obligated to satisfy all submitted demands. Unsatisfied demands are designed as a negative stock level (corresponding to *backorders*) and maintained over time. Consequently, if a demand for a specific time step exceeds the corresponding stock level, a backorder cost, denoted by b^i_j, is applied for each time step and unit of negative stocks. This assumption implies that when the backorder cost is specifically high, the agent may find it challenging to develop cost-effective solutions if it incurs backorders. As a result, an agent should prefer a policy that leads to preserving stocks in advance, even if it means incurring storage costs. By including backorder costs in our model, we offer a more complex and practical scenario for evaluating the DRL algorithms and static inventory policies, enhancing our understanding of the trade-offs between satisfying customer demands and reducing supply chain costs. A summary of the SCIM notation is available in Table 2.1.

Table 2.1: The adopted SCIM notation accompanied by a relative explanation (and units of measure).

Parameter	Explanation	Parameter	Explanation
I	Number of Product Types	p^i_0	Production Cost (per unit)
J	Number of Local Warehouses	z^i_j	Transportation Cost (per unit)
T	Episode Horizon (time step)	$q^i_{j,t}$	Stock Level (units)
L_j	Length of Lead Times (time step)	c^i_j	Storage Capacity (units)
$d^i_{j,t}$	Demand (units)	h^i_j	Storage Cost (per unit and time step)
$a^i_{j,t}$	Production and Shipping Level (units)	b^i_j	Backorder Cost (per unit and time step)

2.3.1 Markov Decision Process Formulation

In this subsection, we formalize the considered SCIM problem using the previously described MDP framework. More precisely, we introduce and define the main components related to the MDP formulation, that is, the *state vector*, the *action vector*, and the *reward function*.

State Vector

Considering time step t, we denote the stocks on hand at warehouse j, $j \in \{0, \dots, J\}$, by:

$$\mathbf{q}_{j,t} := \begin{pmatrix} q^1_{j,t} \\ q^2_{j,t} \\ \vdots \\ q^I_{j,t} \end{pmatrix}, \tag{2.1}$$

where $q^i_{j,t}$ represents the stock level for product type i, $i \in \{1, \dots, I\}$. It is important to note that, due to backorders, the stock level can assume negative values.

The quantity of product type i already ordered from local warehouse j but not yet delivered at time step $t-1, \dots, t-L_j+1$ is represented by $x^i_{j,t}$. In this respect, we express the vector of total *in-transit orders* as follows:

$$\mathbf{x}_{j,t} := \begin{pmatrix} x^1_{j,t-1} & x^1_{j,t-2} \cdots & x^1_{j,t-L_j+1} \\ x^2_{j,t-1} & x^2_{j,t-2} \cdots & x^2_{j,t-L_j+1} \\ \vdots & \vdots & \vdots \\ x^I_{j,t-1} & x^I_{j,t-2} \cdots & x^I_{j,t-L_j+1} \end{pmatrix}. \tag{2.2}$$

The state vector of the MDP can then be defined as:

$$\mathbf{s}_t := (\mathbf{q}_{0,t}, \mathbf{q}_{1,t}, \mathbf{x}_{1,t}, \dots, \mathbf{q}_{J,t}, \mathbf{x}_{J,t}). \tag{2.3}$$

Action Vector

Regarding the actions, the quantities produced and shipped at time step t are denoted by:

$$\mathbf{a}_{j,t} := \begin{pmatrix} a^1_{j,t} \\ a^2_{j,t} \\ \vdots \\ a^I_{j,t} \end{pmatrix}, \tag{2.4}$$

where $j = 0$ indicates the production level and $1 \leq j \leq J$ denotes the shipping level, respectively. In our formulation, we assume that for every warehouse j, $j \in \{0, \dots, J\}$, and every product type i, $i \in \{1, \dots, I\}$, the quantities shipped or produced cannot exceed their respective storage capacities, formally $a^i_{j,t} \leq c^i_j$.

The action vector of the MDP is consequently defined as:

$$\mathbf{a}_t := (\mathbf{a}_{0,t}, \dots, \mathbf{a}_{J,t}), \tag{2.5}$$

with a size of $I(J+1)$.

Reward Function

The goal of our study is to minimize the total cost associated with the assumed inventory control system. Consequently, the optimization problem we aim to address is a multi-variable, non-linear, non-convex, mixed-integer programming problem (MINPP).

Mathematically, this optimization problem can be defined as follows:

$$\min \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{i=1}^{I} z_j^i a_{j,t}^i + h_j^i (q_{j,t}^i)^+ + b_j^i (-q_{j,t}^i)^+ \quad + \sum_{t=1}^{T}\sum_{i=1}^{I} p_0^i a_{0,t}^i + h_0^i q_{0,t}^i \tag{2.6}$$

$$\text{s.t. } a_{j,t}^i \leq c_j^i \qquad \forall\, j \in \{0,\dots,J\}, \forall\, i \in \{1,\dots,I\} \tag{2.7}$$

$$a_{j,t}^i + q_{j,t}^i \leq c_j^i \qquad \forall\, j \in \{0,\dots,J\}, \forall\, i \in \{1,\dots,I\} \tag{2.8}$$

$$\sum_{j=1}^{J} a_{j,t}^i \leq q_{0,t}^i \qquad \forall\, i \in \{1,\dots,I\} \tag{2.9}$$

$$q_{0,t}^i = q_{0,t-1}^i + a_{0,t}^i - \sum_{j=1}^{J} a_{j,t-1}^i \qquad \forall\, j \in \{0\}, \forall\, i \in \{1,\dots,I\} \tag{2.10}$$

$$q_{j,t}^i = q_{j,t-1}^i + x_{j,t-L_j}^i - d_{j,t}^i \qquad \forall\, j \in \{1,\dots,J\}, \forall\, i \in \{1,\dots,I\} \tag{2.11}$$

$$a_{j,t}^i \in \mathbb{N} \qquad \forall\, j \in \{0,\dots,J\}, \forall\, i \in \{1,\dots,I\} \tag{2.12}$$

$$q_{j,t}^i \in \mathbb{N} \qquad \forall\, j \in \{0\}, \forall\, i \in \{1,\dots,I\} \tag{2.13}$$

$$q_{j,t}^i \in \mathbb{Z} \qquad \forall\, j \in \{1,\dots,J\}, \forall\, i \in \{1,\dots,I\}, \tag{2.14}$$

where $(\cdot)^+$ denotes $max(0,\cdot)$.

The objective function expressed in Equation (2.6) consists of several terms representing various costs. The first term accounts for the transportation cost of quantity $a_{j,t}^i$ from the central to local warehouses. The second term quantifies the storage cost at local warehouses considering the current stock level. This latter is updated through Equation (2.11) by adding incoming orders $x_{j,t-L_j}^i$ and subtracting the demand $d_{j,t}^i$ from the on-hand stocks $q_{j,t-1}^i$ at the time step $t-1$. The third term, analogous to the second, represents the backorder cost. Clearly, it is possible to incur either storage or backorder costs, but not both, depending on whether the stock level, $q_{j,t}^i$, is positive or negative. Lastly, the final two terms delineate the production and storage costs at the central warehouse, respectively.

Moving on to the constraints, Equations (2.7) and (2.8) address the capacity bounds, c_j^i, of each warehouse j and product type i. Specifically, Equation (2.7) ensures that neither the production nor the shipping level can exceed the warehouse's maximum capacity. Furthermore, Equation (2.8) also considers the current stock level in a more

restrictive form. Equation (2.9) specifies that the total sum of products shipped to all local warehouses of a specific product type i cannot exceed the on-hand stocks in the central warehouse, $q^i_{0,t}$. This constraint is crucial to guarantee that the central warehouse does not fall into backorders. Finally, Equations (2.10) and (2.11) delineate the stock dynamics for the central and local warehouses, respectively, while Equations (2.12) to (2.14) define the domains for the variables, ensuring that production or shipment of a negative number of products is not allowable.

It should be noted that our MINPP formulation is challenging to solve as it is an *NP-hard problem*. When considering a deterministic version of the model (for example, assuming demand as deterministic or known in advance), the MINPP becomes a linear programming problem solvable for a proper number of warehouses and product types and length of lead times. However, with stochastic and seasonal demand distributions, even stochastic programming approaches struggle to find an effective solution within a reasonable amount of time, mainly due to the curse of dimensionality [43]. In what follows, we consequently focus on a simplified versions of our MINPP. Specifically, we address the problem under the conditions of $I = \{1, 2\}$, $J = \{1, 2, 3\}$, and $L = \{1, 3\}$. This implies that only one or two product types are sold by all local warehouses, which can range in number from one to three, with lead times of either one or three. These lead times apply uniformly to all product types and local warehouses. Consequently, we removed the subscript j from L_j. Although we have defined specific values for I and J in our assumptions, we use the notation I and J in a more general sense. In the end, the size of the MDP state vector reduces to $I + IJ(L + 1)$.

Based on our MDP formulation and exploiting the MINPP problem, we can now define the reward at time t when taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$ as:

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = -\left(\sum_{j=1}^{J} \sum_{i=1}^{I} z^i_j a^i_{j,t} + h^i_j (q^i_{j,t})^+ + b^i_j (-q^i_{j,t})^+ + \sum_{i=1}^{I} p^i_0 a^i_{0,t} + h^i_0 q^i_{0,t} \right). \tag{2.15}$$

The objective of the MDP is thus to determine an optimal policy that maximizes the sum of expected discounted rewards through a mapping between states and actions. Specifically, our approach employs DRL algorithms trained to minimize the supply chain costs (thereby defining the reward as the negative cost).

System Dynamics

Let us define the demand vector for all product types and each local warehouse j, $j \in \{1, \dots, J\}$, as:

$$\mathbf{d}_{j,t} := \begin{pmatrix} d^1_{j,t} \\ d^2_{j,t} \\ \vdots \\ d^I_{j,t} \end{pmatrix}. \tag{2.16}$$

Then, as illustrated in Figure 2.3, at each time step t, $t \in \{1, 2, \dots, T\}$, a sequence of events characterizing our system dynamics occurs in the following order [20, 8]:

1. New orders, $\mathbf{x}_{j,t-L_j}$, are received and added to the stocks on hand, $\mathbf{q}_{j,t}$, for every warehouse j, $j \in \{0, \dots, J\}$, and product type i, $i \in \{1, \dots, I\}$.
2. Based on the actual state, $\mathbf{s}_t$, ordering decisions, $\mathbf{a}_t$, are made.
3. Demands, $\mathbf{d}_{j,t}$, are then realized for every local warehouse j, $j \in \{1, \dots, J\}$, and product type i, $i \in \{1, \dots, I\}$, and satisfied using the current stock level, accounting for potential backorders.
4. Cost, r_t, is computed.
5. The next state, $\mathbf{s}_{t+1}$, containing new stocks on hand and in transit is determined.

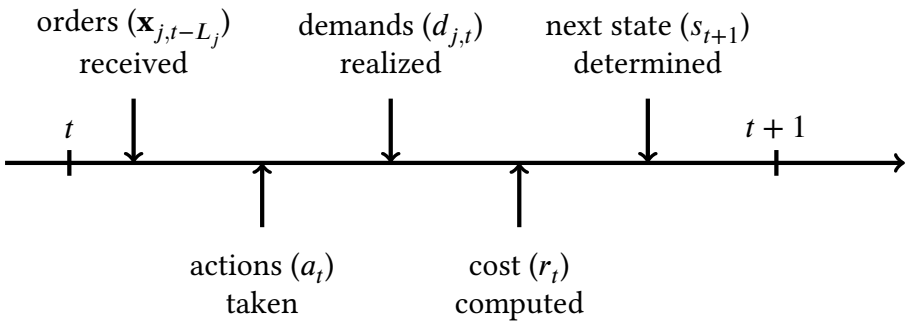


Figure 2.3: Representation of the dynamics of the inventory control system considered.

To determine the next state $\mathbf{s}_{t+1}$, the vector representing the total in-transit orders for each local warehouse is updated by shifting existing values to the right and appending the new ordering decisions $\mathbf{a}_{j,t}$ to the first column on the left:

$$\mathbf{x}_{j,t+1} = \begin{pmatrix} a^1_{j,t} & x^1_{j,t-1} \cdots & x^1_{j,t-L_j+2} \\ a^2_{j,t} & x^2_{j,t-1} \cdots & x^2_{j,t-L_j+2} \\ \vdots & \vdots & \vdots \\ a^I_{j,t} & x^I_{j,t-1} \cdots & x^I_{j,t-L_j+2} \end{pmatrix}. \tag{2.17}$$

Finally, the stocks dynamics as delineated in Equations (2.10) and (2.11) are defined as:

$$\begin{cases} q^i_{0,t+1} = \min\left[\left(q^i_{0,t} + a^i_{0,t} - \sum_{j=1}^{J} a^i_{j,t}\right), c^i_0\right] & \forall\, j \in \{0\},\ \forall\, i \in \{1, \dots, I\}, \\ q^i_{j,t+1} = \min\left[\left(q^i_j + x^i_{j,t-L_j} - d^i_{j,t}\right), c^i_j\right] & \forall\, j \in \{1, \dots, J\},\ \forall\, i \in \{1, \dots, I\}. \end{cases} \tag{2.18}$$

This means that, at the beginning of the next time step $t+1$, the central warehouse stock level equals the initial stocks, plus the units produced, minus the units shipped. Similarly, the local warehouses' stock levels are equal to the initial stocks, plus the units received, minus the current demands. To ensure compliance with Equation (2.8), the system prohibits storing products exceeding storage capacities, directly discarding any units in excess. Clearly, if the stock level is positive, a storage cost is incurred; otherwise, a backorder cost applies.

2.3.2 Deep Reinforcement Learning Formulation

In this section, we explain how we adapt DRL algorithms to fit the previously described MDP formulation.

State Vector

The state vector defined in Section 2.3.1 contains the current stocks on hand and in transit for each warehouse and product type. In addition, we also include the demand values from the last τ time steps, defining it as:

$$\left(\mathbf{s}_t, \hat{\mathbf{d}}_{1,t} \dots, \hat{\mathbf{d}}_{J,t}\right), \tag{2.19}$$

where:

$$\hat{\mathbf{d}}_{j,t} := \begin{pmatrix} d^1_{j,t-\tau} & d^1_{j,t-\tau+1} \cdots & d^1_{j,t-1} \\ d^2_{j,t-\tau} & d^2_{j,t-\tau+1} \cdots & d^2_{j,t-1} \\ \vdots & \vdots & \vdots \\ d^I_{j,t-\tau} & d^I_{j,t-\tau+1} \cdots & d^I_{j,t-1} \end{pmatrix}. \tag{2.20}$$

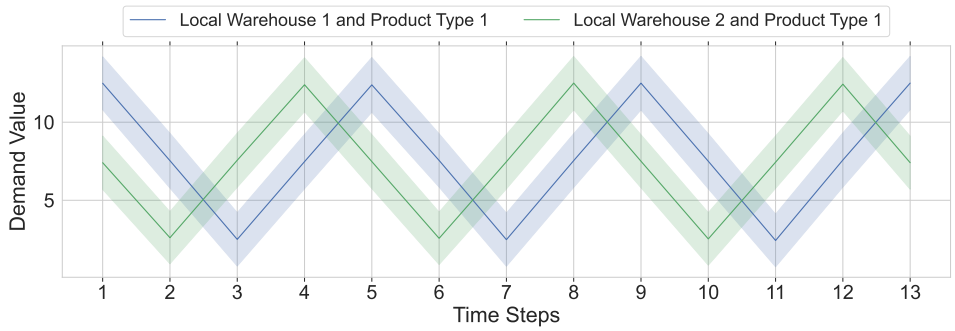
It is crucial to highlight that the size of the newly defined state vector is $I + IJ(L+\tau+1)$.

To simulate a stochastic and seasonal behavior, drawing inspiration from Kemmer et al. [24], Peng et al. [34], and Stranieri and Stella [44], we represent the demand as a sinusoidal function augmented with a stochastic component, as defined by the following equation:

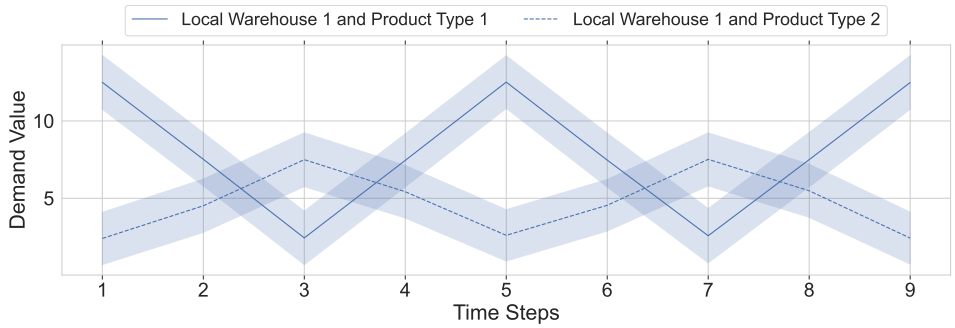
$$d^i_{j,t} = \left\lfloor \frac{d^i_{max}}{2} \left[1 + \sin\left(\frac{\pi(2i+j+t)}{2}\right)\right] + \mathscr{U}\left(0, d^i_{var}\right) \right\rfloor, \tag{2.21}$$

where $\lfloor \cdot \rfloor$ denotes the floor function, d^i_{max} is the maximum demand value for each product type i, and $\mathscr{U}$ is a uniformly distributed random variable on the support $(0, d^i_{var})$ representing demand variations (i.e., the *uncertainty*). At each time step, the demand can vary for each local warehouse and product type while maintaining the same general behavior, as depicted in Figure 2.4.

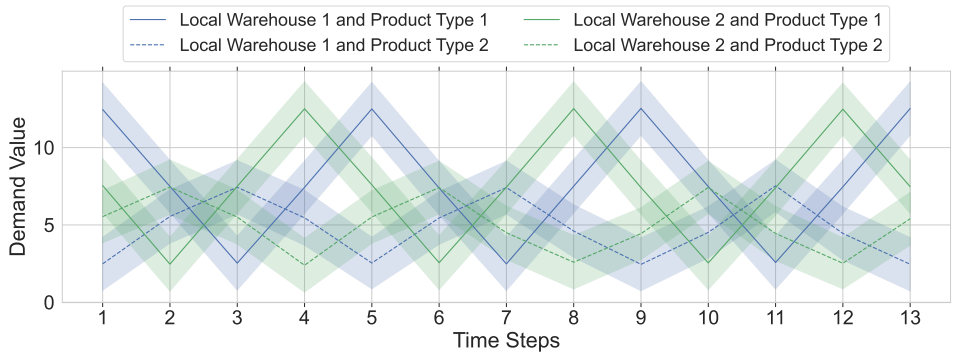
To provide DRL agents with limited knowledge about the demand history, we include the most recent demand values within the state vector. This approach benefits the agent in developing a basic understanding of demand fluctuations. It is important to note that



(a) $I = 1$ and $J = 2$ with $L = 3$.



(b) $I = 2$ and $J = 1$ with $L = 1$.



(c) $I = 2$ and $J = 2$ with $L = 3$.

Figure 2.4: Different demands behaviors generated according to Section 2.3.2 fixing $d^1_{max} = 10$, $d^1_{var} = 5$ and $d^2_{max} = 5$, $d^2_{var} = 10$ for distinct typologies and configurations of the SCIM problem. Specifically, a represents one product type and two local warehouses with lead times of three; b shows two product types and one local warehouse with a lead time of one; and c depicts two product types and two local warehouses with lead times of three.

the actual demand d_t will not be included within the state vector $\mathbf{s}_t$ until the subsequent time step $t + 1$. This design choice enables DRL agents to benefit from learning demand patterns, integrating a sort of *demand forecasting* directly into their policy.

Action Vector

Regarding the action vector $\mathbf{a}_t$ represented in Section 2.3.1, we opted for a *continuous action space*, which means that the ANN generates action values directly, as it scales linearly with the size of the problem. Specifically, each value $\mathbf{a}_{j,t}$ represents the number of units produced at the factory and the number of units shipped to each local warehouse for each product type.

In practical terms, a relatively small and identical upper bound is commonly adopted for all action values to minimize computational effort. However, this method can lead to a significant drop in terms of performance. Indeed, if the upper bound is too small, the agent may choose an inefficient action since the optimal one lies outside the admissible range. On the other hand, if the upper bound is too high, the agent may repeatedly select an incoherent action that falls within the admissible range but exceeds a specified maximum capacity, thus slowing down the training process. Accordingly, our implementation provides a continuous action space with *independent bounds* for each action value, ensuring a balance between computational efficiency and DRL agents' performance.

Consequently, we set the lower bound for each value at zero, as producing or shipping negative quantities of products would be not allowed. Conversely, the upper bound for each warehouse corresponds to its maximum capacity for each product type. Formally, these bounds are represented as $0 \leq a^i_{j,t} \leq c^i_j$, reflecting our assumption of a specific storage capacity for each warehouse with respect to each product type. Accordingly, we have a clearly defined and coherent action space that adheres to the constraint described in Equation (2.7). This approach is expected to enhance training times and performance since the action space is bounded (and therefore limited) but contains only coherent actions.

Balanced Allocation Rule

When the stocks on hand in the central warehouse are insufficient to satisfy orders from all local warehouses, the central warehouse must allocate the available stocks according to a specific decision rule. In the work of Kaynov et al. [23], two different allocation rules for a multi-discrete action space were proposed for selecting feasible actions. The first one is the proportional allocation. However, this does not provide the DRL agents with an incentive during training to determine feasible and effective actions. The second one is the sequential allocation rule, which randomizes the sequence of the local warehouses at every time step and executes the determined actions sequentially. Nevertheless, following this rule, the last local warehouses in the sequence could receive an amount of product close to zero, making it harder to recover from possible backorders.

To address this issue and respect the constraint detailed in Equation (2.9), we propose a new *balanced allocation rule* for continuous action spaces. This rule ensures a fair distribution of products among the local warehouses while maintaining the stocks on

hand in the central warehouse as non-negative. Please note that the proposed allocation rule should also be applicable to multi-discrete action spaces.

As described in Algorithm 1, for each product type, if the total number of products to ship exceeds the available products at the central warehouse, proceed with the following steps:

1. Randomly select one local warehouse.
2. If the determined quantity to ship to the selected local warehouse is greater than zero, then decrease this quantity by one.
3. Repeat steps 1 and 2 until the total number of products to ship reaches the available products at the central warehouse.

Algorithm 1 The proposed balanced allocation rule for continuous action spaces.

for all $i \in \{1, \dots, I\}$ **do**
  if $\sum_{j=1}^{J} a_{j,t}^{i} > q_{0,t}^{i}$ **then**
    while $\sum_{j=1}^{J} a_{j,t}^{i} > q_{0,t}^{i}$ **do**
      $x \leftarrow \mathcal{U}(1, J)$
      if $a_{x,t}^{i} > 0$ **then**
        $a_{x,t}^{i} \leftarrow a_{x,t}^{i} - 1$

This allocation rule guarantees that the central warehouse does not ship more products than those available while maintaining a balanced distribution among local warehouses and preventing the issue of some warehouses receiving close to zero products. It is important to note that DRL agents do not explicitly consider on-hand stocks when selecting an action. However, due to Equation (2.18), the system constraint expressed in Equation (2.8) is satisfied, and producing or shipping more products than those that can be stored leads to a cost, resulting in an *implicit penalty* for the agents. An alternative formulation involves considering the action masking technique to prevent agents from taking invalid actions, as proposed for a discrete action space by Peng et al. [34].

Reward Function and System Dynamics

The *reward function* for each time step t is detailed in Section 2.3.1. Also, the state update rule follows the system dynamics of the MDP presented in Section 2.3.1. It is worth highlighting that demand values within the state vector are also updated, discarding the oldest value and appending the most recent one.

2.4 Numerical Experiments

In this section, we conduct a numerical analysis to assess the performance of various agents. First, we detail the parameters of the environment, followed by an explanation

of the DRL algorithms' hyperparameters and static inventory policies' parameters. Then, we present and discuss the results.

2.4.1 Environment Parameters

We designed and conducted a comprehensive set of *numerical experiments* to compare the performances of DRL algorithms and static inventory policies under two distinct scenarios, specifically with two different values of replenishment lead times (i.e., $L = \{1, 2\}$). To thoroughly assess the adaptability and robustness of DRL algorithms, each scenario encompasses a varying number of product types (i.e., $I = \{1, 2\}$) and local warehouses (i.e., $J = \{1, 2, 3\}$), resulting in a total of 12 experiments. Moreover, each scenario is associated with its specific environment parameters, as detailed in Table 2.2. It is crucial to highlight that the capacity of the central warehouse is adjusted based on the number of local warehouses to ensure it can fully *absorb* the demand for each product type. In detail, it is set to 15 when there is one local warehouse, 25 for two local warehouses, and 35 for three, respectively. In the authors' opinion, this experimental setup facilitated an extensive evaluation of DRL algorithms and static inventory policies under different conditions, providing valuable insights into their effectiveness and adaptability.

Table 2.2: The environment parameters associated with the experiments conducted. For episode duration, the first value pertains to the experiments with a lead time of one, while the second is for the experiments with lead times of three. Concerning backorder costs, the two values represent experiments with one and two product types, respectively. Finally, for maximum demand value and variation, the first value corresponds to the first product type, while the second value relates to the second product type.

Parameter	Values
Episode Horizon	{9, 13}
Maximum Demand Value	{10, 5}
Maximum Demand Variation	{5, 10}
Production Cost	1
Transportation Cost	0.05
Storage Capacity Local Warehouses	15
Storage Cost Central Warehouse	0.01
Storage Cost Local Warehouses	0.1
Backorder Cost	{10, 20}

For the first product type, the demand value is upper-bounded by 10 units. However, at each time step, it can stochastically increase by up to 5 additional units, leading to a maximum potential demand of 15 units. For the second product type, these values are inverted with an upper bound of 5 units and a potential increase of up to 10 units.

Each local warehouse has a capacity of 15 units and can store stocks as needed at a unit storage cost of 0.01 per time step. Similarly, the cost of storing each unit at the central warehouse is 0.1 per time step. The cost to produce a single product unit is set at 1. For every time step and every unit of unsatisfied demand, the backorder cost is 10 with one product type and 20 with two. Finally, the transportation cost from the central warehouse to a local one is 0.03 per unit. It is worth noting that we focus on a *symmetrical environment*, which means that all local warehouses share identical values for the specified parameters. This simplification allows for a more precise comparison of the performance of different agents while still maintaining a level of complexity that is representative of real-world supply chain systems.

For each episode, we selected different time horizons, T, based on lead times, L. Specifically, we evaluated seven time steps. To account for the lead times effect, we added L time steps at the beginning of each episode, during which the demand is set to zero, as this demand cannot be satisfied. Similarly, we added L time steps at the end of the episode, in which the value of the actions is set to zero. We made this decision because the corresponding orders for the L time steps after T will never arrive, and the optimal actions would inherently be zero, providing no additional information for evaluation purposes. This method enables us to consider the lead time effects while ensuring a proper evaluation of the agents.

2.4.2 Implemented Algorithms

After specifying the environment parameters, we implemented the DRL agents using three different state-of-the-art DRL algorithms, namely A3C, PG, and PPO, which were briefly introduced in Section 2.2. For our implementation, we relied on Ray [31], an open-source Python framework that includes RLib, a scalable RL library, and Tune, a scalable hyperparameter tuning library. A significant advantage of Ray is its native support for the OpenAI Gym APIs [6] that we utilized to develop the *simulator* representative of the environment and used for the agents' training process. We chose the hyperparameters for the DRL algorithms to tune based on the Ray documentation and discussions in the papers of Alves and Mateus [1], Gijsbrechts et al. [16], and Stranieri and Stella [44], as reported in Table A.1 of Appendix A.1.

To evaluate and compare the performance of the DRL algorithms, we implemented two distinct static inventory policies, the well-known base-stock policy (referred to as the BS policy) and the (s, Q)-policy (referred to as the sQ policy). The BS policy requires ordering up to S units whenever the stock level falls below this threshold. In contrast, the sQ policy involves ordering Q units whenever the stock level drops below s.

In our implementation, we opted to make reordering decisions independently, meaning that the policy parameter values (i.e., S_j^i for the BS policy and s_j^i and Q_j^i for the sQ policy) can differ based on the specific warehouse and product type. Despite this flexibility, these policies are still considered *static* because their values remain constant over time. To determine the optimal parameter values that minimize Equation (2.6),

we adopted a data-driven method using Bayesian optimization. This method does not necessitate any assumptions or simplifications. As a result, it can be applied to any supply chain typology or configuration, sharing the same simulator as the DRL algorithms. It is crucial to note that our experimental setup employs data-driven approaches that interact with the simulator under the assumption that they have no access to *additional information*, such as forecasts.

We also implemented an additional model called EVP (an acronym that means "expected value problem"). In the EVP model, the demand variables $d^i_{j,t}$ in Equation (2.6) are replaced at each time step t with the *expected demand value* for each product type i and local warehouse j. To compare DRL agents with static inventory policies, we finally implemented an oracle, which consists of a baseline that knows at each time step t the *exact demand value* $d^i_{j,t}$ in Equation (2.6) for each product type i and local warehouse j. As a result, it can select the optimal actions to take a priori (for this reason, we refer to this model as PI, which stands for "perfect information"). It is important to note that the formulation for the EVP and PI models remains the same as expressed in Equations (2.6) to (2.14). However, when the EVP agent takes an action and solves the problem, it replaces the demand at time step t with its expected value (rounded up) over the episode horizon T. In contrast, in the PI model, the agent possesses additional information and knows the exact demand realization at time step t, which leads it to take optimal actions at each time step t.

2.4.3 Results

To compare the performance of different agents (i.e., DRL algorithms and static inventory policies), we simulated 1000 testing episodes for each experiment, reporting the expected value of perfect information (*EVPI gap*) achieved (calculated as the average of the difference between the profits of a specific agent and the profits of the oracle, divided by the profits of the oracle). This comparison highlights the effectiveness of the agents as opposed to an *optimal solution*, which remains not feasible in real-world deployments because the PI model assumes complete knowledge of future demand realizations.

All experiments were run on a machine equipped with an Apple M2 Pro chip with 10 cores and 16 GB of RAM, ensuring consistent computing resources across all tests. By evaluating the performance of DRL algorithms and static inventory policies under diverse conditions, we can gain valuable insights into their effectiveness, adaptability, and potential for application in different supply chain typologies and configurations.

One Product Type

The results of numerical experiments under one product type are summarized in Figure 2.5.

In the first experiment conducted within the context of a single local warehouse with lead times of one, the PPO and A3C algorithms significantly outperform other agents,

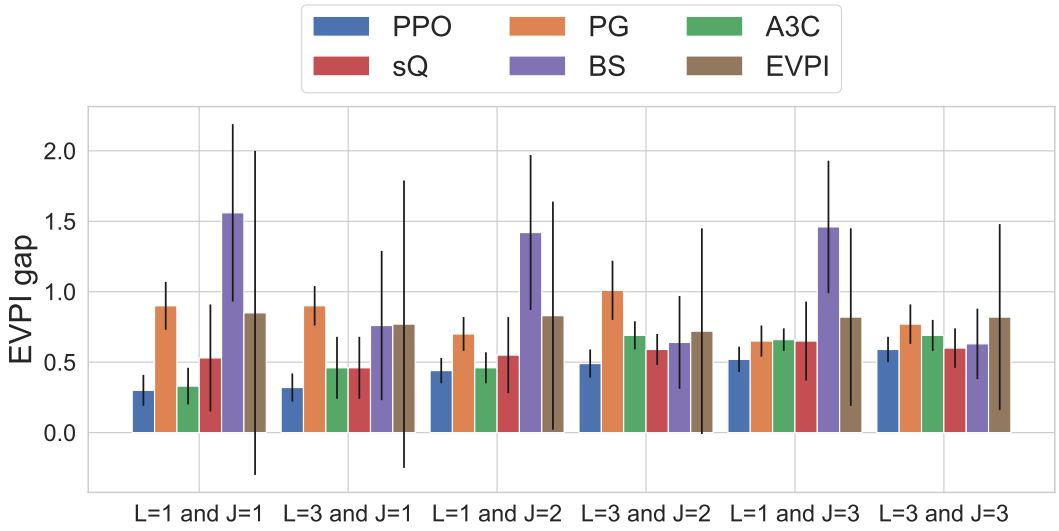


Figure 2.5: EVPI gap, grouped by experiment, for the experimental setup with one product type. For each number of local warehouses and length of lead times, the figure compares the performance of DRL algorithms, static inventory policies, and EVP. The lower the gap, the better the agent.

with the BS policy performing poorly even when compared to the EVP model. When the lead times increase to three, the performance of A3C and the sQ policy become more balanced, while PPO continues to demonstrate superior results, and both the BS policy, PG, and EVP underperform.

With two local warehouses and lead times of one, PPO still performs better than all other agents, while the performance of the sQ policy falls between that of the A3C and PG algorithms. Meanwhile, the BS policy performs worse than EVP. Extending lead times to three further accentuates the excellent performance of PPO, with the sQ policy following closely behind, while the BS policy slightly surpasses both A3C and PG. In this experiment, static inventory policies tend to perform marginally better than A3C and PG, with the former having an advantage over the latter.

Upon introducing a third local warehouse to the experimental setup, PPO consistently maintains its superior level of performance compared to other agents, albeit in a less marked manner. With $L = 1$, the sQ policy achieves results comparable to PG and A3C, while the BS policy continues to perform behind EVP. Conversely, when $L = 3$, static inventory policies slightly outperform both A3C and PG.

Two Product Types

Figure 2.6 summarizes the results under two product types.

In the experiment with a single local warehouse and $L = 1$, the PPO algorithm demonstrates a clear advantage over other agents. A3C and PG both surpass the BS

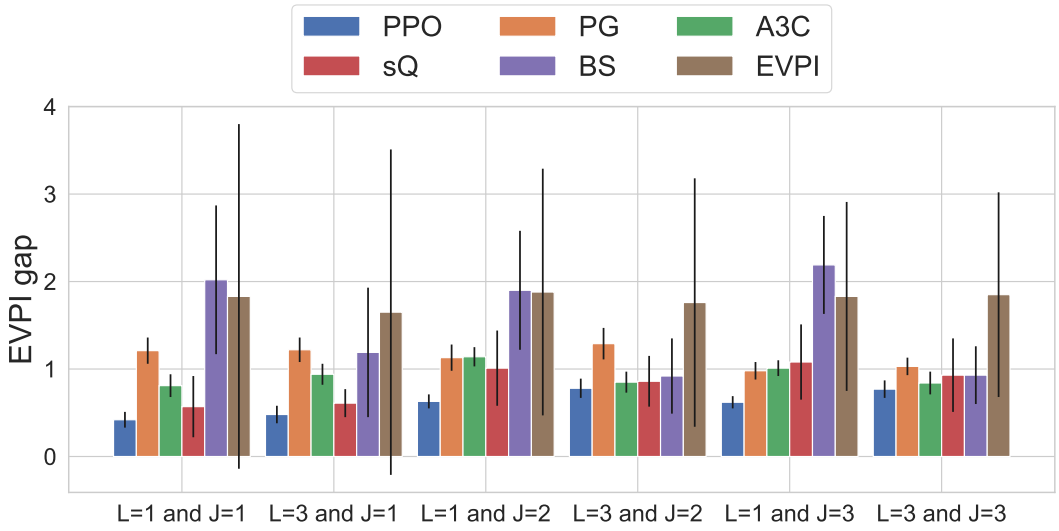


Figure 2.6: EVPI gap, grouped by experiment, for the experimental setup with two product types. For each number of local warehouses and length of lead times, the figure compares the performance of DRL algorithms, static inventory policies, and EVP. The lower the gap, the better the agent.

policy, whereas the sQ policy performance closely aligns with PPO. When $L = 3$, the overall relative performance essentially persists, although the BS policy now outperforms the EVP agent.

When the experimental setup includes two local warehouses and lead times of one, PPO still maintains its advantage over others, followed by the sQ policy, while A3C and PG performance become more comparable in this experiment. Extending the lead times to three reveals a substantial equilibrium in performance across all agents except for PG, which slightly underperforms.

Lastly, in scenarios with three local warehouses, DRL algorithms surpass the sQ policy when $L = 1$, although the performance gap with PPO remains consistent, and the BS policy encounters difficulty. When $L = 3$, static inventory policies achieve results comparable to DRL algorithms, with PPO still outperforming others.

2.5 Discussion and Insights

In summary, the effectiveness of various DRL algorithms and static inventory policies in SCIM depends on factors such as the number of product types and local warehouses and the length of lead times. PPO consistently emerges as the *best-performing algorithm* across all conducted experiments, offering a robust approach for addressing the SCIM problem within the designed experimental setup. Although the sQ policy does not achieve the same level of performance as PPO, it still performs commendably and

appears to be the *second-best choice* among the evaluated algorithms.

Specifically, for a single product type and lead times set to one, the A3C algorithm is preferable to PG, particularly in experiments with one and two local warehouses. Meanwhile, the sQ policy shows superior results compared to the BS policy, which consistently underperforms. Conversely, by extending lead times to three, performance appears to balance as the number of local warehouses increases.

In experiments involving two product types, the performance gap between various agents remains consistent. Typically, when reducing the number of local warehouses and decreasing lead times, A3C – and especially PG – tend to perform worse than the sQ policy, with the BS policy showing markedly poor performance with lead times set to one. Contrarily, as the number of local warehouses and lead times increase, static inventory policies gradually align with the results of DRL algorithms.

Moreover, DRL algorithms consistently and significantly outperform the EVP model, suggesting that the complexities of the problem cannot be adequately captured by merely considering the average demand value. More robust strategies are thus essential in these scenarios. These findings underscore the articulate interplay between the number of product types and local warehouses, the length of lead times, and the relative performance of DRL algorithms versus static inventory policies. Decision-makers should carefully consider these factors when choosing an *appropriate agent* that balances performance and computational efficiency.

It is crucial to emphasize that all agents reach *convergence* in each experiment conducted, as documented in Figure A.1 of Appendix A.1. Specifically, PG tends to converge faster than other DRL agents, leading to reduced training times. In contrast, the A3C algorithm usually takes longer to converge, resulting in extended training durations, while PPO holds an intermediate position in terms of convergence and training times.

In the context of static inventory policies, the sQ policy requires more training time to converge than the BS policy. This difference can be attributed to the increased number of parameters to optimize inherent to the sQ policy. Despite the extended training duration, the sQ policy generally performs better, highlighting the trade-offs that decision-makers need to consider when selecting an appropriate agent based on factors such as *convergence, training time, and performance.*

Ultimately, the training times between DRL algorithms and static inventory policies are reasonably similar. For completeness, Table A.2 of Appendix A.2 provides the *average training times* for each agent.

2.5.1 Conclusion and Future Research

In this study, we presented an MDP formulation for a two-echelon inventory control system, demonstrating that the PPO algorithm consistently arises as the best-performing algorithm across various scenarios. While there is a trade-off between implementation complexity and performance, its results remain consistent across all the experiments

we analyzed. Additionally, we highlighted the sQ policy as a robust alternative to DRL algorithms such as A3C and PG, offering a balance between performance and computational efficiency.

Building on these findings, several potential avenues for *future research* can be explored. For example, our system could benefit from incorporating alternative allocation rules for managing the on-hand stocks of the central warehouse. While we have introduced a balanced rule, we have not compared it to others, such as a priority rule that gives priority to local warehouses with the highest backorder cost in an asymmetrical environment. We reserve this topic for further investigation in future studies.

Another two areas for exploration in the context of multiple product types are the substitution effect and perishable products. Currently, there is limited literature on multi-echelon inventory control systems that incorporate these specific assumptions, and expanding research in this direction could significantly enhance our understanding of SCIM strategies in more challenging and practical environments.

Lastly, investigating machine learning techniques for demand forecasting presents a promising avenue. The idea is to identify the best technique through a benchmark and then use it to evaluate dynamic inventory policies against static ones. Moreover, such a forecast could be integrated into the state vector of the DRL agents. While preliminary studies have investigated this approach [12], a comparative analysis among different DRL algorithms has not yet been addressed and represents a promising focus for future research.

References

[1] Júlio César Alves and Geraldo Robson Mateus. "Deep Reinforcement Learning and Optimization Approach for Multi-echelon Supply Chain with Uncertain Demands." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 584–599. DOI: 10.1007/978-3-030-59747-4_38. URL: https://doi.org/10.1007/978-3-030-59747-4_38.

[2] Sven Axsäter and Lars Juntti. "Comparison of echelon stock and installation stock policies for two-level inventory systems." In: *International Journal of Production Economics* 45.1–3 (Aug. 1996), pp. 303–310. ISSN: 0925-5273. DOI: 10.1016/0925-5273(95)00121-2. URL: http://dx.doi.org/10.1016/0925-5273(95)00121-2.

[3] Sven Axsäter and Kaj Rosling. "Notes: Installation vs. Echelon Stock Policies for Multilevel Inventory Control." In: *Management Science* 39.10 (Oct. 1993), pp. 1274–1280. ISSN: 1526-5501. DOI: 10.1287/mnsc.39.10.1274. URL: http://dx.doi.org/10.1287/mnsc.39.10.1274.

[4] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[5] Nabila Bouti, Ibrahim Boukallal, and Fatima El Khoukhi. "Optimization of Inventory Management: A Literature Review." In: *Digital Technologies and Applications*. Springer Nature Switzerland, 2023, pp. 923–933. ISBN: 9783031298608. DOI: 10.1007/978-3-031-29860-8_92. URL: http://dx.doi.org/10.1007/978-3-031-29860-8_92.

[6] Greg Brockman et al. *OpenAI Gym*. 2016. DOI: 10.48550/ARXIV.1606.01540. URL: https://arxiv.org/abs/1606.01540.

[7] S. Kamal Chaharsooghi, Jafar Heydari, and S. Hessameddin Zegordi. "A reinforcement learning model for supply chain ordering management: An application to the beer game." In: *Decision Support Systems* 45.4 (Nov. 2008), pp. 949–959. DOI: 10.1016/j.dss.2008.03.007. URL: https://doi.org/10.1016/j.dss.2008.03.007.

[8] Xiuli Chao et al. "Approximation Algorithms for Perishable Inventory Systems." In: *Operations Research* 63.3 (June 2015), pp. 585–601. DOI: 10.1287/opre.2015.1386. URL: https://doi.org/10.1287/opre.2015.1386.

[9] Fangruo Chen. "Optimal Policies for Multi-Echelon Inventory Problems with Batch Ordering." In: *Operations Research* 48.3 (June 2000), pp. 376–389. ISSN: 1526-5463. DOI: 10.1287/opre.48.3.376.12427. URL: http://dx.doi.org/10.1287/opre.48.3.376.12427.

[10] Andrew J. Clark and Herbert Scarf. "Optimal Policies for a Multi-Echelon Inventory Problem." In: *Management Science* 6.4 (July 1960), pp. 475–490. ISSN: 1526-5501. DOI: 10.1287/mnsc.6.4.475. URL: http://dx.doi.org/10.1287/mnsc.6.4.475.

[11] Marc A. De Bodt and Stephen C. Graves. "Continuous-Review Policies for a Multi-Echelon Inventory Problem with Stochastic Demand." In: *Management Science* 31.10 (Oct. 1985), pp. 1286–1299. ISSN: 1526-5501. DOI: 10.1287/mnsc.31.10.1286. URL: http://dx.doi.org/10.1287/mnsc.31.10.1286.

[12] Henri Dehaybe, Daniele Catanzaro, and Philippe Chevalier. "Deep Reinforcement Learning for inventory optimization with non-stationary uncertain demand." In: *European Journal of Operational Research* (Oct. 2023). DOI: 10.1016/j.ejor.2023.10.007. URL: https://doi.org/10.1016/j.ejor.2023.10.007.

[13] Rolf Forsberg. "Exact evaluation of (R, Q)-policies for two-level inventory systems with Poisson demand." In: *European Journal of Operational Research* 96.1 (Jan. 1997), pp. 130–138. ISSN: 0377-2217. DOI: 10.1016/s0377-2217(96)00137-3. URL: http://dx.doi.org/10.1016/S0377-2217(96)00137-3.

[14] Vincent François-Lavet et al. "An Introduction to Deep Reinforcement Learning." In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. DOI: 10.1561/2200000071. URL: https://doi.org/10.1561/2200000071.

[15] Kevin Geevers, Lotte van Hezewijk, and Martijn R. K. Mes. "Multi-echelon inventory optimization using deep reinforcement learning." In: *Central European Journal of Operations Research* (July 2023). ISSN: 1613-9178. DOI: 10.1007/s10100-023-00872-2. URL: http://dx.doi.org/10.1007/s10100-023-00872-2.

[16] Joren Gijsbrechts et al. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems." In: *Manufacturing & Service Operations Management* 24.3 (May 2022), pp. 1349–1368. DOI: 10.1287/msom.2021.1064. URL: https://doi.org/10.1287/msom.2021.1064.

[17] Geoffrey J Gordon. *Approximate solutions to Markov decision processes*. Carnegie Mellon University, 1999.

[18] Lotte van Hezewijk et al. "Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem." In: *International Journal of Production Research* 61.6 (Apr. 2022), pp. 1955–1978. ISSN: 1366-588X. DOI: 10.1080/00207543.2022.2056540. URL: http://dx.doi.org/10.1080/00207543.2022.2056540.

[19] Christian D. Hubbs et al. *OR-Gym: A Reinforcement Learning Library for Operations Research Problems*. 2020. DOI: 10.48550/ARXIV.2008.06319. URL: https://arxiv.org/abs/2008.06319.

[20] Woonghee Tim Huh et al. "Asymptotic Optimality of Order-Up-To Policies in Lost Sales Inventory Systems." In: *Management Science* 55.3 (Mar. 2009), pp. 404–420. DOI: 10.1287/mnsc.1080.0945. URL: https://doi.org/10.1287/mnsc.1080.0945.

[21] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms." In: *Neural Computation* 6.6 (Nov. 1994), pp. 1185–1201. DOI: 10.1162/neco.1994.6.6.1185. URL: https://doi.org/10.1162/neco.1994.6.6.1185.

[22] Ilya Jackson, Maria Jesus Saenz, and Dmitry Ivanov. "From natural language to simulations: applying AI to automate simulation modelling of logistics systems." In: *International Journal of Production Research* (Nov. 2023), pp. 1–24. ISSN: 1366-588X. DOI: 10.1080/00207543.2023.2276811. URL: http://dx.doi.org/10.1080/00207543.2023.2276811.

[23] Illya Kaynov et al. "Deep Reinforcement Learning for One-Warehouse Multi-Retailer inventory management." In: *International Journal of Production Economics* 267 (Jan. 2024), p. 109088. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109088. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109088.

[24] Lukas Kemmer et al. "Reinforcement learning for supply chain optimization." In: *European Workshop on Reinforcement Learning*. Vol. 14. 2018.

[25] Ton de Kok et al. "A typology and literature review on stochastic multi-echelon inventory models." In: *European Journal of Operational Research* 269.3 (Sept. 2018), pp. 955–983. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.02.047. URL: http://dx.doi.org/10.1016/j.ejor.2018.02.047.

[26] Chaaben Kouki, Joachim Arts, and M. Zied Babai. "Performance evaluation of a two-echelon inventory system with network lost sales." In: *European Journal of Operational Research* 314.2 (Apr. 2024), pp. 647–664. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.10.009. URL: http://dx.doi.org/10.1016/j.ejor.2023.10.009.

[27] H.T. Lee and J.C. Wu. "A study on inventory replenishment policies in a two-echelon supply chain system." In: *Computers & Industrial Engineering* 51.2 (Oct. 2006), pp. 257–263. ISSN: 0360-8352. DOI: 10.1016/j.cie.2006.01.005. URL: http://dx.doi.org/10.1016/j.cie.2006.01.005.

[28] X. Liu and Y. L. Tu. "Capacitated production planning with outsourcing in an OKP company." In: *International Journal of Production Research* 46.20 (Jan. 2008), pp. 5781–5795. DOI: 10.1080/00207540701348779. URL: https://doi.org/10.1080/00207540701348779.

[29] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1928–1937.

[30] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[31] Philipp Moritz et al. "Ray: A Distributed Framework for Emerging AI Applications." In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 561–577. ISBN: 978-1-939133-08-3. URL: https://www.usenix.org/conference/osdi18/presentation/moritz.

[32] Despoina Ntio et al. "A Simulation-Based Inventory Management Approach for Divergent Multi-echelon Systems." In: *Advances in Applied Macroeconomics*. Springer Nature Switzerland, 2025, pp. 441–450. ISBN: 9783031766589. DOI: 10.1007/978-3-031-76658-9_23. URL: http://dx.doi.org/10.1007/978-3-031-76658-9_23.

[33] Afshin Oroojlooyjadid et al. "A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization." In: *Manufacturing & Service Operations Management* 24.1 (Jan. 2022), pp. 285–304. DOI: 10.1287/msom.2020.0939. URL: https://doi.org/10.1287/msom.2020.0939.

[34] Zedong Peng et al. "Deep Reinforcement Learning Approach for Capacitated Supply Chain optimization under Demand Uncertainty." In: *2019 Chinese Automation Congress (CAC)*. IEEE, Nov. 2019. DOI: 10.1109/cac48633.2019.8997498. URL: https://doi.org/10.1109/cac48633.2019.8997498.

[35] Sushil Punia et al. "Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail." In: *International Journal of Production Research* 58.16 (Mar. 2020), pp. 4964–4979. ISSN: 1366-588X. DOI: 10.1080/00207543.2020.1735666. URL: http://dx.doi.org/10.1080/00207543.2020.1735666.

[36] Benjamin Rolf et al. "A review on reinforcement learning algorithms and applications in supply chain management." In: *International Journal of Production Research* 61.20 (Nov. 2022), pp. 7151–7179. ISSN: 1366-588X. DOI: 10.1080/00207543.2022.2140221. URL: http://dx.doi.org/10.1080/00207543.2022.2140221.

[37] Herbert Scarf. "The Optimality of (S, s) Policies in the Dynamic Inventory Problem." In: *Herbert Scarf's Contributions to Economics, Game Theory and Operations Research*. Palgrave Macmillan UK, 2005, pp. 1–7. ISBN: 9781137024381. DOI: 10.1057/9781137024381_1. URL: http://dx.doi.org/10.1057/9781137024381_1.

[38] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[39] John Schulman et al. "Trust Region Policy Optimization." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html.

[40] Md Shajalal, Petr Hajek, and Mohammad Zoynul Abedin. "Product backorder prediction using deep neural network on imbalanced data." In: *International Journal of Production Research* 61.1 (Mar. 2021), pp. 302–319. ISSN: 1366-588X. DOI: 10.1080/00207543.2021.1901153. URL: http://dx.doi.org/10.1080/00207543.2021.1901153.

[41] Rohit Sharma et al. "The role of artificial intelligence in supply chain management: mapping the territory." In: *International Journal of Production Research* 60.24 (Feb. 2022), pp. 7527–7550. ISSN: 1366-588X. DOI: 10.1080/00207543.2022.2029611. URL: http://dx.doi.org/10.1080/00207543.2022.2029611.

[42] David Silver et al. "Mastering the game of Go without human knowledge." In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. DOI: 10.1038/nature24270. URL: https://doi.org/10.1038/nature24270.

[43] Francesco Stranieri, Edoardo Fadda, and Fabio Stella. "Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem." In: *International Journal of Production Economics* 268 (Feb. 2024), p. 109099. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109099. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109099.

[44] Francesco Stranieri and Fabio Stella. "Comparing Deep Reinforcement Learning Algorithms in Two-Echelon Supply Chains." In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer Nature Switzerland, 2025, pp. 454–469. ISBN: 9783031746406. DOI: 10.1007/978-3-031-74640-6_37. URL: http://dx.doi.org/10.1007/978-3-031-74640-6_37.

[45] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039246.

[46] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." In: *Nature* 575.7782 (Oct. 2019), pp. 350–354. DOI: 10.1038/s41586-019-1724-z. URL: https://doi.org/10.1038/s41586-019-1724-z.

[47] Hao Wang et al. "Dynamic inventory replenishment strategy for aerospace manufacturing supply chain: combining reinforcement learning and multi-agent simulation." In: *International Journal of Production Research* 60.13 (Jan. 2022), pp. 4117–4136. ISSN: 1366-588X. DOI: 10.1080/00207543.2021.2020927. URL: http://dx.doi.org/10.1080/00207543.2021.2020927.

[48] Ronald J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine Learning* 8.3-4 (May 1992), pp. 229–256. DOI: 10.1007/bf00992696. URL: https://doi.org/10.1007/bf00992696.

[49] Cathy Wu et al. *Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines*. 2018. DOI: 10.48550/ARXIV.1803.07246. URL: https://arxiv.org/abs/1803.07246.

[50] Yimo Yan et al. "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities." In: *Transportation Research Part E: Logistics and Transportation Review* 162 (June 2022), p. 102712. ISSN: 1366-5545. DOI: 10.1016/j.tre.2022.102712. URL: http://dx.doi.org/10.1016/j.tre.2022.102712.

Chapter 3

Combining Deep Reinforcement Learning and Multi-Stage Stochastic Programming to address the Supply Chain Inventory Management problem

Francesco Stranieri[a,b], Edoardo Fadda[c], and Fabio Stella[a]

[a]Department of Informatics, Systems, and Communication (DISCo), University of Milano-Bicocca, Viale Sarca, 336, Milan, 20126, Italy
[b]Department of Control and Computer Engineering (DAUIN), Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy
[c]Department of Mathematical Sciences (DISMA), Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy

This is the original manuscript of an article published by Elsevier in the International Journal of Production Economics on 15 November 2023, available at: https://doi.org/10.1016/j.ijpe.2023.109099.

Abstract

We introduce a novel heuristic designed to address the supply chain inventory management problem in the context of a two-echelon divergent supply chain. The proposed heuristic advances the current state-of-the-art by combining deep reinforcement learning with multi-stage stochastic programming. In particular, deep reinforcement learning is employed to determine the number of batches to produce, while multi-stage stochastic programming is applied to make shipping decisions. To support further research, we release a publicly available software environment that simulates a wide range of two-echelon divergent supply chain settings, allowing the manipulation of various parameter values, including those associated with seasonal demands. We then present a comprehensive set of numerical experiments considering constraints on production and warehouse capacities under fixed and variable logistic costs. The results demonstrate that the proposed heuristic significantly and consistently outperforms pure deep reinforcement learning algorithms in minimizing total costs. Moreover, it overcomes several inherent limitations of multi-stage stochastic programming models, thus underscoring its potential advantages in addressing complex supply chain scenarios.

3.1 Introduction

Supply chain inventory management (SCIM) is a critical challenge faced by several companies; it involves making decisions regarding the *number of batches* to produce at the factory and how many of them ship to each distribution warehouse. While higher production levels and inventory stocks allow companies to better meet customers' demands, they come at greater costs. Therefore, the goal of SCIM is to find a trade-off between satisfying customer demands and minimizing supply chain costs while maintaining market competitiveness [11].

Although the SCIM problem is typically formulated as a multi-stage (MS) stochastic, mixed-integer linear problem (MILP), existing models often exhibit certain limitations due to a rapid increase in the number of scenarios to consider, which often prevents their practical application [2]. In particular, when risk factors are characterized by seasonality patterns, the number of stages required leads to mathematical models that cannot be solved with reasonable computational resources or within an acceptable amount of time. This challenge, known as the *"curse of dimensionality,"* underscores the importance of developing efficient heuristics capable of handling large-scale, complex problems without sacrificing solution quality or computational tractability [25].

A potential approach to address these limitations is through the use of reinforcement learning (RL), which, as evidenced by Boute et al. [6], has rarely been applied to the SCIM domain. Recently, deep reinforcement learning (DRL) algorithms have attracted increasing attention and have been employed to tackle the SCIM problem, demonstrating their effectiveness and efficiency [42]. However, several challenges persist [14], including: i) problem-specific properties can be difficult to capture as most DRL algorithms are model-free; ii) performances of DRL algorithms can be highly sensitive due to the complex task of hyperparameter tuning; and iii) significant training time is required as DRL algorithms are computationally intensive.

Furthermore, several critical aspects of the SCIM problem have not yet been effectively addressed in the RL context, for example: i) incorporating a seasonal and stochastic demand, which considerably complicates the determination of actions to be taken; ii) considering production constraints that make it necessary, for instance, to maintain stocks in advance to prevent myopic behavior; and iii) addressing fixed and variable logistic costs (i.e., costs for each vehicle utilized and each unit of item shipped, respectively), which needs proper optimization of production and shipments. Notably, accounting for this final point is crucial not only for economic efficiency but also for reducing the environmental impact of the supply chain and fostering *sustainability*.

Consequently, there is a growing need for more advanced RL approaches that can effectively tackle these challenges and deliver better performance in complex SCIM settings. In an effort to fill these gaps in the existing literature, in this paper we:

- Consider a two-echelon supply chain (i.e., consisting of a factory and multiple distribution warehouses) with a single-item type characterized by seasonal demand,

constraints on production and warehouses' capacity, and logistics with fixed and variable costs.

- Design and develop a SCIM software environment capable of modeling all the aforementioned characteristics. We have made the developed SCIM environment accessible as an open-source library on GitHub [1].
- Propose a novel heuristic that combines DRL algorithms and MS stochastic programming to compute an effective solution for the presented SCIM problem.
- Demonstrate the usefulness of the proposed heuristic through a series of numerical experiment settings (e.g., by varying the uncertainty associated with the demand and the number of distribution warehouses).

The rest of the paper is organized as follows: Section 3.2 provides a literature review on the SCIM problem, including state-of-the-art algorithms used to address it. Section 3.3 is devoted to mathematically introducing the SCIM problem, while Section 3.4 presents the solution techniques implemented in this work. Results from numerical experiments are reported in Section 3.5. Finally, conclusions and potential directions for further research are discussed in Section 3.6.

3.2 Literature Review

The SCIM problem has been tackled through several approaches due to its sequential and stochastic nature, making it one of the most challenging and significant problems in production research. The most prominent solution approaches include multi-stage stochastic programming [24, 30, 21], reordering policies [43, 17], and, more recently, reinforcement learning [28, 22, 16, 35].

MS optimization employs *scenario trees* to represent uncertainty. Despite some applications when demand is stationary [24, 21], its use in settings characterized by seasonal demand has been limited. In fact, MS requires a large number of stages to effectively model seasonality, which leads to an increase in the number of *decision variables*, thus requiring remarkable computation resources to obtain a solution. As a result, much of the existing literature focuses on computationally lighter solution methods, such as reordering policies [32].

More in detail, reordering policies are *decision rules* that determine the number of items (or batches) to produce and ship. One of the earliest rules derived from lot-sizing literature is the *base-stock policy* or (s, S)-policy [41]. According to this rule, if the stock quantity falls below s, an order is placed to bring inventory back up to the base-stock level S. A variation of this rule is the so-called (s, Q)-policy, where $Q = S - s$ [40].

[1] https://github.com/frenkowski/SCIMAI-Gym.

Therefore, under an (s, Q)-policy, whenever the stock level falls below s, an order of quantity Q is placed to replenish the inventory.

Reordering policies pave the way for several other rules. In particular, reordering policies are further generalized in RL, which considers rules encompassing estimations of the future. One of the most common approaches for solving the SCIM problem through RL algorithms is by *Q-learning* [31, 13, 37, 27]. However, upon examining various RL studies, it becomes evident that the implemented Q-tables are typically huge and, thus, unscalable, and this becomes worst when seasonal and stochastic demand is considered. Consequently, since a tabular representation is out of the question, more sophisticated representations may be useful.

One of the most promising techniques that has been successfully applied in several settings is DRL, which combines *deep learning* methods with the classical reinforcement learning paradigm. Despite its success achieved in other domains, to the best of the authors' knowledge, only a few papers have implemented DRL algorithms to address the SCIM problem.

In Hubbs et al. [22], the authors examine a four-echelon supply chain, employing the Proximal Policy Optimization (PPO) algorithm. The study compares the performance of various operations research methods with a DRL approach in two scenarios, one with and one without backorder costs. Numerical experiments prove that PPO outperforms reordering policies in both scenarios.

Using a supply chain structure with ten warehouses and a normal demand distribution, in Gijsbrechts et al. [16], the authors conduct two distinct numerical experiments (adapted from Roy et al. [33]) by applying and tuning the Asynchronous Advantage Actor–Critic (A3C) algorithm. The proposed solution method involves a state-dependent base-stock policy to restrict the action space, with A3C used to select the base-stock level at each timestep. The study shows that A3C achieved performance comparable to state-of-the-art heuristics and other RL algorithms, although its initial hyperparameter tuning remains computationally intensive.

Despite their successes, DRL approaches face several challenges, including a lack of robust convergence properties, strong sensitivity to hyperparameters, high sample complexity, and function approximation errors. These issues can result in poor policies and inaccurate value estimation [15, 20, 19]. To deal with these challenges, researchers have attempted to combine DRL techniques with MILP.

One such approach is presented in Bertsekas [4], which demonstrates that by adding a rough approximation of the value function to the MS objective function, it is possible to enhance the performance of MS significantly.

Another possible solution method is provided by Harsha et al. [19], where the authors propose RL methodologies that compute actions by solving a MILP problem. Here, the objective function combines an immediate reward with a value function estimated by a trained neural network. Interestingly, due to the simple structure of the neural network and the use of ReLU activation functions, the objective function can be expressed as a linear term, thus leading to a linear objective function. The authors found that this

technique outperforms other methods in various realistic settings.

In this work, we adopt a different approach compared to previous studies. Instead of using RL to reduce the myopic behavior of mathematical models, we *decompose* the SCIM problem into two parts. First, we use RL to make production decisions that require considering a long horizon. Second, we employ MS programming for the logistic decisions that need to be made over a shorter horizon. While various techniques exist to address settings with independent and identically distributed (i.i.d.) demand [42], our work delves into more complex settings. In particular, we consider a two-echelon supply chain with seasonal demand, constraints on production, limited warehouse capacities, and fixed and variable logistic costs, making our environment more realistic.

In fact, to the best of the authors' knowledge, the only study in the RL literature assuming seasonal demand within a two-echelon supply chain is Peng et al. [28]. Here, a DRL approach based on the Vanilla Policy Gradient (VPG) algorithm is used. The authors evaluate the VPG performance through numerical experiments on three distinct cases. The findings underscore that the VPG algorithm outperforms the (s, Q)-policy employed as a baseline in all three cases. Nevertheless, despite their setting being similar to ours, they do not consider production constraints, warehouse capacities, and fixed and variable logistic costs, making our work the first to explore such a setting.

3.3 Problem Formulation

In this work, we deal with a *divergent supply chain*, i.e., a supply chain characterized by a single participant in the first echelon and multiple participants in the second echelon. More specifically, we investigate a *single-item-type, two-echelon supply chain* consisting of a single factory at the first echelon and a set $\mathcal{J} = \{1,2, \dots, J\}$ of distribution warehouses at the second echelon, as depicted in Figure 3.1. Despite its apparent simple structure, it aptly describes several settings, such as the fuel industry supply chain in which there is a refinery and multiple points of sale or, more generally, two-echelon supply chains where each item type is managed independently.

In detail, we consider episodes of length T (i.e., the time horizon) subdivided into a given number of equally spaced timesteps, implying a supply chain under *periodic review*. At the beginning of each episode, each warehouse $j = 0, 1, \dots, J$ has an initial number of batches $\bar{I}_{j,0}$. At each timestep t, the agent decides the number of batches to produce, x_t, (which must be lower than the factory production limit $X^{\max}$), paying c_0 for each of them. Batches can either be stored in the factory or shipped to distribution warehouses. In the first case, the agent pays h_0 for each batch and can store up to $I_0^{\max}$ of them. In the second case, the agent ships $z_{j,t}$ batches to the distribution warehouse j. Each distribution warehouse j has a maximum capacity of $I_j^{\max}$, a storage cost of h_j per batch, and a stock level at timestep t equal to $I_{j,t}$.

We assume *infinite delivery capacity*, meaning we can ship any quantity of stocks at the same cost. The logistic cost comprises a fixed price p_j paid for each vehicle used

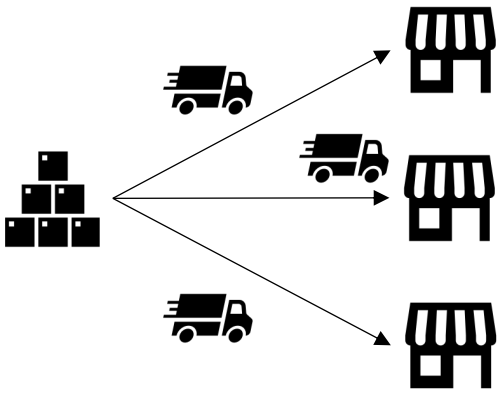


Figure 3.1: A two-echelon supply chain consisting of a factory (first echelon) and three distribution warehouses (second echelon).

to transport batches to warehouse j, irrespective of the truckload, and a variable price l_j dependent on the number of batches shipped. Defining V_j as the capacity of the vehicles going to the distribution warehouse j, $y_{j,t} = \lceil \frac{z_{j,t}}{V_j} \rceil$ represents the associated number of vehicles used for the shipment. Clearly, from a logistic perspective, the goal of the agent is to achieve *full-truckload shipping*. In fact, by fully utilizing the truck's capacity, the agent can optimize vehicle resources, leading to lower shipping costs and increased overall efficiency in the supply chain. This approach helps minimize the number of partially filled trucks, reducing fuel consumption and emissions, as well as the number of expeditions required to satisfy demand. Consequently, it contributes to a more cost-effective and environmentally friendly SCIM strategy.

We consider a *seasonal and stochastic demand*, $d_{j,t}$, that realizes, for each timestep t, at distribution warehouse j. Unsatisfied demand is backordered, meaning that if the demand cannot be satisfied, a penalty cost b_j is incurred until the specific distribution warehouse j is able to satisfy it. Finally, we assume no lead times for production and logistics; thus, inventories are used to make a *seasonal stock* [11]. A summary of the notation is provided in Table 3.1.

The *dynamics of the system* are graphically represented in Figure 3.2 and described by the following sequence of events:

1. Starting from the current state of the environment, the agent determines the number of batches to produce, x_t, and ship, $z_{j,t}$, to each distribution warehouse $j = 1, \dots, J$.

2. Each distribution warehouse $j = 1, \dots, J$ receives the sent batches of stocks, $z_{j,t}$.

3. Demand $d_{j,t}$ at each distribution warehouse $j = 1, \dots, J$ is either satisfied or backordered.

Table 3.1: The considered SCIM notation with relative explanation (and units of measure).

Notation	Explanation	Notation	Explanation
T	Time Horizon	$I_j^{\max}$	Storage Capacity (batches)
J	Number of Distribution Warehouses	h_j	Storage Cost (per batch)
$\mathcal{J}$	Set of Distribution Warehouses	p_j	Logistic Cost Fixed (per vehicle)
$d_{j,t}$	Demand (batches)	l_j	Logistic Cost Variable (per batch)
x_t	Production (batches)	$z_{j,t}$	Shipping (batches)
c_0	Production Cost (per batch)	V_j	Vehicles Capacities (batches)
$X^{\max}$	Maximum Production (batches)	$y_{j,t}$	Number of Vehicles
$I_{j,t}$	Stock Level (batches)	b_j	Backorder Cost (per batch)

4. The *per-step cost*, C_t, is calculated according to the following formula:

$$\begin{aligned} C_t = & \underbrace{c_0 \cdot x_t}_{\text{production costs}} + \underbrace{\sum_{j=1}^{J} l_j \cdot z_{j,t}}_{\text{variable logistic costs}} + \underbrace{\sum_{j=1}^{J} p_j \cdot y_{j,t}}_{\text{fixed logistic costs}} \\ & + \underbrace{\sum_{j=0}^{J} h_j \cdot \max[I_{j,t}, 0]}_{\text{storage costs}} - \underbrace{\sum_{j=0}^{J} b_j \cdot \min[I_{j,t}, 0]}_{\text{backorder costs}}, \end{aligned} \tag{3.1}$$

   where the first term represents production costs, the second and the third consist of variable and fixed logistic costs, respectively, the fourth denotes storage costs, and the last one quantifies backorder costs (which are introduced with a minus sign because stock levels would be negative in the eventuality of unsatisfied demand).

5. The state related to the next timestep is determined, transferring surplus stocks or unsatisfied demands. Formally, the evolution of the inventories is defined as follows:

$$\begin{cases} I_{0,t+1} = \min\left[\left(I_{0,t} + x_t\right), I_0^{max}\right] - \sum_{j=1}^{J} z_{j,t} \\ I_{1,t+1} = \min\left[\left(I_{1,t} + z_{1,t-1}\right), I_1^{\max}\right] - d_{1,t} \\ \quad \vdots \\ I_{J,t+1} = \min\left[\left(I_{J,t} + z_{J,t-1}\right), I_J^{\max}\right] - d_{J,t}. \end{cases} \tag{3.2}$$

   This implies that at the beginning of the timestep $t+1$, the factory's stocks are equal to the sum of the stocks at timestep t and the production during the same period, minus the batches shipped during timestep t. Similarly, the distribution warehouses' stocks at timestep $t+1$ are equal to the stocks at timestep t, plus the batches received from the factory during the same period, minus the demand at timestep t. It is worth mentioning that the environment does not allow storing

a number of batches exceeding the storage capacity constraints, automatically discarding all batches that, if accepted, would violate these constraints.

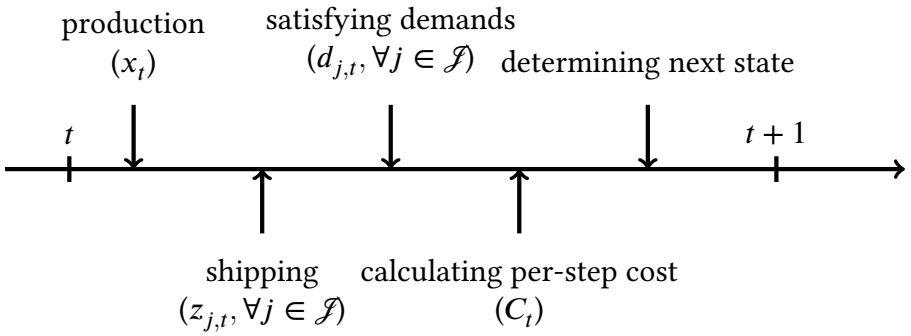


Figure 3.2: Representation of the dynamics of the SCIM system considered.

3.4 Solution Methods

In this section, we present the techniques used to solve the problem described in Section 3.3. Section 3.4.1 offers a brief introduction to deep reinforcement learning, while Section 3.4.2 defines the multi-stage stochastic programming model. Finally, Section 3.4.3 describes the proposed heuristic, which combines DRL and MS optimization.

3.4.1 Deep Reinforcement Learning

RL adopts the Markov decision process (MDP) framework to represent the interactions between a *learning agent* and an *environment*. As shown in Figure 3.3, at each timestep t, the agent observes the current state of the environment, $\mathbf{s}_t$, selects an action, $\mathbf{a}_t$, and obtains a reward, $R_{t+1} \in \mathbb{R}$; afterward, the environment transitions into a new state, $\mathbf{s}_{t+1}$. The goal of RL is to find an *optimal policy* – a function that maps the states of the environment to a set of actions – that maximizes the *expected discounted return*, $\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$, where $0 \leq \gamma \leq 1$ represents the discount rate [29]. In the SCIM problem, the objective is to minimize the total cost, therefore we define the reward as the negative cost, i.e., $R_t = -C_t$.

DRL combines RL with deep learning, offering the potential to scale to previously intractable decision-making problems. In detail, DRL is rooted in neural networks, which are universal approximators capable of expressing an approximation of highly nonlinear functions. The specific DRL algorithm we used belongs to *policy-based methods*, which learn a parameterized and stochastic policy to select actions directly (in contrast to value-based methods like Q-learning [38]). To formally address and solve the SCIM problem within the context of an MDP, we thus need to define the state vector, the action space, and the reward function relating to the SCIM problem assumed.

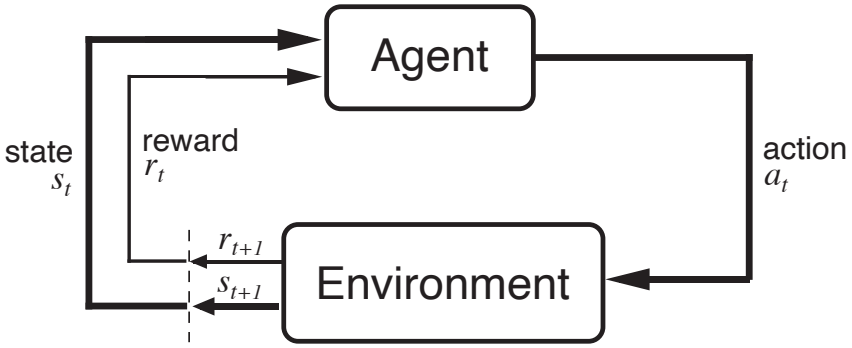


Figure 3.3: Agent-environment interface in an MDP [38].

The *state vector* includes all current stock levels (for the factory and each distribution warehouse) and the last τ demand values; this provides the agent with limited knowledge of demand history, which, in turn, allows for a basic understanding of its fluctuations (similar to what was initially proposed by Kemmer et al. [23]). In mathematical terms, we have:

$$\mathbf{s}_t = \left(I_{0,t}, \dots, I_{J,t}, \mathbf{d}_{t-1}, \dots, \mathbf{d}_{t-\tau}\right), \tag{3.3}$$

where $\mathbf{d}_t = \left(d_{0,t}, \dots, d_{J,t}\right)$. The *state's updating rule* is dictated by Equation (3.2) for the inventory, while the values of the demands are updated based on the current timestep. It is worth noting that during timestep t, the actual demand $\mathbf{d}_t$ is only known after decisions regarding production and shipping have been made; this ensures that the agent can benefit from learning the demand pattern, allowing for the integration of *demand forecasting* directly into the policy. Further, by observing and learning from the demand history, the agent can develop a more effective decision-making process, anticipating future demand fluctuations and adjusting its actions accordingly [35].

Concerning the *action space*, it contains information about production and shipping controls:

$$\mathbf{a}_t = \left(x_t, z_{1,t} \dots, z_{J,t}\right). \tag{3.4}$$

The considered agent uses a *continuous action space* – wherein the neural network directly generates the action values – since it scales better than a discrete action space and can be applied to wider action spaces [39]. As a result, the agent can specify the factory's production level and the number of batches to ship to each distribution warehouse.

To guarantee that the actions generated by the neural network belong to the feasible action space, we adopted a continuous action space based on independent bounds [35]. This involves: i) setting the lower bound for each action value to zero; ii) setting the upper bound for the factory to its maximum production level (i.e., $0 \leq x_t \leq X^{\max}$); and iii) setting the upper bound for each warehouse to its corresponding storage capacity (i.e., $0 \leq z_{j,t} \leq I_j^{\max}$, $\forall j \in \mathcal{J}$). It is worth noticing that if the agent produces or ships a number of batches exceeding the free storage availability (for example, due

to stocks preserved in the warehouses), the environment directly discards those in excess, according to Equation (3.2). This results in a cost, which implicitly penalizes the agent, thus encouraging it to learn a policy that prevents overproduction and excessive shipping. Additionally, when factory stocks are insufficient to meet orders from all distribution warehouses, we employ the allocation rule described in Stranieri, Stella, and Kouki [36] to ensure a fair distribution of batches among the distribution warehouses while maintaining factory stocks non-negative.

Finally, the *reward function* is defined by means of Equation (3.1).

3.4.2 Multi-Stage Stochastic Programming

In multi-stage stochastic programming, the uncertainty associated with demand is represented through a *scenario tree*, as depicted in Figure 3.4. Each layer of the tree corresponds to a stage and models the available information. In the SCIM problem, because new information (i.e., demand realization) becomes available at each time step, the concepts of stage and time steps are equivalent.

The leftmost node (labeled as node 0 in Figure 3.4) represents the current state of the environment where we determine the *first-stage decisions*, each associated with an immediate and definite per-step cost. These decisions influence the second-stage ones, made after observing the realized demand (as illustrated by nodes 1 and 2 in Figure 3.4). Then, in the subsequent stages of the scenario tree, the process is repeated [5]. Notably, decisions made after the first stage can be viewed as *contingent plans* based on future demands realizations.

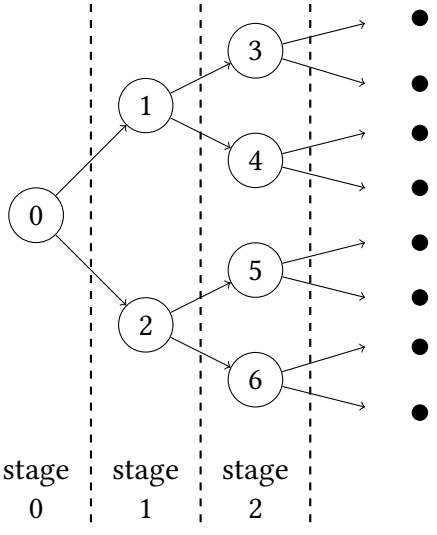


Figure 3.4: Scenario tree representation where each node represents a possible demand realization.

To this end, we let:

- $\mathcal{N}$ be the set of nodes of the scenario tree, $\mathcal{N}^+ = \mathcal{N} \setminus \{0\}$, with 0 being the root node.
- $p(n)$ be the *parent* of node $n \in \mathcal{N}^+$; for example, in Figure 3.4, $p(1) = p(2) = 0,\ p(3) = p(4) = 1, \dots$.
- $\pi^{[n]}$ be the *unconditional probability* of node n (i.e., the likelihood of demand realization in node n expressed considering the information available from the root node) [10]; for example, in Figure 3.4, we have $\pi^{[0]} = 1$, $\pi^{[1]} = \pi^{[2]} = \frac{1}{2}$, $\pi^{[3]} = \pi^{[4]} = \pi^{[5]} = \pi^{[6]} = \frac{1}{4}, \dots$.
- $d_j^{[n]}$ be the item demand from distribution warehouse j at node $n \in \mathcal{N}$.

In the MS model, given that the decision variables are contingent on the node of the scenario tree (which includes all available information), we replace the subscript $\cdot_t$ with the superscript $\cdot^{[n]}$ for all variables. The superscript $\cdot^{[n]}$ denotes a specific node of the scenario tree and is consequently associated with a single stage and a single time step. For example, x_t in Table 3.1 becomes $x^{[n]}$, representing the quantity produced at node n. It is essential to note that our primary interest lies in the variables calculated in the first stage, in particular, the quantity to produce ($x^{[0]}$), the number of batches to ship ($z_j^{[0]}, \forall j \in \mathcal{J}$), and the number of vehicles to be employed ($y_j^{[0]}, \forall j \in \mathcal{J}$).

The MS optimization model can thus be defined as follows:

$$\min_{\substack{x^{[n]} \\ \forall n \in \mathcal{N} \\ y_j^{[n]}, z_j^{[n]}, I_j^{[n]} \\ \forall n \in \mathcal{N}, \forall j \in \mathcal{J}}} \sum_{n \in \mathcal{N}} \pi^{[n]} \left[c_0 x^{[n]} + \sum_{j=1}^{J} l_j z_j^{[n]} + \sum_{j=1}^{J} p_j y_j^{[n]} + \sum_{j=0}^{J} h_j I_j^{+[n]} + \sum_{j=0}^{J} b_j I_j^{-[n]} \right] \tag{3.5}$$

$$\text{s.t.} \quad I_0^{[n]} = I_0^{[p(n)]} + x^{[n]} - \sum_{j=1}^{J} z_j^{[n]} \qquad \forall n \in \mathcal{N}^+ \tag{3.6}$$

$$I_j^{[n]} = I_j^{[p(n)]} + z_j^{[p(n)]} - d_j^{[n]} \qquad \forall n \in \mathcal{N}^+, \forall j \in \mathcal{J} \tag{3.7}$$

$$I_j^{[n]} = I_j^{+[n]} - I_j^{-[n]} \qquad \forall n \in \mathcal{N},\ \forall j \in \mathcal{J} \cup \{0\} \tag{3.8}$$

$$I_j^{[0]} = \bar{I}_j^{[0]} \qquad \forall j \in \mathcal{J} \cup \{0\} \tag{3.9}$$

$$y_j^{[n]} \geq \frac{z_j^{[n]}}{V_j} \qquad \forall n \in \mathcal{N},\ \forall j \in \mathcal{J} \tag{3.10}$$

$$x^{[n]} \in \{0, \dots, X^{\max}\} \qquad \forall n \in \mathcal{N} \tag{3.11}$$

$$z_j^{[n]} \in \mathbb{Z}_+, \quad y_j^{[n]} \in \mathbb{Z}_+ \qquad \forall n \in \mathcal{N},\ \forall j \in \mathcal{J} \tag{3.12}$$

$$I_j^{[n]} \in \mathbb{Z}, \quad I_j^{+[n]} \in \{0, \dots, I_j^{\max}\}, \quad I_j^{-[n]} \in \mathbb{Z}_+ \quad \forall n \in \mathcal{N},\ \forall j \in \mathcal{J} \cup \{0\}. \tag{3.13}$$

The objective function in Equation (3.5) represents the expected total costs, expressed as the sum of production, logistics, storage, and backorder costs. Constraints (3.6)-(3.7) dictate the evolution of the stocks in the factory and distribution warehouses. These constraints are analogous to Equation (3.2) but contextualized to MS stochastic programming. Moreover, constraints (3.8) define the relationship between the variable $I^{[n]}$, the inventory $I^{+[n]}$, and the backorder $I^{-[n]}$. These latter two are variables used to linearize the max and min operators in Equation (3.1). Lastly, constraints (3.9) establish the initial stock condition, constraints (3.10) determine the number of vehicles used for shipping, and constraints (3.11)-(3.13) describe the domains to which the variables belong. It is important to note that model (3.5)-(3.13) implicitly enforces the *non-anticipative constraints*, meaning that the decision-maker cannot exploit future information [10]. For example, in Figure 3.4, at node 2, the demand realizations for both nodes 5 and 6 are possible. By solving this MS optimization model, the SCIM problem can be optimized to *minimize total costs* while trying to satisfy the overall demand from the distribution warehouses.

3.4.3 Deep Reinforcement Learning-Based Decomposition

The proposed heuristic combines the main points of strength of both DRL and MS programming, leveraging their complementary characteristics to address their respective weaknesses.

The principal advantage of MS programming lies in its model-based problem formulation, where both the objective function and the constraints are *explicitly defined* within the model; this is particularly effective when the model accurately represents the problem, as in the considered context. Indeed, the mathematical model adeptly captures the computation of inventories evolution and backlogs, the number of vehicles to use, and their consequent impact on the objective function. By contrast, the main drawback of MS programming stands in its potential need for multiple stages to properly represent a stochastic and seasonal demand, thus leading to challenging problems that, in the worst case, cannot be solved by an exact solver within a reasonable amount of time. Specifically, the bottleneck in MS programming is the number of scenarios required to approximate the *out-of-sample cost* of the solution when complex demands are considered. Seasonal demand is a prime example where several stages – and thus several scenarios – are needed to characterize the evolving demand distribution.

In general, given J distribution warehouses, N possible demand realizations for each stage, and T stages (assuming the number of stages equals the time horizon), we end up with N^T scenarios and $(5J+2) \cdot N^T$ variables. Even small values of J, N, and T prevent an exact solver from computing the optimal solution of the model. Furthermore, in the

SCIM problem, the *risk mitigation strategy* involves producing and storing batches in warehouses, thereby paying relative storage costs. Hence, inventory decisions require the longest look ahead. Conversely, seasonality is not a major issue for DRL since it adapts its strategy to random demand based on the *policy learned* during the training phase. Although the training phase for complex demands can take time, it is only required once. Thus, in a testing environment, the solution is nearly immediate. Nevertheless, being a model-free method, DRL suffers from slow convergence, a weakness that becomes even more marked as the number of constraints increases, as it happens in our current setting.

The complementary strengths and weaknesses of MS programming and DRL suggest a potential synergy if combined to develop a novel heuristic, exploiting the policy provided by DRL for decisions requiring a longer time horizon and using MS to determine all other variables that need a shorter time horizon.

More in detail, we consider two distinct agents at play, one driven by DRL and the other by MS programming. In the following, we will employ the MDP notation $\cdot_t$ to refer to the variables associated with the DRL agent while using the notation $\cdot^{[n]}$ for those tied to the MS agent. Thus, for instance, x_t denotes the MDP variable that describes the production quantity at timestep t, whereas $x^{[0]}$ pertains to the MS model (3.5)-(3.13), which will be solved with a rolling horizon approach to determine the production quantity at timestep t.

Initially, we train the DRL agent over a set of episodes; then, we use the trained DRL policy to select the number of batches to produce, thereby setting the value x_t. Identifying an optimal value for this decision variable requires deep foresight; moreover, any value within the range $[0, X^{\max}]$ yields a feasible solution, eliminating the need for the DRL agent to learn complex constraints). Once the decision about the number of batches to produce has been made, the second agent implements an MS model to handle the logistics decisions. Specifically, it solves model (3.5)-(3.13) by fixing the $x^{[0]}$ variable with the DRL's derived solution.

Since the production decisions now depend only on the DRL agent, the MS agent no longer needs to consider an extensive number of stages. Thus, we reduce the number of stages to two, leading to a significantly more manageable model. However, the number of scenarios within the second stage might still lead to computational issues. To counteract this, we reduce the number of scenarios using Monte Carlo techniques, such as *moment matching generation* or *importance sampling* [8]. Since all the second-stage variables are used to account for the future, we relax them to be continuous, and we only constrain the first-stage variables $y_j^{[0]}$ to be integers. This strategy has been successfully employed in *fix-and-relax heuristics* [9]. By adopting this approach, despite the introduction of approximation errors at future stages, we can guarantee the quick achievement of a feasible solution.

The pseudocode summarizing the heuristic is outlined in Algorithm 2. In the following sections, we will refer to this heuristic as deep reinforcement learning-based decomposition (DRLBD).

Algorithm 2 Deep Reinforcement Learning-Based Decomposition (DRLBD) algorithm.

Train a DRL agent
for $t = 1, \dots, T$ **do**
    Observe the current state of the environment, s_t
    Determine the production level x_t using the DRL agent
    Generate a scenario tree
    Set $x^{[0]} \leftarrow x_t$ in model (3.5)-(3.13)
    Relax variables $y_j^{[n]} \in \mathbb{R}_+, \forall j = 1, \dots, J, n \neq 0$
    Solve model (3.5)-(3.13) using the MS agent
    Set $z_{j,t} \leftarrow z_{j,t}^{[0]}, \ \forall j = 1, \dots, J$
    Produce x_t and ship $z_{j,t}, \ \forall j = 1, \dots, J$
    Satisfy demand $d_{j,t}, \forall j = 1, \dots, J$
    Calculate per-step cost C_t and determine next state s_{t+1}

3.5 Numerical Experiments

This section presents the numerical experiments designed to evaluate the performance of the proposed heuristic algorithm. All experiments were executed on a machine equipped with an Apple M2 Pro chip with 10 cores and 16 GB of RAM. We developed the code using Python 3.10. Specifically, we used: i) the OpenAI Gym APIs [12] to implement the SCIM environment; ii) the Ray Python library [26] for importing the DRL algorithms; iii) Gurobi version 10.0 (via its Python APIs) [18] to solve the MS optimization model.

We organize the numerical experiments into *two different settings*. The first setting is elementary and aims to compare the methods under consideration with the derived optimal solution. The second setting involves more challenging experiments to evaluate in-depth the performance of the proposed heuristic in scenarios where an optimal solution cannot be computed within a reasonable amount of time. The specific values of the parameters that define the SCIM environment for both small and large settings are reported in Appendix B.1. Due to the lack of real-world data or benchmark instances, our experimental plan relies on *synthetic and realistic data*. However, by providing open access to our code, we aim to offer a reference point for future research, thereby addressing this limitation. While the data used in Section 3.5.1 ensure a specific problem structure but lacks realism, the data in Section 3.5.2 has been validated in collaboration with Bristol-Myers Squibb [2] to simulate a realistic supply chain.

Following Kemmer et al. [23], and Peng et al. [28], we define the seasonal demand

[2] https://www.bms.com/

for warehouse j as:

$$d_{j,t} = \left\lfloor D_j \left(1 + \sin\left(\frac{2\pi(t-\phi_j)}{P_j}\right)\right)\right\rfloor + \epsilon_j, \qquad \forall\, j = 1, \dots, J,\ \forall t = 1, \dots, T, \tag{3.14}$$

where $\lfloor \cdot \rfloor$ represents the floor function, D_j denotes the amplitude of the reference demand value (set to exceed the maximal production $X^{\max}$), ϕ_j is the phase, P_j represents the period (set as a fraction of T), and ϵ_j indicates a *random noise* which will be defined differently in each experimental setting. Figure 3.5 provides an empirical example of a 95% confidence interval where $D_j = 5$, $P_j = 5$ over a time horizon of $T = 7$ timesteps, and $\phi_j = 0$, with ϵ_j as in Equation (3.16).

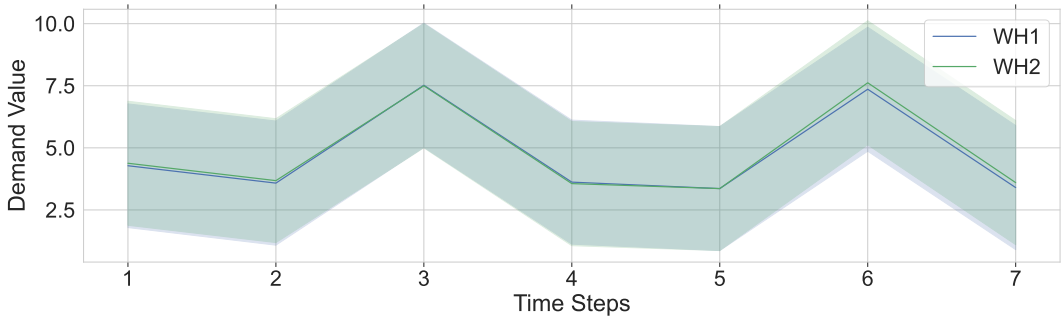


Figure 3.5: 95% confidence interval computed over 250 realizations of the seasonal demand with two distribution warehouses (i.e., WH1 and WH2), $D_j = 5$, $P_j = 5$, $T = 7$, $\phi_j = 0$, and ϵ as in Equation (3.16).

If $\phi_j \neq 0$ and $P_j \neq P_k, \forall j \neq k \in \mathcal{J}$, demands manifest peaks at different timesteps. Accordingly, summing the demands across all warehouses will diminish the impact of these peaks, enabling the agent to make good decisions with a short look-ahead. Given these insights, we consider $\phi_j = 0$, $\forall j \in \mathcal{J}$, and $P_j = P_k$, $\forall j, k \in \mathcal{J}$; this represents the most challenging scenario, characterized by *complete demands overlapping* that results in pronounced demand peaks, thus necessitating a wise production and stocking strategy. Such a scenario is particularly plausible when warehouses are located in homogeneous regions that follow equivalent demand patterns. Examples include the paper notebook industry, where sales synchronize with the academic calendar (consistent at a national level), or the food industry producing items linked to specific times of the year (such as Christmas cakes). In fact, production might only occur in just a few periods of the year in these supply chains.

For the sake of simplicity, we assume that random noises ϵ_j are i.i.d. across all distribution warehouses j and that all D_j values are equal. As a result, we will omit the subscript j from ϵ_j, P_j, and D_j in subsequent sections. Additionally, we assume that

when production capacity is used strategically, all demands can be satisfied; mathematically, this means that:

$$\frac{P \cdot X^{\max}}{J} \geq \sum_{t=0}^{P} \left\lfloor D \left(1 + \sin \left(\frac{2\pi(t - \phi_j)}{P} \right) \right) \right\rfloor + \bar{\epsilon}, \tag{3.15}$$

where $\bar{\epsilon}$ represents the average demand noise from a single distribution warehouse.

For performance comparisons, we calculate the total costs as $\sum_{t=1}^{T} C_t$ for each experiment, and we replicate this process 250 times. As benchmark techniques, we exploit the PPO algorithm [34], given its demonstrated superiority in comparable settings over other state-of-the-art DRL algorithms, such as VPG and A3C [35, 36]. For the same reasons, DRLBD also adopts PPO as its foundation algorithm. The hyperparameters for DRLBD and PPO are detailed in Appendix B.2.

3.5.1 Small Settings

In the first context, we examine experiments attributable to a *small setting* involving two distribution warehouses with demand parameters $D = 5$ and $P = 5$, spanning a time horizon of $T = 7$ timesteps where each timestep conceptually represents a *day*. The primary goal of this setting is to compute the optimality gap of the proposed heuristic rather than evaluate its performance in a realistic context.

In detail, two experiments are conducted within this setting: the first considers a noise ϵ distributed according to a Bernoulli distribution with a probability of 0.5, while the second experiment employs a different distribution for the noise ϵ, defined as follows:

$$\epsilon = \begin{cases} 0 & \text{with probability } 0.5 \\ 5 & \text{with probability } 0.5 \end{cases}. \tag{3.16}$$

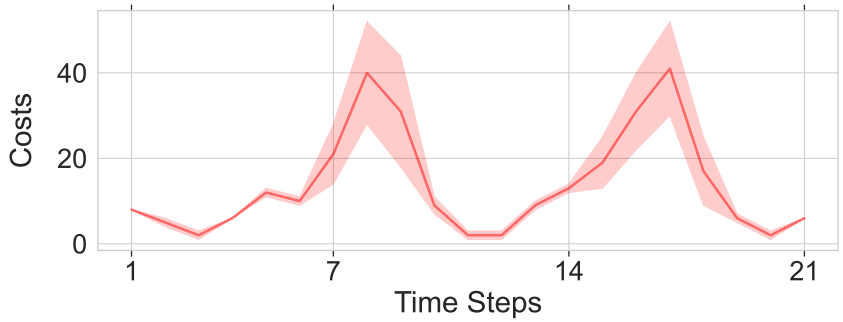
We select these two distributions because, for a specific number of batches, they model to either have an increased demand over a short number of timesteps or none at all. Furthermore, an optimal solution to the problem can be easily computed using these distributions.

One viable strategy would involve value iteration or policy iteration [7]. However, these methods require an excessive amount of time to reach convergence. As a result, we opt to directly solve the MS problem as expressed in model (3.5)-(3.13). In fact, since ϵ can take only four distinct values (i.e., two possible outcomes for each of the two distribution warehouses), considering T stages gives rise to 4^T different scenarios. For small values of T, this leads to a mathematical model that can be solved by an exact solver, thereby providing the *optimal solution* (in that case, the scenarios tree precisely describes every possible demand realization with its exact probability).

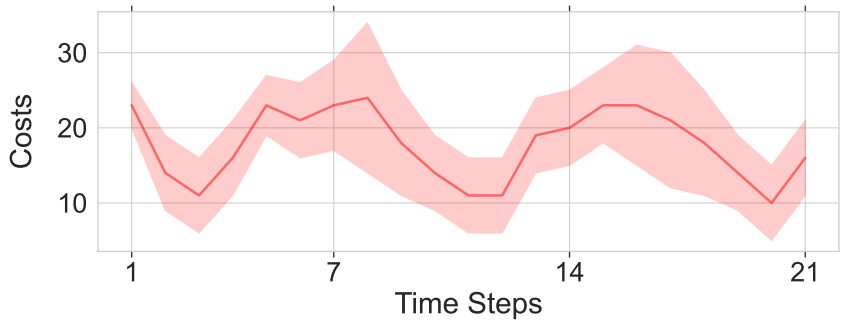
It is worth noting that there is no advantage in considering a number of stages exceeding the demand period P since, according to Equation (3.15), decisions made

during the first stage have a *limited influence* on the future. Although not detailed here due to their irrelevance to the scope of the paper, computational experiments suggest that considering only four stages is sufficient to obtain a practically equal solution derived from a model considering all T stages. Thus, the resulting MS stochastic problem considers just (4^4 =) 256 scenarios and can be solved by off-the-shelves solvers like Gurobi.

Before delving into a comparison of the proposed heuristic, it is essential to assess the impact of *initial conditions*. To this end, we investigate the cost incurred per timestep starting from empty inventories over a period of $T = 21$ timesteps. The results for the MS stochastic model are reported in Figure 3.6. From both Figures 3.6a and 3.6b, it is evident that there is almost no *transient phase*, as evidenced by the periodicity depicted in the two plots.



(a) Cost per timestep with $\epsilon \sim \mathscr{B}(0.5)$.



(b) Cost per timestep with ϵ as in Equation (3.16).

Figure 3.6: Cost per timestep over 250 episodes, as computed by the multi-stage programming model. The X-axis represents the timesteps, while the Y-axis displays the per-step cost. From both Figures 3.6a and 3.6b, which refers to the experiments of the small settings (but with $T = 21$), it is possible to observe the periodic behavior of the costs.

To assess the performance of the proposed heuristic, we benchmark it against the PPO algorithm, the (s, Q)-policy (hereafter referred to as sQ), and the *expected value problem* (EVP). Specifically, when employing EVP, we solve the model (3.5)-(3.13) by

replacing each random variable with its expected value; this leads to a deterministic problem that exact solvers can handle given that the scenario tree depicted in Figure 3.4 is reduced to a single chain of length T.

We present the *percentage optimality gaps* derived from simulations in Table 3.2. Within the table, we also report the performance of the *perfect information* (PI) model. This model is obtained by solving the model (3.5)-(3.13) and substituting each random variable with its future realization. Analogous to EVP, this results in a scenario tree comprising T nodes, which can be addressed using exact solvers. It is important to highlight that the PI model assumes complete knowledge of future demand outcomes, making it not implementable in a real-world deployment. Nevertheless, its introduction is helpful for quantifying the inherent value of perfect information regarding the future [5]. It also serves as a reference for strategically setting the production capacity by empirically verifying through simulation that all demands can be satisfied. Accordingly, Equation (3.15) is applied as an additional analytical check to ensure the robustness of this calibration.

Table 3.2: Average opt-gap (expressed as a percentage) over 250 episodes with respect to the multi-stage programming model. The standard deviation is provided in round brackets. The lower the value, the better the algorithm.

Algorithm	Opt-gap [%]
DRLBD	6.10 (2)
PPO	24.67 (9)
sQ	64.09 (30)
EVP	116.95 (59)
PI	-7.79 (3)

(a) Average opt-gap (as a percentage) with $\epsilon \sim \mathscr{B}(0.5)$.

Algorithm	Opt-gap [%]
DRLBD	8.66 (3)
PPO	35.66 (9)
sQ	47.71 (12)
EVP	198.62 (160)
PI	-33.63 (9)

(b) Average opt-gap (as a percentage) with ϵ as in Equation (3.16).

As the reader can observe, EVP underperforms in both settings, reporting costs that are 116% and 198% worse than the average MS solution, respectively. Trailing behind EVP, there is sQ, which has an optimality gap of 64% and 47%. Such performance can be attributed to its static nature, which often leads to unsatisfactory solutions, especially in scenarios with seasonal demand. Interestingly, sQ emerges as the only method where

the optimality gaps diminish as the variance associated with the noise increases; this observation can be rationalized by recalling that, in the i.i.d. demand context, the (s, Q)-policy is intrinsically optimal [41]. Moreover, it is noteworthy that as the noise variance increases, the impact of seasonality on performance diminishes.

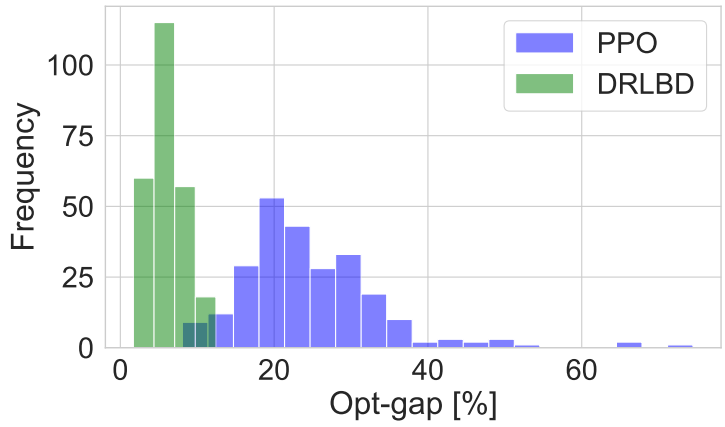
In comparison, the PPO algorithm achieves better optimality gaps than sQ, reporting gaps of 24% and 35%, respectively, and attesting to the improved performance of dynamic rules over static ones. Meanwhile, DRLBD reaches an optimality gap of only 6% and 8%, consolidating its position as the most effective approach. Its performance, combined with a narrow standard deviation, reveals that our proposed heuristic consistently earns the lowest optimality gap. In the authors' opinion, these considerable improvements can be mainly attributable to the enhanced logistics management promoted by the MS model.

Lastly, the PI model, as intended, manifests negative optimality gaps, a direct consequence of its ability to access future knowledge. Furthermore, as it is intuitive to guess, PI excels particularly in settings characterized by higher demand variance, where a precise foreknowledge of upcoming scenarios becomes an invaluable asset.

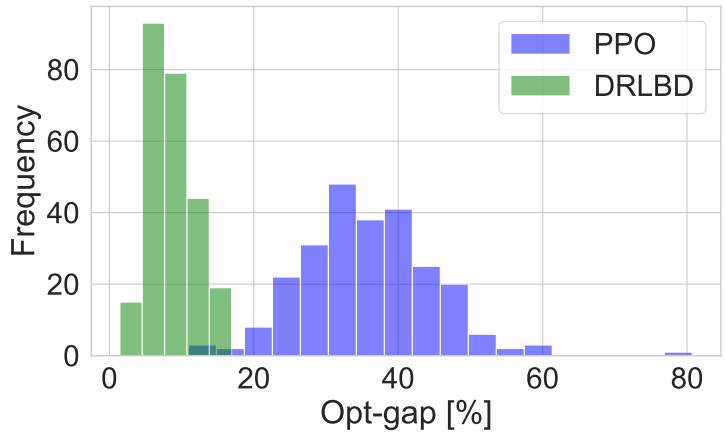
While average values and standard deviations provide some insight, they do not fully describe the actual distribution of costs. Therefore, Figure 3.7 delves deeper, presenting the average optimality gaps through histograms of *frequency distributions*. Given that the primary goal of this small experiment setting is to evaluate the optimality gap of the proposed methodologies, we focus exclusively on the gaps rather than the absolute cost values. Moreover, our analysis considers only PPO and DRLBD, as they emerged as the most performing techniques.

From Figures 3.7a and 3.7b, it is evident that the cost interval for DRLBD is considerably more confined than that of PPO. In particular, the poorest results for DRLBD – approximately 10% for the first experiment and 15% for the second – are significantly better than those achieved by PPO, which hover around 70% and 80% for the first and second experiments, respectively. Moreover, DRLBD reaches values close to optimality in both experiments, whereas PPO consistently achieves optimality gaps greater than the average values achieved by DRLBD, thus suggesting that DRLBD may offer a more robust and efficient solution than PPO.

Regarding *computational times* for the different techniques, Table 3.3 contains the training times (in minutes) and the time needed to execute a single episode, referred to as testing times (in milliseconds). Training times for MS, EVP, and PI are omitted from the table, as these methods do not necessitate performing any training phase, while DRLBD has the same training time as PPO since they share the same underlying DRL agent. In this first setting, the computational times are particularly fast. Specifically, all training times require less than 3 minutes, and all testing executions are completed in less than a second. However, certain methods diverge from the rest, i.e., MS, which needs to solve a MILP with a considerable number of variables, and EVP, which needs to consider the full-time horizon.



(a) Average opt-gap (as a frequency distribution) with $\epsilon \sim \mathcal{B}(0.5)$.



(b) Average opt-gap (as a frequency distribution) with ϵ as in Equation (3.16).

Figure 3.7: Average opt-gap (expressed as a frequency distribution) over 250 episodes in comparison to the multi-stage programming model.

3.5.2 Large Settings

In the second context, we consider experiments within a *large setting* involving 5 or 10 distribution warehouses (i.e., $J = 5$ or $J = 10$).

The demand parameters of Equation (3.14) are $D = 2$ and $P = 6$ over a time horizon of one year ($T = 12$), such that each timestep ideally represents a *month*. We consider the noise ϵ of the demand to be a random variable distributed according to a negative binomial distribution with parameters $r = 3$ and $p = 0.7$ [1, 3]. Due to the size of the experiments and the demand distribution, in this setting, we do not exploit an MS benchmark since the number of scenarios required to approximate the out-of-sample cost leads to a model that runs *out of memory*. Therefore, we report the results by

Table 3.3: Average training and testing times for the two experiments conducted under small settings (i.e., with $\epsilon \sim \mathscr{B}(0.5)$ and ϵ as in Equation (3.16), respectively) with their standard deviation (in brackets).

Algorithm	Small Setting (with $\epsilon \sim \mathscr{B}(0.5)$)		Small Setting (with ϵ as in Equation (3.16))	
	Training Time [min]	Testing Time [ms]	Training Time [min]	Testing Time [ms]
DRLBD	2.41 (0.01)	12 (2.01)	2.42 (0.01)	12 (2.01)
PPO	2.41 (0.01)	8 (15.70)	2.42 (0.01)	8 (3.39)
sQ	2.40 (0.01)	5 (0.61)	2.36 (0.01)	5 (0.56)
MS	-	279 (30.30)	-	298 (30.90)
EVP	-	49 (0.52)	-	39 (14.50)
PI	-	7 (0.83)	-	7 (1.18)

computing the average gap achieved for each method with respect to the performances reached by PI.

Accordingly, Table 3.4 presents the results of the comparison among the different benchmark techniques in terms of the expected value of perfect information (EVPI). By analyzing the values, it is evident that EVP underperforms, corroborating the conclusion that addressing uncertainty using ad-hoc methods is the appropriate approach. Indeed, the costs incurred by EVP are more than five times higher than those of DRLBD. These poor performances are mainly due to the inability of EVP to produce a sufficiently high amount of items during the initial timesteps of the considered time horizon. As a result, it suffers an initial backlog that becomes impossible to satisfy during the seasonal peak, leading to even higher backlogs. The second underperforming method is sQ which still outperforms EVP since it adopts a massive Q value to prevent backorder accumulation, unlike EVP.

As in the previous experiments, PPO achieves better results than sQ by leveraging its ability to develop a dynamic decision rule proper for seasonal behavior. Finally, DRLBD significantly outperforms all other techniques, showing its superiority compared to the inability of PPO to make advantageous logistic decisions, which, in turn, grows logistic and storage costs.

It is worth noting that all the gaps increase as the number of warehouses grows since it leads to a more challenging problem. Among all methods, sQ and EVP are most impacted by the shift from $J = 5$ to $J = 10$, with performance reductions of 43.17% and 28.56%, respectively. This drop also involves DRLBD and PPO, which decrease their performance solely by 7.00% and 7.62%, respectively, thus proving the stability of the proposed heuristic when confronted with an increasing number of distribution warehouses.

To gain a better understanding of the *cost distribution*, we present it in Figure 3.8, expressing the cost in thousands of euros (k€). Both PPO and DRLBD exhibit a similar distribution in both experiments. However, PPO achieves higher average costs, reaching

Table 3.4: Average EVPI-gap (expressed as a percentage) over 250 episodes concerning the expected value of perfect information model. The standard deviation is provided in round brackets. The lower the value, the better the algorithm.

Algorithm	EVPI-gap [%]
DRLBD	98.08 (21)
PPO	132.31 (32)
sQ	145.54 (34)
EVP	548.87 (161)

(a) Average EVPI-gap (as a percentage) with $J = 5$ distribution warehouses.

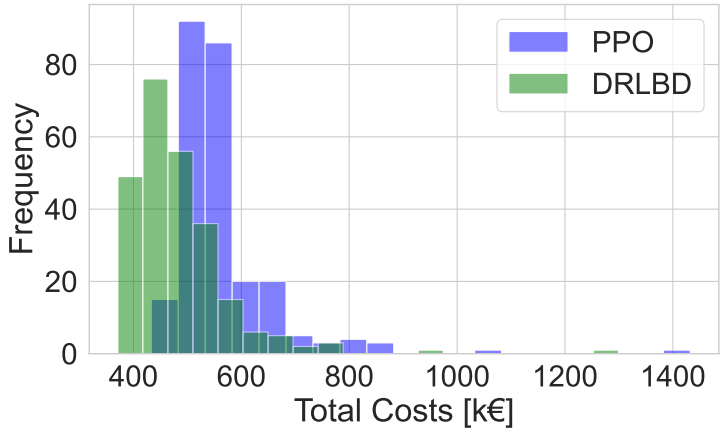
Algorithm	EVPI-gap [%]
DRLBD	105.08 (18)
PPO	139.93 (26)
sQ	188.71 (41)
EVP	577.43 (124)

(b) Average EVPI-gap (as a percentage) with $J = 10$ distribution warehouses.

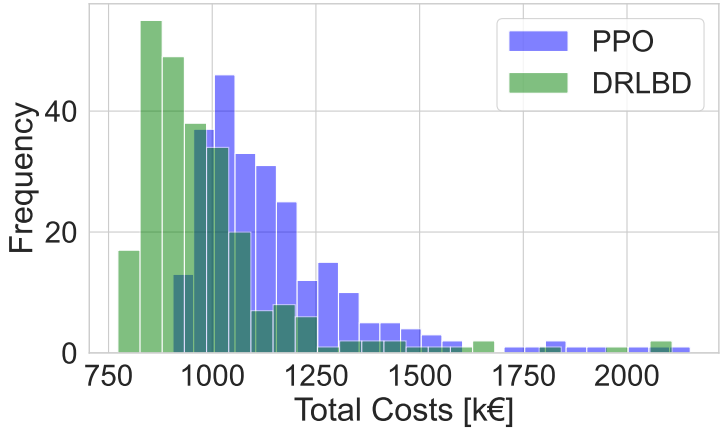
an average annual cost of 550k€ for the setting with $J = 5$ and 1100k€ for the setting with $J = 10$. In contrast, DRLB incurs average costs of 450k€ and 800k€ for the same two settings, respectively. Furthermore, the distribution queue for both methods appears fatter in the second experiment, especially for PPO; this is a result of the increased complexity derived from a greater number of distribution warehouses for which determining an appropriate quantity to be shipped becomes more challenging, thus leading to higher costs.

The *computational times* for both training and testing are detailed in Table 3.5. As in the previous subsection, we report the training time (in minutes) needed to learn a policy and the testing time (in milliseconds) required to execute a single episode. In comparison to small settings, the training time increases by a factor of three to five. Notably, sQ faces the most significant increase, requiring a training time of 12.17 minutes for the large setting experiment with $J = 10$, in contrast to approximately 2.40 minutes in the smaller setting experiments.

Despite the challenges posed by the large settings, all testing times remain under one second, thus proving that the primary computational bottleneck is the training phase of the algorithms. Moreover, such small computational times needed to compute actions allow for the execution of multiple experiments in a reasonable amount of time. This promptness is especially desirable for real-world applications, as decision-makers can swiftly conduct *what-if analyses*.



(a) Total costs histogram with $J = 5$ distribution warehouses.



(b) Total costs histogram with $J = 10$ distribution warehouses.

Figure 3.8: Total costs histograms over 250 episodes in relation to PPO and DRLBD techniques.

Among all methods, EVPI is most affected by the increment of J, increasing its computational time by 4.16 times. Finally, it is crucial to note that the testing times for DRLBD appear small; this can be attributed to the small number of stages and the variable relaxation applied to the mathematical model used to determine the shipping quantity, underscoring the DRLBD effectiveness also in this context.

3.5.3 Sensitivity Analysis

To further investigate how demand variations affect the results described and commented in the previous subsection, we design and execute additional experiments, replicating those from Section 3.5.2 while using different parameter values. Specifically, we focus on the setting consisting of 5 distribution warehouses (i.e., $J = 5$), as it facilitates to

Table 3.5: Average training and testing times for the two experiments conducted under large settings (i.e., with $J = 5$ and $J = 10$ distribution warehouses, respectively) with their standard deviation (in brackets).

Algorithm	Large Setting (with $J = 5$)		Large Setting (with $J = 10$)	
	Training Time [min]	Testing Time [ms]	Training Time [min]	Testing Time [ms]
DRLBD	6.40 (0.01)	28 (1.96)	11.30 (0.01)	40 (2.14)
PPO	6.40 (0.01)	18 (1.23)	11.30 (0.01)	25 (1.56)
sQ	6.04 (0.01)	11 (0.51)	12.17 (0.01)	14 (0.77)
EVP	-	49 (0.58)	-	204 (0.32)
PI	-	245 (612)	-	362 (1627)

perform numerical experiments that are both quick and easily manageable. The demand defined by Equation (3.14) depends on several parameters, such as amplitude D, period P, and phase ϕ_j of the reference demand value, along with parameters r and p associated with the random noise ϵ. Since conducting a comprehensive analysis of the results involving all these parameters would be complex, we opted to provide only some select and possibly valuable insights here. Nevertheless, the publicly available software code allows replicating the results and considering different parameter values.

In detail, we introduced variations in P and D values, i.e., $P = \{3, 6, 9\}$ and $D = \{1, 2, 3\}$. By adopting these parameters, we can adjust the number of demand peaks throughout the time horizon ($T = 12$) as well as the amplitude of the demand. Additionally, varying D allows us to regulate the balance between deterministic and stochastic demand components. Compatible with the previous subsection, we keep $\phi_j = 0$, expressing the harshest condition, while we expand the capacities of the distribution warehouses (i.e., $I_j^{\max} = 8$) according to the maximum D value employed (i.e., $D = 3$). This choice guarantees a standardized setting across all experiments, preventing the risk of backlogs caused by insufficient storage restriction. All other values are consistent with those presented in Section 3.5 (and are reported in Appendix B.1). As in our prior evaluations, we considered DRLBD, PPO, sQ, and EVP. In Table 3.6, we reported the relative gaps compared to PI (i.e., the EVPI gap).

The most challenging scenario for all techniques emerged when $P = 9$ and $D = 1$. In this configuration, a single demand peak appears over the time horizon, and the deterministic component is relatively small; this makes preventing backlogs particularly complex for all techniques. However, as the deterministic component grows, or as P decreases, the problem becomes more tractable since all methods begin to reduce costs. This outcome is reasonable as a greater deterministic component allows more accurate forecasting of the future. Moreover, when peaks occur more frequently, they can be captured with a shorter look-ahead interval, facilitating more effective production and shipping decisions. The most favorable scenario emerges when $D = 3$ and $P = 3$. Here, a demand peak appears every 3 time steps, and the demand is primarily influenced by its

Table 3.6: Average EVPI gap (as a percentage) by varying demand parameters $P = \{3, 6, 9\}$ and $D = \{1, 2, 3\}$, and considering $J = 5$ distribution warehouses. Adjusting these parameters allows us to alter the number of demand peaks and the balance between deterministic and stochastic demand components. The standard deviation is provided in round brackets. The lower the value, the better the algorithm.

		$D = 1$	$D = 2$	$D = 3$
$P = 3$	DRLBD	155.98 (21)	112.23 (25)	30.85 (28)
	PPO	199.63 (31)	167.63 (47)	45.88 (38)
	sQ	234.11 (42)	197.91 (55)	70.68 (47)
	EVP	860.06 (290)	595.91 (170)	116.68 (72)
$P = 6$	DRLBD	151.77 (21)	105.01 (22)	31.78 (27)
	PPO	196.01 (34)	156.54 (42)	47.10 (36)
	sQ	228.05 (44)	223.23 (70)	75.49 (52)
	EVP	858.57 (288)	571.42 (168)	122.25 (71)
$P = 9$	DRLBD	164.53 (23)	120.70 (25)	51.22 (29)
	PPO	221.37 (39)	187.65 (56)	59.00 (33)
	sQ	244.44 (42)	195.58 (42)	96.40 (48)
	EVP	901.54 (300)	618.92 (184)	183.21 (71)

deterministic component, thus promoting enhanced production and shipment planning.

It is interesting to notice that D affects the performance of the methods more than P. In fact, while the variation along the columns is around 30%, the variation along the rows is approximately 6%. The most substantial growth occurs when transitioning from $D = 2$ to $D = 3$; this is motivated by an increment of the deterministic demand component that enables all methods to make more knowledgeable decisions.

When comparing various techniques, it becomes evident that the proposed heuristic consistently outperforms its competitors in all settings. Among the others, PPO demonstrates superior results when compared to sQ and EVP, with sQ surpassing EVP. Interestingly, the most significant performance improvements across different D values are noticed with EVP, underscoring its heightened sensitivity to uncertainty. In contrast, the performance variations of the other methods do not reveal any substantial differences, thus confirming the conclusion that addressing uncertainty with appropriate methods is the proper approach.

3.6 Conclusions

In this paper, we introduce a novel heuristic for addressing the supply chain inventory management problem within a divergent two-echelon supply chain. The proposed

heuristic *decomposes* the problem using a deep reinforcement learning algorithm to determine the number of batches to produce and multi-stage stochastic programming to establish the quantities of batches to ship to each distribution warehouse. In practical terms, this approach *combines* the strengths of both techniques: the model-based approach of MS programming for immediate logistics decisions and the simulation-optimization abilities of DRL to make production decisions that require a longer look ahead interval.

We assessed the performance of the proposed heuristic through a series of numerical experiments. In smaller settings, where computing an optimal solution was feasible, DRLBD demonstrated performance reasonably close to exact methods. Moreover, in larger and more complex settings where determining an optimal solution was impracticable, DRLBD performed robustly, consistently outperforming both the PPO algorithm and the (s, Q)-policy used as benchmarks. To further substantiate these findings, we conducted a sensitivity analysis by varying specific demand parameters. This analysis confirmed the stability of our results through different value combinations, also proving that experiments featuring a low frequency of peaks and a small deterministic component are more challenging to handle.

From a computational time point of view, DRLBD rapidly computes actions, with the DRL training phase emerging as the primary potential bottleneck. This efficiency enables decision-makers to employ the proposed heuristic as an effective tool to conduct what-if analysis in a reasonable amount of time.

In conclusion, these findings underscore the potential of the proposed heuristic in adeptly addressing the SCIM problem across a range of distinct scenarios. Future research directions will explore more complex supply chain environments, encompassing critical factors like fixed production costs, uncertainty in item availability for the factory, and other relevant aspects. One such aspect will involve investigating the impact of adopting more complex and challenging demand distributions. We expect that the proposed heuristic will maintain its effectiveness, particularly in settings similar to those studied in this work, where strategic production and logistics decisions are essential to counteract myopic decision-making processes effectively. Lastly, investigations will extend to supply chain settings with more than two echelons to better evaluate the applicability and performance of the proposed heuristic. Through these efforts, we aspire to advance the understanding of the SCIM problem in increasingly complex and dynamic environments.

References

[1] Narendra Agrawal and Stephen A. Smith. "Estimating negative binomial demand for retail inventory management with unobservable lost sales." In: *Naval Research Logistics* 43.6 (Sept. 1996), pp. 839–861. DOI: 10.1002/(sici)1520-6750(199609)43:6<839::aid-nav4>3.0.co;2-5. URL: https://doi.org/10.10

02/(sici)1520-6750(199609)43:6%3C839::aid-nav4%3E3.0.co;2-5.

[2] A. Alonso-Ayuso et al. "An Approach for Strategic Supply Chain Planning under Uncertainty based on Stochastic 0-1 Programming." In: *Journal of Global Optimization* 26.1 (2003), pp. 97–124. DOI: 10.1023/a:1023071216923. URL: https://doi.org/10.1023/a:1023071216923.

[3] Yossi Aviv and Awi Federgruen. "Stochastic Inventory Models with Limited Production Capacity and Periodically Varying Parameters." In: *Probability in the Engineering and Informational Sciences* 11.1 (Jan. 1997), pp. 107–135. DOI: 10.1017/s026996480000471x. URL: https://doi.org/10.1017/s026996480000471x.

[4] Dimitri Bertsekas. *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific, 2021.

[5] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer New York, 2011. ISBN: 9781461402374. DOI: 10.1007/978-1-4614-0237-4. URL: http://dx.doi.org/10.1007/978-1-4614-0237-4.

[6] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[7] Paolo Brandimarte. *From Shortest Paths to Reinforcement Learning*. Springer International Publishing, 2021. DOI: 10.1007/978-3-030-61867-4. URL: https://doi.org/10.1007/978-3-030-61867-4.

[8] Paolo Brandimarte. *Handbook in Monte Carlo simulation: applications in financial engineering, risk management, and economics*. John Wiley & Sons, 2014.

[9] Paolo Brandimarte. "Multi-item capacitated lot-sizing with demand uncertainty." In: *International Journal of Production Research* 44.15 (Aug. 2006), pp. 2997–3022. DOI: 10.1080/00207540500435116. URL: https://doi.org/10.1080/00207540500435116.

[10] Paolo Brandimarte. *Quantitative Methods*. Wiley, Apr. 2011. DOI: 10.1002/9781118023525. URL: https://doi.org/10.1002/9781118023525.

[11] Paolo Brandimarte and Giulio Zotteri. *Introduction to Distribution Logistics*. Wiley, Mar. 2007. DOI: 10.1002/9780470170052. URL: https://doi.org/10.1002/9780470170052.

[12] Greg Brockman et al. *OpenAI Gym*. 2016. DOI: 10.48550/ARXIV.1606.01540. URL: https://arxiv.org/abs/1606.01540.

[13] S. Kamal Chaharsooghi, Jafar Heydari, and S. Hessameddin Zegordi. "A reinforcement learning model for supply chain ordering management: An application to the beer game." In: *Decision Support Systems* 45.4 (Nov. 2008), pp. 949–959. DOI: 10.1016/j.dss.2008.03.007. URL: https://doi.org/10.1016/j.dss.2008.03.007.

[14] Gabriel Dulac-Arnold et al. "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis." In: *Machine Learning* 110.9 (Apr. 2021), pp. 2419–2468. DOI: 10.1007/s10994-021-05961-4. URL: https://doi.org/10.1007/s10994-021-05961-4.

[15] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 1587–1596. URL: https://proceedings.mlr.press/v80/fujimoto18a.html.

[16] Joren Gijsbrechts et al. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems." In: *Manufacturing & Service Operations Management* 24.3 (May 2022), pp. 1349–1368. DOI: 10.1287/msom.2021.1064. URL: https://doi.org/10.1287/msom.2021.1064.

[17] Chandandeep S. Grewal, S.T. Enns, and Paul Rogers. "Dynamic reorder point replenishment strategies for a capacitated supply chain with seasonal demand." In: *Computers & Industrial Engineering* 80 (Feb. 2015), pp. 97–110. DOI: 10.1016/j.cie.2014.11.009. URL: https://doi.org/10.1016/j.cie.2014.11.009.

[18] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[19] Pavithra Harsha et al. "Math Programming based Reinforcement Learningfor Multi-Echelon Inventory Management." In: *SSRN Electronic Journal* (2021). DOI: 10.2139/ssrn.3901070. URL: https://doi.org/10.2139/ssrn.3901070.

[20] Peter Henderson et al. "Deep Reinforcement Learning That Matters." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[21] Yongxi Huang, Chien-Wei Chen, and Yueyue Fan. "Multistage optimization of the supply chains of biofuels." In: *Transportation Research Part E: Logistics and Transportation Review* 46.6 (Nov. 2010), pp. 820–830. DOI: 10.1016/j.tre.2010.03.002. URL: https://doi.org/10.1016/j.tre.2010.03.002.

[22] Christian D. Hubbs et al. *OR-Gym: A Reinforcement Learning Library for Operations Research Problems*. 2020. DOI: 10.48550/ARXIV.2008.06319. URL: https://arxiv.org/abs/2008.06319.

[23] Lukas Kemmer et al. "Reinforcement learning for supply chain optimization." In: *European Workshop on Reinforcement Learning*. Vol. 14. 2018.

[24] Moutaz Khouja. "Optimizing inventory decisions in a multi-stage multi-customer supply chain." In: *Transportation Research Part E: Logistics and Transportation Review* 39.3 (May 2003), pp. 193–208. DOI: 10.1016/s1366-5545(02)00036-4. URL: https://doi.org/10.1016/s1366-5545(02)00036-4.

[25] Ton de Kok et al. "A typology and literature review on stochastic multi-echelon inventory models." In: *European Journal of Operational Research* 269.3 (Sept. 2018), pp. 955–983. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.02.047. URL: http://dx.doi.org/10.1016/j.ejor.2018.02.047.

[26] Philipp Moritz et al. "Ray: A Distributed Framework for Emerging AI Applications." In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 561–577. ISBN: 978-1-939133-08-3. URL: https://www.usenix.org/conference/osdi18/presentation/moritz.

[27] Ahmad Mortazavi, Alireza Arshadi Khamseh, and Parham Azimi. "Designing of an intelligent self-adaptive model for supply chain ordering management system." In: *Engineering Applications of Artificial Intelligence* 37 (Jan. 2015), pp. 207–220. DOI: 10.1016/j.engappai.2014.09.004. URL: https://doi.org/10.1016/j.engappai.2014.09.004.

[28] Zedong Peng et al. "Deep Reinforcement Learning Approach for Capacitated Supply Chain optimization under Demand Uncertainty." In: *2019 Chinese Automation Congress (CAC)*. IEEE, Nov. 2019. DOI: 10.1109/cac48633.2019.8997498. URL: https://doi.org/10.1109/cac48633.2019.8997498.

[29] Warren B. Powell. *Approximate Dynamic Programming*. John Wiley & Sons, Inc., Aug. 2011. DOI: 10.1002/9781118029176. URL: https://doi.org/10.1002/9781118029176.

[30] Margaretha Preusser et al. "LP Modelling and Simulation of Supply Chain Networks." In: *Supply Chain Management und Logistik*. Physica-Verlag, 2010, pp. 95–113. DOI: 10.1007/3-7908-1625-6_4. URL: https://doi.org/10.1007/3-7908-1625-6_4.

[31] Kiran Kumar Ravulapati, Jaideep Rao, and Tapas K Das. "A reinforcement learning approach to stochastic business games." In: *IIE Transactions* 36.4 (Apr. 2004), pp. 373–385. DOI: 10.1080/07408170490278698. URL: https://doi.org/10.1080/07408170490278698.

[32] Benjamin Rolf et al. "A review on reinforcement learning algorithms and applications in supply chain management." In: *International Journal of Production Research* 61.20 (Nov. 2022), pp. 7151–7179. ISSN: 1366-588X. DOI: 10.1080/00207543.2022.2140221. URL: http://dx.doi.org/10.1080/00207543.2022.2140221.

[33] B. Van Roy et al. "A neuro-dynamic programming approach to retailer inventory management." In: *Proceedings of the 36th IEEE Conference on Decision and Control*. IEEE, 1997. DOI: 10.1109/cdc.1997.652501. URL: https://doi.org/10.1109/cdc.1997.652501.

[34] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[35] Francesco Stranieri and Fabio Stella. "Comparing Deep Reinforcement Learning Algorithms in Two-Echelon Supply Chains." In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer Nature Switzerland, 2025, pp. 454–469. ISBN: 9783031746406. DOI: 10.1007/978-3-031-74640-6_37. URL: http://dx.doi.org/10.1007/978-3-031-74640-6_37.

[36] Francesco Stranieri, Fabio Stella, and Chaaben Kouki. "Performance of deep reinforcement learning algorithms in two-echelon inventory control systems." In: *International Journal of Production Research* 62.17 (Mar. 2024), pp. 6211–6226. ISSN: 1366-588X. DOI: 10.1080/00207543.2024.2311180. URL: http://dx.doi.org/10.1080/00207543.2024.2311180.

[37] Zheng Sui, Abhijit Gosavi, and Li Lin. "A Reinforcement Learning Approach for Inventory Replenishment in Vendor-Managed Inventory Systems With Consignment Inventory." In: *Engineering Management Journal* 22.4 (Dec. 2010), pp. 44–53. DOI: 10.1080/10429247.2010.11431878. URL: https://doi.org/10.1080/10429247.2010.11431878.

[38] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039246.

[39] Nathalie Vanvuchelen and Robert N. Boute. "The Use of Continuous Action Representations to Scale Deep Reinforcement Learning: An Application to Inventory Control." In: *SSRN Electronic Journal* (2022). DOI: 10.2139/ssrn.4253600. URL: https://doi.org/10.2139/ssrn.4253600.

[40] Paul Vincent. "Exact Fill Rates For Items With Erratic Demand Patterns." In: *INFOR: Information Systems and Operational Research* 23.2 (May 1985), pp. 171–181. DOI: 10.1080/03155986.1985.11731953. URL: https://doi.org/10.1080/03155986.1985.11731953.

[41] Harvey M. Wagner and Thomson M. Whitin. "Dynamic Version of the Economic Lot Size Model." In: *Management Science* 5.1 (Oct. 1958), pp. 89–96. ISSN: 1526-5501. DOI: 10.1287/mnsc.5.1.89. URL: http://dx.doi.org/10.1287/mnsc.5.1.89.

[42] Yimo Yan et al. "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities." In: *Transportation Research Part E: Logistics and Transportation Review* 162 (June 2022), p. 102712. ISSN: 1366-5545. DOI: 10.1016/j.tre.2022.102712. URL: http://dx.doi.org/10.1016/j.tre.2022.102712.

[43] Paul Zipkin. *Foundations of inventory management*. en. Maidenhead, England: Irwin Professional Publishing, Apr. 2000.

Chapter 4

Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity

Francesco Stranieri[a,b], Chaaben Kouki[c], Willem van Jaarsveld[d], and Fabio Stella[a]

[a]Department of Informatics, Systems, and Communication (DISCo), University of Milano-Bicocca, Viale Sarca, 336, Milan, 20126, Italy
[b]Department of Control and Computer Engineering (DAUIN), Polytechnic of Turin, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy
[c]Department of Operations Management and Decision Science (OMDS), ESSCA School of Management, Rue Joseph Lakanal, 1, Angers, 49100, France
[d]Department of Industrial Engineering & Innovation Sciences (IE&IS), Eindhoven University of Technology, Groene Loper, 3, Eindhoven, 5612 AE, Netherlands

This is the original manuscript of a preprint available at: https://doi.org/10.48550/arXiv.2501.10895.

Abstract

We study inventory control policies for pharmaceutical supply chains, addressing challenges such as perishability, yield uncertainty, and non-stationary demand, combined with batching constraints, lead times, and lost sales. Collaborating with Bristol-Myers Squibb (BMS), we develop a realistic case study incorporating these factors and benchmark three policies–order-up-to (OUT), projected inventory level (PIL), and deep reinforcement learning (DRL) using the proximal policy optimization (PPO) algorithm–against a BMS baseline based on human expertise. We derive and validate bounds-based procedures for optimizing OUT and PIL policy parameters and propose a methodology for estimating projected inventory levels, which are also integrated into the DRL policy with demand forecasts to improve decision-making under non-stationarity. Compared to a human-driven policy, which avoids lost sales through higher holding costs, all three implemented policies achieve lower average costs but exhibit greater cost variability. While PIL demonstrates robust and consistent performance, OUT struggles under high lost sales costs, and PPO excels in complex and variable scenarios but requires significant computational effort. The findings suggest that while DRL shows potential, it does not outperform classical policies in all numerical experiments, highlighting 1) the need to integrate diverse policies to manage pharmaceutical challenges effectively, based on the current state-of-the-art, and 2) that practical problems in this domain seem to lack a single policy class that yields universally acceptable performance.

4.1 Introduction

Pharmaceuticals are closely tied to patient health, highlighting the critical importance of accurate inventory control policies in supply chains. However, managing medical product inventories is challenging due to the interaction of multiple *factors*, including random yields, perishability, batching constraints, non-stationary demand caused by product life cycles, and lost sales [see also 30]. Although the impact of these factors in isolation is reasonably well understood [see, e.g., 6, 32, 31, 17], what is their relative importance, and how do they interact in real-world pharmaceutical supply chains? Moreover, while several policies in the literature address one of these challenges individually, is it clear how to adapt them to realistic supply chains that feature multiple overlapping challenges? What performance can be expected from different types of policies?

To address these questions, we developed a realistic case study in close collaboration with a senior supply chain manager from Bristol-Myers Squibb (BMS), a global manufacturer and distributor of medical products that faces typical pharmaceutical inventory challenges. The case study focuses on a specific product for which production yield uncertainty, product lifetime, and batching considerations are quantified using company data and expert input. BMS faces unique demand patterns for each product, necessitating an inventory management process that involves demand forecasting and timely ordering from manufacturing facilities. The case study incorporates a baseline policy based on expert human planners at BMS, who rely on forecasting models to predict demand uncertainty and fine-tune inventory levels to prevent and mitigate the risks of excess stock, product expiration, and lost sales. To explore the impact of demand uncertainty and non-stationarity, we used *synthetic data* from BMS regarding product life cycles to develop a realistic demand process covering the product's lifetime for 20 years and a variant with a shorter lifetime of 5 years. For parameters that are challenging to estimate accurately, *sensitivity analyses* were conducted over a broad range of values centered around company-provided data. This approach enabled the creation of a set of experiments that effectively represent the complexities of managing medical product inventories, including considerations of batching, product lifetime, and yield uncertainty. The resulting case study provides a robust foundation for answering key questions about inventory policies in pharmaceutical supply chains.

In detail, we contribute to the literature by adapting three general-purpose policies: *i)* an order up-to-level (OUT) policy, with a bounds-based search procedure to determine appropriate safety stock levels for non-stationary demand; *ii)* a projected inventory level (PIL) policy, which has been shown to perform well in settings involving lost sales and perishability, supported by a bounds-based procedure to optimize policy parameters; and *iii)* a deep reinforcement learning (DRL) approach, implemented via the proximal policy optimization (PPO) algorithm [29], which has been promoted as a general-purpose solution for inventory management, though its practical applications remain limited [4]. To successfully train the DRL algorithm, we introduce a novel method for designing features in the presence of non-stationarity, lost sales, and product expiration. Departing

from the standard practice of using the inventory vector directly as input to the neural network [see, e.g., 27, 15, 11, 22, 33, 34], we estimate future projected inventory levels for each age category based on the current state and use these estimates as input. Furthermore, we incorporate specific time-dependent demand features consistent with demand forecasts used by human planners, enabling PPO to account for the current life-cycle phase accurately.

In our experiments, we benchmark the performance of these three policies against a BMS baseline derived from human planners' actions. This comparison yields new and valuable insights into inventory management for medical products:

- OUT and PIL policies can be readily applied to *pharmaceutical supply chains* using the bounds-based optimization procedures developed in this paper, while PPO can be successfully implemented based on the designed features. All three policies demonstrate competitive performance.

- While there are considerable performance differences among the three solution approaches, none consistently outperforms the others. In fact, each policy (OUT, PIL, and PPO) is outperformed by more than 10% in specific, realistic experiments by one of the other policies. The OUT policy, in particular, can perform poorly when lost sales costs are high, with performance gaps exceeding 100%. These findings imply: *a)* Companies are advised to explore multiple policy types when addressing pharmaceutical inventory challenges and should *avoid relying solely on the OUT policy. b)* Despite decades of inventory research, practical inventory problems in this domain appear to *lack a single policy class that delivers universally acceptable performance*, let alone an interpretable, near-optimal policy.

- Although DRL algorithms are often promoted as general-purpose solutions for complex, realistic inventory problems [4, 40], we find PPO to be competitive but not consistently superior to classical policies across all experiments.

- The proposed OUT, PIL, and PPO policies can significantly reduce total company costs by maintaining lower inventory levels. However, although they outperform the human-driven policy in terms of average cost, they exhibit higher *performance variability*, which may render them less robust in certain scenarios. Our findings suggest that human planners focus on maintaining high service levels to avoid lost sales and ensure patient health, often achieving this by increasing safety stock levels. This practice reduces the risk of unsatisfied demand but results in higher holding costs. Enhancing the proposed policies with safeguards to address significant demand uncertainty could improve BMS results by balancing cost and service level requirements.

The remainder of this paper is structured as follows: Section 4.2 presents a comprehensive review of recent literature on classical policies and DRL. Section 4.3 outlines the

mathematical formulation of the pharmaceutical supply chain developed in collaboration with BMS, while Section 4.4 details and formalizes the OUT and PIL policies and describes the implementation of the PPO algorithm. Section 4.5 presents the results of numerical experiments evaluating the performance of the implemented policies. Finally, Section 4.6 concludes the paper.

4.2 Related Work

Our case study is motivated by a realistic inventory problem arising in pharmaceutical supply chains, which involves multiple factors, including perishability and yield uncertainty. While *random yield* is a well-studied problem in the literature when considered independently [32, 31, 3], its interaction with *perishability*, as observed in our real-world case study, has received limited attention. To position our work within these research streams, we review studies that address these factors in-depth, along with a concise overview of OUT and PIL policies and the application of DRL algorithms in inventory management.

4.2.1 Classical Policies

Perishable inventory systems have been extensively studied since the seminal work of [39]. A key challenge in managing such systems is tracking the age categories of stock to determine optimal order quantities. This process significantly increases the problem's dimensionality, making it computationally complex. Commonly used approaches, such as the OUT policy, are suboptimal because order quantities must consider the age distribution of existing stock [30]. [24] contributed by analyzing optimal policies for perishable systems with zero lead times, emphasizing the trade-off between holding and expiration costs. Managing perishable products remains challenging because the need to account for all age categories makes exact solutions through dynamic programming infeasible, especially when positive lead times are involved or product lifetimes exceed two periods. Consequently, much subsequent research has focused on developing heuristic or approximation-based approaches [10].

Early research by [26] pioneers near-myopic heuristics to address the computational challenges of dynamic programming in managing fixed-lifetime perishable products. The estimated withdrawal and aging (EWA) policy proposed by [6] represents a significant advancement. This policy accounts for positive lead times and varying demand patterns while estimating the quantity of expired items during lead times. By explicitly considering the impact of product expiration, the EWA policy significantly improves upon the standard OUT policy, resulting in better cost performance. Further exploration of the EWA policy by [18] highlights the benefits of incorporating estimated expired quantities into base-stock policies. Their findings suggest that adding expired estimates can significantly improve performance compared to traditional base-stock policies that

do not include such adjustments. Additionally, [17] examines supply chains for platelet concentrates, providing analytical approximations for key performance indicators such as expected on-hand inventory, order size, lost sales, and expired items. This research considers several complexities, including non-stationary demand with weekly variations and adjustable safety stock levels.

The *asymptotic optimality* of policy classes has received increasing attention in recent studies. [20] demonstrate that for non-perishable products in lost-sales settings, the base-stock (OUT) policy converges to optimality as penalty costs increase. For perishable products, [7] investigate the effectiveness of the OUT policy in lost-sales settings with no lead times and in backorder settings with positive lead times, establishing its asymptotic optimality as penalty or expiration costs grow. The PIL policy has primarily been studied in the context of asymptotic optimality. [21] analyze it for the standard lost-sales setting, providing results for long lead times and high penalty costs. Additionally, [8] examine perishable products with positive lead times, demonstrating that the PIL policy is optimal for single-period product lifetimes under bounded demand. For scenarios with unbounded demand and high penalty costs, the PIL policy still achieves optimality. [16] provides further insights into the concept of asymptotic optimality.

The main thrust of our paper is to evaluate classical policies and DRL in a pharmaceutical supply chain. Our *adaptations* of the classical OUT and PIL policies for this purpose build on the reviewed literature as follows. First, in contrast to the EWA policy examined by [6] and [17], which approximates expired stock using the mean demand value, our adaptation of the PIL policy also considers lost sales and uses the true distribution of demand to estimate expired stock more accurately. Additionally, in the context of the OUT policy with lost sales, perishable products, and positive lead times, we define both lower and upper bounds, heuristically addressing cases of non-stationary demand not analyzed by [7]. While their approach is practical in settings with backlogged demand or zero lead times, our case study focuses on settings involving lost sales and positive lead times. Lastly, we propose a bounds-based procedure to optimize the parameter of the PIL policy in the BMS case study, building on the work of [8] and establishing an effective procedure for accurately estimating the projected inventory level.

4.2.2 Deep Reinforcement Learning

[4] provide a comprehensive roadmap detailing the potential improvements that DRL can offer to inventory systems and policies. [27] extend the deep Q-network (DQN) algorithm to address the beer game problem, enabling the DQN algorithm to learn a near-optimal policy while other entities follow a base-stock policy. [15] implement and fine-tune the asynchronous advantage actor-critic (A3C) algorithm for multi-echelon systems, finding that A3C performs comparably to state-of-the-art heuristics and approximate dynamic programming algorithms.

To the best of our knowledge, no studies have assessed the ability of DRL to solve problems that combine multiple factors such as non-stationary demand, yield uncertainty,

and perishability–challenges commonly arising in real-world cases. Recently, [37] explored the use of DRL in industrial spare parts management. However, that case features neither perishability nor yield uncertainty. In the following, we briefly review DRL applications that incorporate at least one of these factors.

Non-stationarity is challenging to incorporate into classical policies, and DRL algorithms may offer significant advantages in such scenarios. [34] evaluate the performance of DRL algorithms and classical policies in multi-echelon systems with stochastic and seasonal demand, showing that the PPO algorithm performs better than other policies in a wide range of experiments. Similarly, [12] demonstrate that DRL, when supported by demand forecasting, can effectively learn non-stationary policies in supply chains characterized by fixed costs, lead times, and the presence of both backorders and lost sales. Their findings reveal that DRL can match or even surpass the performance of dynamic programming-based heuristics in certain scenarios. Van Hezewijk, Dellaert, and Van Jaarsveld [38] propose scalable DRL algorithms for a lot-sizing problem involving multiple products, where individual items exhibit non-stationarity. Additionally, Dijck et al. [13] apply DRL to a complex capacitated assembly system, showing that DRL is especially valuable in handling non-stationary demand.

A notable research gap exists in applying DRL in the domain of perishable products with fixed lifetimes. [36] develop DRL algorithms tailored to highly stochastic problems arising in operations management, testing their solution approach on three benchmark problems, one of which involves perishable products. [11] conduct a comparative analysis of the DQN algorithm against classical policies and other heuristics, consistently demonstrating that DQN outperforms alternatives across various experiments. Similarly, [1] investigate the use of DRL algorithms for perishable products, finding that these algorithms significantly reduce the likelihood of product stockouts and expirations, thereby enhancing service levels.

Our paper explores the application of DRL in inventory systems involving key factors such as perishability, yield uncertainty, and non-stationary demand–among others–that are critical in pharmaceutical supply chains and are rarely addressed together. Unlike prior studies, which typically examine these factors in isolation, we focus on integrating them into a realistic case study. By leveraging future projected inventory levels as features, our approach bridges theoretical insights and practical implementation, offering a novel method for managing complex inventory problems.

4.3 Case Study Description

The pharmaceutical supply chain analyzed in this paper is modeled based on a *real-world case study*. One of the authors collaborated with a senior supply chain manager at BMS to obtain a validated model of the inventory system and access a realistic demand-generating process for a specific perishable pharmaceutical drug. The supply chain manager also provided key operational parameter values, such as yield uncertainty,

product lifetime, and lead time, as discussed in detail below. Using company data and expert input, we derived a validated set of parameters that produces a realistic model for assessing state-of-the-art inventory policies in the context of challenges typically arising in pharmaceutical supply chains. This section describes the base case, while Section 4.5 explores variants of this case to gain deeper insights into the results.

The modeled supply chain environment is structured as a finite-horizon, periodic-review system over T timesteps ($t = 1, \dots, T$), directly linking a third-party factory with a storage warehouse, as represented in Figure 4.1. We consider a single product type produced by the factory. Product batches may experience up to 10% of items lost due to yield uncertainty during production. In our base case, BMS assumes that yield loss is uniformly distributed between 0 and 10% of each order. This assumption captures the inherent variability in production, enabling us to explore a broader range of yield rate values.

Supply chains commonly operate with planned lead times to provide the factory with sufficient time for production planning [see, e.g., 35]. At BMS, a positive lead time of $L = 12$ timesteps is assumed, with items delivered in batches of $Q = 20$ units and a maximum of $n = 6$ batches per shipment. The company employs a dynamic batch ordering cost $K(q_t)$, which varies with the batch size to account for economies of scale, along with a static unit ordering cost $\hat{c}$ proportional to the number of items ordered.

Other parameters are similarly based on discussions with expert human planners at BMS. Upon receipt at the warehouse, each item has a product lifetime of $m = 12$ timesteps, after which an expiration cost of $\hat{w} = 3.0$ per unit is incurred. The warehouse is assumed to have unlimited storage capacity, with a holding cost of $\hat{h} = 1.0$ per unit per timestep. A lost sales cost of $\hat{b} = 100$ per unit is applied as a penalty when demand exceeds on-hand inventory. Without loss of generality, we assume that any remaining items at the end of the horizon are salvaged at a value of $\hat{c}$ per unit.

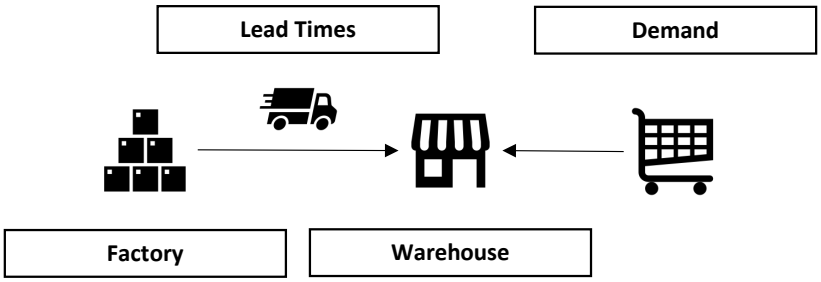


Figure 4.1: Representation of the supply chain environment.

Demand across timesteps follows a non-stationary process. For each timestep t, let D_t represent the one-period demand, with $\mathbb{E}[D_t] < +\infty$. The demand D_t at each timestep t is modeled as $D_t = d_t + \xi_t$, where d_t represents deterministic values provided by BMS that define the demand forecast, and ξ_t denotes random variables with a mean of zero, representing the forecast error. The non-stationarity arises because the standard

deviation of ξ_t, denoted as σ_t, varies across timesteps. BMS provides forecast errors as percentages of the mean demand, which are used to compute the time-dependent standard deviation σ_t for each timestep. The distribution of ξ_t is not necessarily independent and identically distributed (i.i.d.), further capturing the non-stationarity of the forecast error.

4.3.1 Order of Events

As illustrated in Figure 4.2, the order of events in the supply chain environment and their associated costs for each timestep t are defined as follows:

1. The order q_{t-L} placed at timestep $t - L$ arrives at the warehouse.
2. A new order q_t is placed at the factory, incurring ordering costs.
3. Demand D_t is satisfied from the on-hand inventory in the warehouse. Any unsatisfied demand is tracked as lost sales.
4. Costs for timestep t–including ordering, lost sales, expiration, and holding costs–are computed. Expired items are removed from the on-hand inventory at the end of the timestep, incurring expiration costs. The remaining items with usable lifetimes are moved forward to the next timestep, resulting in holding costs.

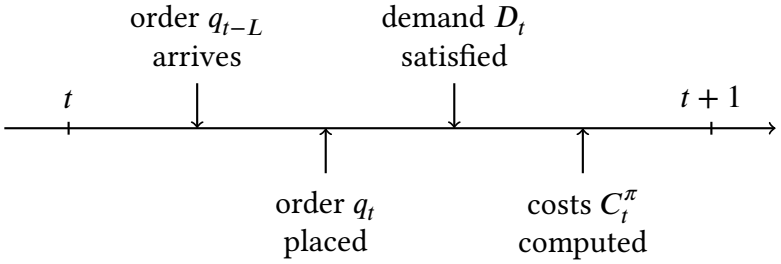


Figure 4.2: Order of events in the supply chain environment.

Let $\mathbf{x}_t := (x_{t,1}, x_{t,2}, \dots, x_{t,m}, \dots, x_{t,m+L-1})$ be the vector representing the inventory in transit and on hand, where $x_{t,i}$ denotes the quantity in transit or on hand with a remaining lifetime of i, for $i = 1, \dots, m+L-1$. Let Z_t be a random variable representing the fractional yield for the order arriving at timestep t with realization z_t. The maximum production loss, denoted by $\hat{z}$, is set at 10%, resulting in $Z_t = 1 - \mathcal{U}(0,1) \times \hat{z}$, where $\mathcal{U}(0,1)$ is a random variable uniformly distributed on the interval $[0,1]$. This yield effectively captures the variability in the drug production process, ensuring it fluctuates between 90% and 100%, as indicated by BMS. Consequently, the order arriving at timestep $t+1$, denoted by $x_{t+1,m}$, is updated based on the yield rate Z_t and the previous order $x_{t,m+1}$.

The full *inventory vector* $\mathbf{x}_t$ is then updated as follows:

$$x_{t+1,i} = \begin{cases} q_t, \; i = m+L-1, \; x_{t,i+1}, \; i = m+1, \dots, m+L-2, \\ \left(Z_t x_{t,i+1} - \left(D_t - \sum_{j=1}^{i} x_{t,j} \right)^+ \right)^+, \; i = m, \\ \left(x_{t,i+1} - \left(D_t - \sum_{j=1}^{i} x_{t,j} \right)^+ \right)^+, \; i = 1, \dots, m-1. \end{cases} \tag{4.1}$$

This update implies that for the inventory at position $i = m + L - 1$, the vector is updated according to the current order quantity q_t placed at timestep t. For intermediate positions, from $i = m + 1$ to $i = m + L - 2$, representing the pipeline of inventory in transit, the vector is shifted by one position. For positions from $i = 1$ to $i = m$, the vector is updated to satisfy the demand D_t using the on-hand inventory, following a FIFO issuing policy from the oldest to the most recent items. Inventory with a lifetime of $i = m$ (i.e., items that have just arrived) directly incorporates the yield rate Z_t. Here, $(\cdot)^+$ denotes $\max(0, \cdot)$.

Let X_t and Y_t denote the random variables representing the total inventory level at timestep t before order arrival and after order arrival but before demand satisfaction, respectively. Their realizations are defined as $x_t = \sum_{i=1}^{m-1} x_{t,i}$ and $y_t = x_t + z_t q_{t-L}$, and the expired quantity at timestep t is given by $O_t = (x_{t,1} - D_t)^+$.

The *total cost* over a finite episode horizon T for a given policy $\pi \in \Pi$, where a policy maps states to actions and Π represents the set of feasible policies, can be expressed as:

$$C^\pi = \mathbb{E}\left[\sum_{t=1}^{T} C_t^\pi - \hat{c} \sum_{i=1}^{m-1} x_{T+1,i} \right], \tag{4.2}$$

where $C_t^\pi = K(q_t) + \hat{c} q_t + \hat{h}(Y_t - D_t)^+ + \hat{b}(D_t - Y_t)^+ + \hat{w} O_t$ and $K(q_t) = \{0, 5, 8, 9, 10\}$, for $n_t = 0, 1, 2, 3, \{4, 5, 6\}$, respectively, with $n_t = \lceil \frac{q_t}{Q} \rceil$ and $\lceil \cdot \rceil$ represents the ceiling function. The first term in C_t^π represents the batch ordering cost, conditional on the batch size, while the second term is the unit ordering cost proportional to the number of items. The subsequent terms denote the holding, lost sales, and expiration costs at timestep t, respectively.

Given our objective to determine an appropriate order quantity q_t at each timestep t, while minimizing the total cost C^π across the episode horizon T, we assume $q_t = 0$ for $t = T - L + 1, \dots, T$, as these orders will not arrive before the episode ends. Additionally, at the start of each episode, the inventory level is assumed to be empty, i.e., $x_{1,i} = 0$ for $i = 1, \dots, L - 1$. Table 4.1 summarizes the notation used throughout the paper.

4.3.2 Cost Transformation

Following a methodology similar to that described in [10], we transform the total cost C^π into an *equivalent expression* that excludes the unit ordering cost $\hat{c}$. The following lemma facilitates this transformation.

Table 4.1: Notation for the supply chain environment with their explanations (and units of measure).

Parameter	Explanation	Parameter	Explanation
$K(q_t)$	Batch Ordering Cost (per batch)	T	Episode Horizon (timesteps)
$\hat{c}$	Unit Ordering Cost (per unit)	L	Lead Time (timesteps)
$\hat{w}$	Expiration Cost (per unit)	m	Product Lifetime (timesteps)
$\hat{b}$	Lost Sales Cost (per unit per timestep)	$\hat{z}$	Production Yield (percentage per batch)
$\hat{h}$	Holding Cost (per unit per timestep)	D_t	Demand (units)

Lemma 1. *The total cost defined in Equation 4.2 can be expressed as:*

$$C^{\pi} = \mathbb{E}\left[\sum_{t=1}^{T} K(q_t) + h(Y_t - D_t)^+ + b(D_t - Y_t)^+ + wO_t\right] + \hat{c}\sum_{t=L+1}^{T}\mathbb{E}[D_t], \tag{4.3}$$

where $h = \hat{h}$, $b = \hat{b} - \hat{c}$, *and* $w = \hat{w} + \hat{c}$.

Proof. The proof is provided in Appendix C.1. □

From Lemma 1, it follows that finding an optimal policy π minimizing C^{π} is equivalent to finding the same optimal policy for $C^{\pi} - \hat{c}\sum_{t=L+1}^{T}\mathbb{E}[D_t]$. Consequently, the optimal policy can be defined as $OPT = \inf_{\pi\in\Pi} C^{\pi}$. In the subsequent sections, we focus on optimizing the transformed cost C^{π}.

4.3.3 Dynamic Programming Formulation

Under the lost sales assumption, determining the optimal order quantity q_t using dynamic programming depends on the inventory vector $\mathbf{x}_t$. Let $V_t(\mathbf{x}_t)$ denote the cost-to-go function from timestep t to T:

$$V_t(\mathbf{x}_t) = \min \mathbb{E}\left[K(q_t) + h(Y_t - D_t)^+ + b(D_t - Y_t)^+ + wO_t + V_{t+1}(\mathbf{x}_{t+1})\right]. \tag{4.4}$$

Solving this dynamic programming model poses significant challenges due to the *curse of dimensionality*, primarily caused by the positive lead time L and the product lifetime m. For instance, [14] demonstrated that even with a lead time of zero, determining the optimal order quantity for products with a lifetime of at least five timesteps requires substantial computational effort. Introducing a lead time of 12 timesteps, combined with a product lifetime of 12 timesteps–as in our real-world case study–further exacerbates the complexity, making it computationally intractable to find an exact solution within a reasonable timeframe. This challenge persists even for non-perishable products, as solving Equation 4.4 is further complicated by the lost sales assumption, as highlighted in prior studies [23, 2].

As expressed in Lemma 1, the unit ordering cost $\hat{c}$ is incorporated into the holding, penalty, and expiration costs, allowing us to focus on solving the transformed total

cost C^{π}. However, the batch ordering cost represented by $K(q_t)$ introduces additional complexities. Developing effective policies for perishable products becomes particularly difficult when such costs are present [12]. As a result, the proposed policies must be carefully evaluated to ensure their effectiveness in addressing these challenges.

4.4 Inventory Policies

This section analyzes two primary *types of inventory policies* implemented under the specific assumptions and challenges of our supply chain environment: classical (analytic) policies and DRL policies. For the classical policies, we define the widely known and adopted OUT policy (Section 4.4.1) and describe the baseline policy currently used at BMS, which refines the OUT policy by incorporating expected expiration in the spirit of the EWA policy (Section 4.4.2). Additionally, we discuss the PIL policy applied in our case study (Section 4.4.3). Lastly, for the DRL policy, we introduce the PPO algorithm (Section 4.4.4). Our analysis outlines the fundamental principles of each policy type, investigates specific theoretical properties, and discusses their practical implementation.

4.4.1 OUT Policy

To implement the OUT policy in our non-stationary case, we propose fixing the safety stock instead of the order-up-to level. This approach ensures that the order-up-to level adjusts dynamically based on the expected demand during lead times, allowing it to increase or decrease in response to fluctuations in demand forecasts. Specifically, the proposed OUT policy is characterized by a single parameter, s, representing the *safety stock*. Denoted by π_s, this policy involves ordering up to the *order-up-to level* $S_t = s + \sum_{j=t}^{t+L} d_j$ at each timestep t. Accordingly, the order quantity $q_t^{\pi_s}$ at timestep t is calculated as:

$$q_t^{\pi_s} = \left(S_t - \sum_{i=1}^{m+L-1} x_{t,i} \right)^+, \tag{4.5}$$

and the number of batches is determined as: $n_t^{\pi_s} = \left\lceil \frac{q_t^{\pi_s}}{Q} \right\rceil$.

The expected total cost over the episode horizon T is given by:

$$C^{\pi_s} = \mathbb{E}\left[\sum_{t=1}^{T} (h(Y_t - D_t)^+ + b(D_t - Y_t)^+ + wO_t) \right]. \tag{4.6}$$

This formulation implies that the order-up-to level equals $s + \sum_{j=t}^{t+L} d_j$, which varies dynamically at each timestep t to account for non-stationary demand. Here, $\sum_{i=1}^{m+L-1} x_{t,i}$ represents the inventory level at timestep t prior to ordering. Given the positive lead

time L, the expected total cost can also be expressed as:

$$C^{\pi_s} = \sum_{t=1}^{T-L} \left(h\mathbb{E}\left[\left(Y_{t+L} - D_{t+L}\right)^+\right] + b\mathbb{E}\left[\left(D_{t+L} - Y_{t+L}\right)^+\right] + w\mathbb{E}\left[\left(x_{t+L,1} - D_{t+L}\right)^+\right]\right). \tag{4.7}$$

Although the demand process is non-stationary due to time-varying forecast errors, we adopt a constant safety stock s to balance policy simplicity and interpretability. The proposed OUT policy remains responsive to evolving demand through the dynamic order-up-to level S_t, which adjusts at each timestep t based on demand forecasts. The fixed parameter s serves as a time-invariant safeguard against forecast uncertainty over the lead time, while its optimization over the entire episode horizon implicitly captures variations in forecast accuracy. This structure offers a practical alternative to time-varying safety stocks, which would considerably increase policy complexity [28]. The latter approach is ideally recommended in the literature for non-stationary settings since the safety stock should vary over time in such cases, and it can represent a natural extension for future work.

Our goal is to determine the optimal safety stock s for the OUT policy under lost sales. To achieve this, we propose a bounds-based search procedure that leverages approximative assumptions and simplifications to establish bounds and then employs a bounds-based procedure to optimize policy parameters. Specifically, we assume that the random variables representing forecast error, ξ_t, are i.i.d. and follow a normal distribution with a mean of zero and a standard deviation σ. Under this assumption, the cumulative forecast error over the lead time is defined as $\mathscr{D}_{L+1} = \sum_{j=t}^{t+L} \xi_j$. Two simplifications are also made to facilitate the derivation of bounds. First, batch ordering costs, $K(q_t)$, are omitted. Prior studies, such as [25] and [41], analyzed fixed ordering costs in perishable systems with fixed lifetimes but limited their focus to scenarios with zero lead time. In pharmaceutical supply chains, transportation costs are typically shared across multiple products, making fixed ordering costs negligible. Second, yield uncertainty is excluded from deriving theoretical bounds to maintain tractability. Given the complexities of deriving closed-form expressions for costs, we adopt *Monte Carlo simulation* to determine the optimal safety stock for the OUT policy. In particular, we generate 2000 simulated episodes, each covering the entire episode horizon, and set the safety stock to minimize the average total cost across these simulated episodes. The effectiveness of this bounds-based procedure will be validated in Section 4.5.

OUT Policy Lower Bound

The following lemma establishes a lower bound for the cost component of the OUT policy.

Lemma 2. *Under the OUT policy with lost sales, for each demand sample path and every*

timestep $t \geq L+1$, the following conditions hold:

$$\begin{cases} \mathbb{E}\left[O_t\right] \geq \frac{\mathbb{E}\left[\left(s-\mathscr{D}_{m+L}\right)^+\right]}{m+L} \\ \mathbb{E}\left[\left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+\right] \geq \mathbb{E}\left[\left(s-\mathscr{D}_{L+1}\right)^+\right] - \frac{L\mathbb{E}\left[\left(s-\mathscr{D}_{m+L}\right)^+\right]}{m+L} \\ \mathbb{E}\left[\left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+\right] \geq \frac{\mathbb{E}\left[\left(\mathscr{D}_{L+1}-s\right)^+\right]}{L+1} \end{cases} . \tag{4.8}$$

Proof. The proof is provided in Appendix C.2. □

This lemma enables us to derive a lower bound, denoted by LB, on the total cost:

$$\begin{aligned} C^{\pi_s} &\geq LB(s) \\ &= T\left(h\mathbb{E}\left[\left(s-\mathscr{D}_{L+1}\right)^+\right] + \frac{b}{L+1}\mathbb{E}\left[\left(\mathscr{D}_{L+1}-s\right)^+\right] + \frac{w-hL}{m+L}\mathbb{E}\left[\left(s-\mathscr{D}_{m+L}\right)^+\right]\right). \end{aligned} \tag{4.9}$$

OUT Policy Upper Bound

We next derive an upper bound, UB, under the assumption that items are non-perishable ($O_t = 0$ for $m = +\infty$ and any $t > L$):

$$\begin{aligned} \left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+ &\leq \left(\sum_{j=t-L}^{t-1} l_j\right) + \left(s - \sum_{j=t-L}^{t} \xi_j\right)^+, \\ \left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+ &\leq \left(\sum_{j=t-L}^{t} \xi_j - s\right)^+. \end{aligned} \tag{4.10}$$

Taking the expectation of the above inequalities yields the upper bound:

$$UB(s) = T\left(h\mathbb{E}\left[\left(s-\mathscr{D}_{L+1}\right)^+\right] + (b+hL)\mathbb{E}\left[\left(\mathscr{D}_{L+1}-s\right)^+\right]\right). \tag{4.11}$$

This bound follows directly from the assumption that ξ_t is stationary and i.i.d., which implies that $\sum_{j=t-L}^{t} \xi_j$ and $\sum_{j=t}^{t+L} \xi_j$ share the same distribution.

We proceed by determining the solution for the upper bound, denoted as s^*_{UB}, which corresponds to the $\frac{b+hL}{b+(L+1)h}$ quantile of the forecast error distribution:

$$s^*_{UB} = \inf\left\{ s \,:\, \mathbb{P}(\mathscr{D}_{L+1} \leq s) \geq \frac{b+hL}{b+(L+1)h} \right\}. \tag{4.12}$$

The upper bound is expected to act as a cap on the order-up-to level of the optimal perishable OUT policy under lost sales. This expectation is based on the formal demonstration provided by [26] for scenarios with zero lead time and further supported by findings for positive lead times in non-perishable systems by [20]. However, providing formal

proof of this conjecture in the context of perishable products remains a challenging open problem.

Finally, given the complexities of deriving closed-form expressions for costs, we adopt *Monte Carlo simulation* to determine the optimal safety stock for the OUT policy. Specifically, we generate 2000 simulated episodes, each covering the entire episode horizon, and set the safety stock to minimize the average total cost across these simulated episodes.

4.4.2 BMS Baseline Policy

We next discuss the BMS baseline policy, which is derived from the expertise of human planners. Similar to the OUT policy, this human-driven policy relies on a time-dependent target level, defined as:

$$\tilde{S}_t = \tilde{s} + \sum_{j=t}^{t+L} d_j, \tag{4.13}$$

where $\tilde{s}$ represents the safety stock.

Accordingly, the BMS baseline orders the quantity $n_t^{\pi_{\tilde{s}}} = \left\lceil \frac{q_t^{\pi_{\tilde{s}}}}{Q} \right\rceil$ at each timestep t, where:

$$q_t^{\pi_{\tilde{s}}} = \left(\tilde{S}_t - \left(\sum_{i=1}^{m+L-1} x_{t,i} - \sum_{j=t}^{t+L-1} \hat{O}_j | \mathbf{x}_t \right) \right)^+ . \tag{4.14}$$

This expression does not account for lost sales occurring during the L timesteps and requires the computation of $\sum_{j=t}^{t+L-1} \hat{O}_j$, which represents a *heuristic estimate of the expired quantity* from timestep t to $t+L-1$, based on the assumption that demand equals its expected value.

In particular, let $\hat{O}_{[t:s]}$ denote the cumulative estimated expired quantity from timestep t to $s \geq t$. According to Equation 3 in [9], this can be computed as:

$$\hat{O}_{[t:s]}(\mathbf{x}_t) = \sum_{j=t}^{s} \hat{O}_j = \max \left\{ \sum_{i=1}^{s-t+1} x_{t,i} - d_{[t:s]}, \hat{O}_{[t:s-1]}(\mathbf{x}_t) \right\}, \tag{4.15}$$

with the convention that $\hat{O}_{[t:-1]}(\mathbf{x}_t) \equiv 0$. This approach is related to the EWA policy proposed by [6], but BMS applies a specific procedure for determining the safety stock.

BMS Baseline Policy Safety Stock

The safety stock for the BMS baseline policy is determined using a standard safety stock factor k_1 to achieve a 99% service level, assuming the forecast error ξ_t follows a normal distribution. Additionally, BMS empirically introduces a second safety stock factor k_2 to

address the uncertainty in production yield. Based on the mean squared error (MSE) dynamically calculated at each timestep t, the safety stock is derived as:

$$\tilde{s} = k_1 \sqrt{(L+1) \cdot MSE} + k_2. \tag{4.16}$$

Note that the MSE, and consequently the safety stock $\tilde{s}$, depends on the quality of the demand forecast. Unlike the OUT policy, which determines the safety stock through cost-based optimization, the BMS baseline employs an internal *fixed rule* for safety stock determination.

4.4.3 PIL Policy

The foundation of the PIL policy has been explored by [21] and further developed by [8] to address perishable products with fixed lifetimes, demonstrating its effectiveness compared to constant-order and OUT policies. However, neither study has examined its application under non-stationary demand. This paper extends the analysis of [8] to investigate such scenarios. In the original formulation of the PIL policy, [21] propose placing an order to raise the *projected inventory level* to a desired target. In our extension, we adopt a slightly modified approach to account for the complexities introduced by non-stationary demand.

Furthermore, both the BMS baseline and EWA policies account for expirations during lead times when determining the size of orders to be placed but ignore lost sales in the computation of Equation 4.14, treating excess demand as backorders. These policies also assume that forecast errors follow a normal distribution and rely solely on mean demand to estimate expirations, disregarding any associated uncertainty. To address these limitations, we propose incorporating the expected inventory level at timestep $t + L$ into the PIL policy.

Let $x_{t+L} = \sum_{i=1}^{m-1} x_{t+L,i}$ denote the inventory level before ordering at timestep $t + L$, and let π_u represent the PIL policy, which involves ordering at each timestep t up to:

$$U_t = u + d_{t+L}, \tag{4.17}$$

where u represents the safety stock employed by the PIL policy. By the *material conservation law*, the inventory level can be expressed as:

$$x_{t+L} = \sum_{i=1}^{m-1} x_{t+L,i} = \sum_{i=1}^{m+L-1} x_{t,i} - \sum_{j=t}^{t+L-1} D_j + \sum_{j=t}^{t+L-1} l_j - \sum_{j=t}^{t+L-1} O_j, \tag{4.18}$$

where $l_t = (D_t - \sum_{i=1}^{m} x_{t,i})^+$.

According to Lemma 1 of [8], it is always possible to order a quantity that brings the expected inventory level at timestep $t + L$ to U_t, denoted as $\mathbb{E}\left[\sum_{i=1}^{m} x_{t+L,i}\right] = U_t$. By

Equation 4.18, the order quantity can be expressed as:

$$q_t^{\pi_u} = \left(U_t - \mathbb{E}\left[\sum_{i=1}^{m-1} x_{t+L,i} | \mathbf{x}_t \right] \right)^+ = \left(u + \sum_{j=t}^{t+L} d_j - \sum_{i=1}^{m+L-1} x_{t,i} + \mathbb{E}\left[\sum_{j=t}^{t+L-1} O_j - l_j \right] \right)^+ . \tag{4.19}$$

Therefore, the optimal order quantity, accounting for cumulative lost sales over L timesteps, is given by:

$$n_t^{\pi_u} = \left\lceil \frac{q_t^{\pi_u}}{Q} \right\rceil . \tag{4.20}$$

In the following subsections, we derive the lower and upper bounds for the total cost associated with the PIL policy.

PIL Policy Lower Bound

For any feasible policy π, the expired quantity at timestep $t+m-1$ can be expressed as:

$$\begin{aligned} O_{t+m-1} &= \left(\sum_{i=1}^{m+L} x_{t-L,i} - \sum_{j=t-L}^{t+m-1} D_j + \sum_{j=t-L}^{t+m-1} l_j - \sum_{j=t-L}^{t+m-2} O_j \right)^+ \\ &\geq \left(\sum_{i=1}^{m+L} x_{t-L,i} - \sum_{j=t-L}^{t+m-1} D_j \right)^+ - \sum_{j=t-L}^{t+m-2} O_j, \end{aligned} \tag{4.21}$$

where $\sum_{i=1}^{m+L} x_{t-L,i} = s + \sum_{j=t-L}^{t} d_j$ for the OUT policy with an order-up-to level $s + \sum_{j=t-L}^{t} d_j$.

Applying the equality in Equation 4.21 to the OUT policy gives:

$$\sum_{j=t-L}^{t+m-1} O_j \geq \left(s - \sum_{j=t-L}^{t+m-1} \xi_j - \sum_{j=t+1}^{t+m-1} d_j \right)^+ \geq \left(s - \sum_{j=t-L}^{t+m-1} \xi_j \right)^+ - \sum_{j=t+1}^{t+m-1} d_j. \tag{4.22}$$

This expression can also be derived as:

$$\begin{aligned} O_{[t,t+m+L-1]}(\mathbf{x}_t) = \sum_{j=t}^{t+m+L-1} O_j &= \max \left\{ \sum_{i=1}^{m+L} x_{t,i} - D_{[t,t+m+L-1]}, O_{[t,t+m+L-2]}(\mathbf{x}_t) \right\} \\ &\geq \left(\sum_{i=1}^{m+L} x_{t,i} - \sum_{j=t}^{t+m+L-1} D_j \right)^+ . \end{aligned} \tag{4.23}$$

Taking the expectation of the left- and right-hand sides yields:

$$(m+L)\mathbb{E}[O_t] \geq \mathbb{E}\left[\left(s - \mathscr{D}_{m+L} \right)^+ \right] - \sum_{j=t+1}^{t+m-1} d_j. \tag{4.24}$$

Similarly,

$$\begin{aligned}\left(\sum_{i=1}^{m} x_{t+L,i} - D_{t+L}\right)^+ &= \left(\sum_{i=1}^{m+L} x_{t,i} - \sum_{j=t}^{t+L} D_j + \sum_{j=t}^{t+L-1} l_j - \sum_{j=t}^{t+L-1} O_j\right)^+ \\ &\geq \left(\sum_{i=1}^{m} x_{t+L,i} - \sum_{j=t}^{t+L} D_j\right)^+ - \sum_{j=t}^{t+L-1} O_j.\end{aligned} \tag{4.25}$$

For the OUT policy with an order-up-to level $s + \sum_{j=t}^{t+L} d_j$, this implies:

$$\mathbb{E}\left[\left(\sum_{i=1}^{m} x_{t+L,i} - D_{t+L}\right)^+\right] \geq \mathbb{E}\left[\left(s - \mathscr{D}_{L+1}\right)^+\right] - L\mathbb{E}[O_t], \tag{4.26}$$

and

$$\begin{aligned}\left(D_{t+L} - \sum_{i=1}^{m} x_{t+L,i}\right)^+ &= \left(\sum_{j=t}^{t+L} D_j - \sum_{j=t}^{t+L-1} l_j + \sum_{j=t}^{t+L-1} O_j - \sum_{i=1}^{m+L} x_{t,i}\right)^+ \\ &\geq \left(\sum_{j=t}^{t+L} D_j - \sum_{i=1}^{m+L} x_{t,i}\right)^+ - \sum_{j=t}^{t+L-1} l_j,\end{aligned} \tag{4.27}$$

Thus:

$$(L+1)\mathbb{E}\left[\left(D_{t+L} - \sum_{i=1}^{m} x_{t+L,i}\right)^+\right] \geq \mathbb{E}\left[\left(\mathscr{D}_{L+1} - s\right)^+\right]. \tag{4.28}$$

Combining the above inequalities, we can establish a bound for the total cost as follows:

$$\begin{aligned}C^{\pi_u} &= \mathbb{E}\left[\sum_{t=1}^{T}(h(Y_t - D_t)^+ + p(D_t - Y_t)^+ + wO_t)\right] \\ &\geq \sum_{t=1}^{T}\Bigg(h\mathbb{E}[(s - \mathscr{D}_{L+1})^+] + \frac{p}{L+1}\mathbb{E}[(\mathscr{D}_{L+1} - s)^+] \\ &\quad + \frac{w - hL}{m+L}\mathbb{E}[(s - \mathscr{D}_{m+L})^+] - \frac{w - hL}{m+L}\sum_{j=t+1}^{t+m-1} d_j\Bigg).\end{aligned} \tag{4.29}$$

Thus:

$$\begin{aligned}C^{\pi_u} &= \mathbb{E}\left[\sum_{t=1}^{T}(h(Y_t - D_t)^+ + p(D_t - Y_t)^+ + wO_t)\right] \\ &\geq T\Bigg(h\mathbb{E}[(s - \mathscr{D}_{L+1})^+] + \frac{p}{L+1}\mathbb{E}[(\mathscr{D}_{L+1} - s)^+] \\ &\quad + \frac{w - hL}{m+L}\mathbb{E}[(s - \mathscr{D}_{m+L})^+] - \frac{w - hL}{m+L}\sum_{t=1}^{T}\sum_{j=t+1}^{t+m-1} d_j\Bigg).\end{aligned} \tag{4.30}$$

PIL Policy Upper Bound

We conclude this subsection by providing an upper bound on the total cost for the PIL policy. Based on the findings of [8], we note that Lemmas 2 and 4 remain applicable when demand is non-stationary. Specifically, by their Lemma 2, we have $\mathbb{E}\left[\sum_{i=1}^{m} x_{t+L,i}\right] = U_{t+L} = u + d_{t+L}$, and by the material conservation law, we can express:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{m} x_{t,i}\right] = d_t + u &= \mathbb{E}\left[\sum_{i=1}^{m+L-1} x_{t-L,i} + q_{t-L} - \sum_{j=t-L}^{t-1}\left(D_j - l_j + O_j\right)\right] \\ &= \sum_{i=1}^{m+L-1} x_{t-L,i} + q_{t-L} - \sum_{j=t-L}^{t-1} d_j - \mathbb{E}\left[\sum_{j=t-L}^{t-1}\left(O_j - l_j\right)\right].\end{aligned} \tag{4.31}$$

This implies:

$$\sum_{i=1}^{m+L-1} x_{t-L,i} + q_{t-L} = u + \sum_{j=t-L}^{t} d_j + \mathbb{E}\left[\sum_{j=t-L}^{t-1}\left(O_j - l_j\right)\right]. \tag{4.32}$$

By Lemma 4 (part b) of [8], the variables $-\sum_{j=t-L}^{t-1} O_j$, $\sum_{j=t-L}^{t-1} l_j$, and $\sum_{j=t-L}^{t-1}(D_j - l_j + O_j)$ are component-wise increasing in D_j, $j = t, \ldots, t+L-1$. We thus define the following associated random variables:

$$\begin{aligned}A_t &= \sum_{j=t-L}^{t-1}\left(-l_j + O_j + \xi_t\right) - \mathbb{E}\left[\sum_{j=t-L}^{t-1}\left(O_j - l_j\right)\right] - \left(d_t + u\right) + d_t + \xi_t, \\ B_t &= \sum_{j=t-L}^{t-1} l_j - \mathbb{E}\left[\sum_{j=t-L}^{t-1} l_j\right], \\ R_t &= \mathbb{E}\left[\sum_{j=t-L}^{t-1} O_j\right] - \sum_{j=t-L}^{t-1} O_j.\end{aligned} \tag{4.33}$$

Since $\mathbb{E}\left[B_t\right] = 0$ and $\mathbb{E}\left[R_t\right] = 0$, by Lemma 7 of [21], we can write:

$$\begin{aligned}\mathbb{E}\left[\left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+\right] &= \mathbb{E}\left[\left(A_t\right)^+\right] \le \mathbb{E}\left[\left(A_t + B_t\right)^+\right] \le \mathbb{E}\left[\left(A_t + B_t + R_t\right)^+\right] \\ &\le \mathbb{E}\left[\left(\sum_{j=t-L}^{t} \xi_j - u\right)^+\right] = \mathbb{E}\left[\left(\mathscr{D}_{L+1} - u\right)^+\right],\end{aligned} \tag{4.34}$$

and

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+\right] &= \mathbb{E}\left[\left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+ + \sum_{i=1}^{m} x_{t,i} - D_t\right] \\
&= \mathbb{E}\left[\left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+\right] + \mathbb{E}\left[\sum_{i=1}^{m} x_{t,i}\right] - \mathbb{E}\left[D_t\right] \\
&\leq \mathbb{E}\left[\left(\sum_{j=t-L}^{t} \xi_j - u\right)^+\right] + u \\
&= \mathbb{E}\left[\left(\sum_{j=t-L}^{t} \xi_j - u\right)^+ + u - \sum_{j=t-L}^{t} \xi_j\right] \\
&= \mathbb{E}\left[\left(u - \sum_{j=t-L}^{t} \xi_j\right)^+\right] = \mathbb{E}\left[\left(u - \mathscr{D}_{L+1}\right)^+\right].
\end{aligned} \tag{4.35}$$

Thus, we can bound the expected expired quantity as:

$$\sum_{j=t}^{m+t-1} O_j \leq \left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+. \tag{4.36}$$

Taking the expectation of the left- and right-hand sides yields:

$$m\mathbb{E}\left[O_t\right] \leq \mathbb{E}\left[\left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+\right] \leq \mathbb{E}\left[\left(u - \sum_{j=t-L}^{t} \xi_j\right)^+\right] = \mathbb{E}\left[\left(u - \mathscr{D}_{L+1}\right)^+\right]. \tag{4.37}$$

Combining the inequalities above, we can finally upper bound the total cost of the PIL policy as:

$$C^{\pi_u} \leq T\left((h + \frac{w}{m})\mathbb{E}\left[\left(u - \mathscr{D}_{L+1}\right)^+\right] + b\mathbb{E}\left[\left(\mathscr{D}_{L+1} - u\right)^+\right]\right). \tag{4.38}$$

PIL Implementation

The proposed PIL policy requires estimating the expected projected inventory level L timesteps into the *future* to calculate the order quantity. This approach involves computing the expected lost sales and expired stock over L timesteps. For this purpose, we adopt a methodology aligned with [41] and [7]. Similar to the OUT policy, the safety stock employed by the PIL policy is determined using Monte Carlo simulations tailored to its specific requirements (see Section 4.4.1 for details).

To estimate the expected inventory level at timestep $t+L$, given the current inventory vector (i.e., $\mathbb{E}\left[\sum_{i=1}^{m} x_{t+L,i} \mid \mathbf{x_t}\right]$), we perform 2000 simulated episodes over the entire episode horizon T, generating 2000 demand sample paths for each timestep t. Specifically,

following the order of events defined in our supply chain environment (Section 4.3), at each timestep t, we: *i)* set the actual order quantity n_t to zero, as it does not affect the computation of the expected inventory level at timestep $t + L$ but impacts timestep $t + L + 1$; *ii)* update the total inventory level based on received inventory in transit, as represented in Equation 4.1; and *iii)* track any potential lost sales and expired stock. Upon completing the simulations for each timestep within the interval from t to $t + L$, we compute the average amounts of expired stock and lost sales over this interval. These averages provide the expected inventory level at timestep $t + L$ necessary for determining the order quantity n_t, as expressed in Equations 4.19 and 4.20.

4.4.4 PPO Algorithm

PPO is an actor-critic algorithm designed to enhance the efficiency and stability of policy updates [29]. It modifies the objective function to prevent large updates to the policy by introducing a *clipped objective function*:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right], \tag{4.39}$$

where $r_t(\theta) = \frac{\pi_{\theta_{new}}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio between the new policy $\pi_{\theta_{new}}$ and the old policy $\pi_{\theta_{old}}$ for taking action a_t in state s_t. The hyperparameter ϵ controls the clipping range to ensure that updates remain constrained. Specifically, the function $\text{clip}(x, 1 - \epsilon, 1 + \epsilon)$ constrains x within the interval $[1 - \epsilon, 1 + \epsilon]$. This mechanism limits the magnitude of policy updates, enhancing stability and reducing the risk of divergence during training.

The advantage estimate $\hat{A}_t$ is computed using *generalized advantage estimation* (GAE), which balances bias and variance in the estimation process:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{4.40}$$

where $\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$ represents the temporal difference error at timestep t, γ is the discount factor, λ is the GAE parameter, and T is the episode horizon. By incorporating cumulative future advantages, GAE enables more nuanced policy updates, optimizing decisions in a stable and computationally efficient manner. The combination of the clipped objective function and GAE forms the basis of PPO, making it well-suited for addressing complex inventory problems. For a more detailed discussion of the PPO algorithm and its theoretical foundations, readers are encouraged to refer to [29].

PPO Implementation

To implement the PPO algorithm and learn a well-performing policy, π_{PPO}, we formalize the supply chain environment as a Markov decision process (MDP) in terms of features (observations), actions, and rewards.

As is standard in inventory management representations, the state is defined as a feature vector (the *observation*) serving as input to the neural network [see 4]. Conventionally, features would consist of the inventory vector $\mathbf{x}_t$ and the timestep t, which together form a *sufficient statistic* for the state. However, relying solely on this information places the entire burden on the neural network to learn the product life-cycle phases and interpret the inventory vector accordingly. Preliminary experiments revealed that, in our case study, this approach results in poor performance, as the neural network struggles to manage these complexities effectively.

To overcome this constraint, we propose enriching the feature vector by including a *forecast of upcoming demands*, similar to those used in the OUT and PIL policies. Additionally, rather than directly using the inventory vector $\mathbf{x}_t$ at timestep t, we adopt projection-based ideas inspired by the literature [6, 21] to estimate the future inventory vector at timestep $t + L$, just before the order placed at timestep t arrives.

As a result, the *feature vector* at timestep t is defined as:

$$\left(\mathbb{E}[\mathbf{x}_{t+L}], d_{[t:t+L]}, t\right), \tag{4.41}$$

where $\mathbb{E}[\mathbf{x}_{t+L}] = \left(\mathbb{E}[x_{t+L,1}], \mathbb{E}[x_{t+L,2}], \dots, \mathbb{E}[x_{t+L,m}]\right)$ represents the *expected inventory levels* at timestep $t+L$ for each age category $i \in \{1, 2, \dots, m\}$. These expected values are computed under the assumption that upcoming demands equal their forecasts, enabling a deterministic estimation. Additionally, $d_{[t:t+L]} = (d_{t,t+1}, d_{t,t+2}, \dots, d_{t,t+L})$ represents the demand forecasts from timestep t to $t + L$, and t is the current timestep. Preliminary experiments demonstrated that this new feature representation outperforms standard representations relying solely on current inventory levels [see, e.g., 27, 15, 11, 22, 33, 34].

The *action* a_t at timestep t determines the order quantity:

$$q_t^{\pi_{PPO}} = a_t Q = n_t^{\pi_{PPO}} Q, \tag{4.42}$$

where a_t represents the number of batches ordered, chosen from the discrete set $\{0, 1, 2, \dots, 6\}$. Here, an upper limit of 6 reflects the maximum number of batches that can be ordered in the BMS case study.

Finally, the *reward* $r_t \in \mathbb{R}$ at timestep t is calculated based on the following cost components:

$$r_t = -\left(K(q_t) + h(Y_t - D_t)^+ + b(D_t - Y_t)^+ + wO_t\right), \tag{4.43}$$

where $K(q_t)$ represents the batch ordering cost, h is the holding cost, b is the lost sales cost, and w is the expiration cost, as detailed in Section 4.3. Note that we define the reward with a negative sign since the objective is to minimize the total cost.

4.5 Numerical Experiments

In this section, we present *numerical experiments* designed to evaluate the performance of the implemented policies under various demand scenarios based on synthetic data

provided by BMS. All experiments were conducted on a machine equipped with an 11th Gen Intel(R) Core(TM) i7-11800H CPU at 2.30 GHz and 32 GB of RAM. The code was developed using Python 3.10, leveraging the OpenAI Gym library [5] to define the supply chain environment and the Stable Baselines 3 library [19] to implement the PPO algorithm. The code is publicly available as an open-source library on GitHub [1].

In detail, we consider two demand scenarios for the numerical experiments, both modeled as non-stationary to reflect the *complete lifecycle* of a specific perishable pharmaceutical drug, as discussed with BMS. The lifecycle begins with initial growth as the product gains market acceptance, reaches a peak during its maturity phase, and then declines as alternatives emerge or patents expire. This demand pattern closely mirrors real-world conditions, providing a realistic basis for evaluating the proposed policies in pharmaceutical supply chains.

The *first scenario* is adapted from company-provided data and covers a period of 5 years ($T = 60$ monthly timesteps), incorporating two levels of demand noise. In the *worst-case setting*, demand noise is modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$, representing high fluctuations during the peak phase. In contrast, the *balanced setting* models noise as $\xi_t \sim \mathcal{N}(0, d_t \times 15\%)$, reflecting more moderate fluctuations. These settings were chosen to evaluate the effectiveness of the implemented policies under varying levels of uncertainty. Specifically, the worst-case setting assesses policy performance under high uncertainty, while the balanced setting represents more stable market conditions. In both cases, the forecast error ξ_t follows i.i.d. normal distributions with a mean of zero and a standard deviation σ. This assumption is essential for deriving the lower and upper bounds necessary to optimize the OUT and PIL policies, as discussed in Sections 4.4.1 and 4.4.3.

The *second scenario* is based on real-world data provided by BMS, modeling the complete lifecycle of a specific perishable pharmaceutical drug over a 20-year period ($T = 240$ monthly timesteps), as illustrated in Figure 4.3. This scenario employs the same balanced noise setting described in the first scenario and serves as a reference for more practical and comprehensive analysis.

For each experiment, we compare the performance of the policies by analyzing the *average total cost* over 2000 simulated episodes. By assessing performance across varying levels of demand noise and over both short-term and long-term horizons, this study provides valuable insights into the cost-effectiveness and robustness of the implemented policies in addressing typical pharmaceutical inventory challenges.

4.5.1 First Scenario

In the first scenario, we evaluate the three inventory policies introduced in Section 4.4–OUT, PIL, and PPO–under the worst-case and balanced settings. The *experimental*

[1] https://github.com/frenkowski/SCIMAI-Gym.

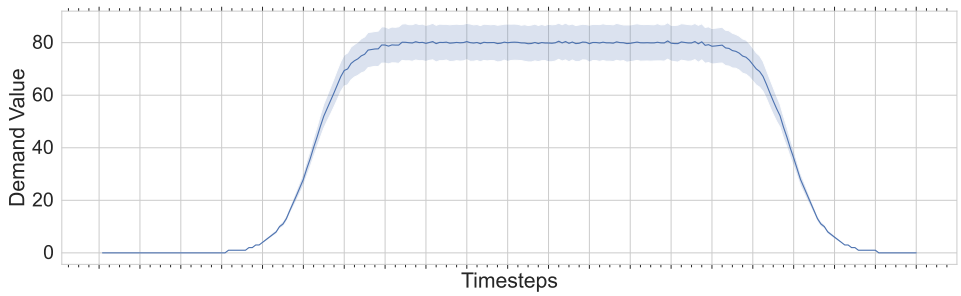


Figure 4.3: A 95% confidence interval based on 2000 simulated episodes for the demand in the second scenario derived from real-world data over an episode horizon of $T = 240$ timesteps.

plan involves varying several parameters of the supply chain environment. Specifically, the expiration cost (w) is set to 2 and 4, the product lifetime (m) ranges from 2 to 4, and the lost sales cost (b) takes on values of 10, 50, 100, and 1000. Other parameters are kept constant to simplify the analysis. The holding cost (h) is fixed at 1, the lead time (L) is set to 2, and both the batch ordering cost ($K(q_t)$) and yield rate ($\hat{z}$) are set to 0. This setup results in 24 unique experiments for each of the two settings (worst-case and balanced). These experiments systematically evaluate the policies under varying cost structures and product characteristics, providing valuable insights into their performance across diverse market conditions.

Worst-Case Setting

The bounds derived for the optimal parameters in the worst-case setting simplify the optimization process by constraining the *search interval* for the OUT and PIL policies. As shown in Table C.1 of Appendix C.3, the optimal values for both the OUT and PIL policies consistently fall within this interval. These bounds are particularly advantageous for the PIL policy, which requires more computational resources than OUT due to the need to calculate expected inventory levels. Although these bounds were derived based on the total cost, they also provide valuable estimates for the optimal parameters of both policies. While formal proof that the optimal values strictly lie within this interval is lacking, the numerical results strongly support this conjecture.

In this first scenario, since there is no upper limit on the number of batches that can be ordered, the upper bound constraining the *action space* of the PPO algorithm was determined by selecting the maximum of the optimal values between the OUT and PIL policies, which minimize Equation 4.11 and Equation 4.38, respectively. This choice is motivated by the observation that optimal values tend to be lower for non-perishable products. Consequently, the non-perishable bounds derived from the OUT and PIL policies were effectively used to constrain the PPO action space.

Turning to the results, as depicted in Figures 4.4 and 4.5, when the product lifetime

m is set to 2, PIL and PPO exhibit similar performance, both slightly outperforming OUT when the lost sales cost b is low. However, as b increases, PPO significantly outperforms OUT while maintaining competitiveness with PIL, with the performance gap widening as b reaches its highest value. As the product lifetime increases to $m = 3$, PIL remains the most effective policy as lost sales costs rise, with PPO performing on par with it while both consistently outperform OUT. Finally, when the product lifetime extends to $m = 4$, the performance gap among the three policies tends to narrow. PIL continues to excel, particularly at the highest value of b. In contrast, PPO's performance slightly declines, even compared to OUT in some cases when the expiration cost w is set to 2. However, OUT demonstrates higher variability in terms of standard deviation, reflecting its sensitivity to cost fluctuations, especially when $b = 1000$.

Note that the OUT policy exhibits significant variability, primarily driven by fluctuations in lost sales costs. Specifically, as lost sales costs increase, even minor changes between episodes can substantially impact the average total cost. In contrast, expiration and holding costs remain relatively stable and smaller in magnitude. The *dominance* of lost sales costs means that differences or outliers disproportionately affect the standard deviation. This variability is likely due to demand uncertainty, which overshadows the stability of other cost components, resulting in significant variability in the average total cost, particularly when lost sales costs reach their highest values.

Balanced Setting

To analyze the performance of the three implemented policies further, we evaluate them under more moderate fluctuations. Figure 4.6 presents the results for this balanced setting. When the product lifetime m is set to 2, the OUT policy significantly underperforms under high lost sales costs and exhibits a high standard deviation, consistent with its behavior in the worst-case setting. In contrast, PPO consistently achieves lower total costs compared to PIL, particularly as lost sales costs increase.

As the product lifetime increases to $m = 3$, PPO continues to achieve the lowest total cost. The OUT and PIL policies perform similarly, with OUT occasionally outperforming PIL, particularly when $b = 1000$, although these differences remain minimal.

Finally, when the product lifetime is extended to $m = 4$, and the expiration cost w is set to 2, all three policies yield nearly identical total costs. However, when the expiration cost increases to $w = 4$, and the lost sales cost b is set to 1000, PPO performs slightly worse than the other policies, while PIL and OUT maintain consistent performance levels.

Interestingly, the OUT policy shows a slight improvement in this balanced setting, emerging as the most efficient policy in specific experiments, even when lost sales costs are high. This improvement is particularly evident in experiments with longer product lifetimes ($m = 3$ or $m = 4$). However, it continues to underperform in experiments with the shortest product lifetime ($m = 2$) as lost sales costs reach $b = 1000$. The PPO algorithm performs efficiently in nearly all experiments, although it exhibits a

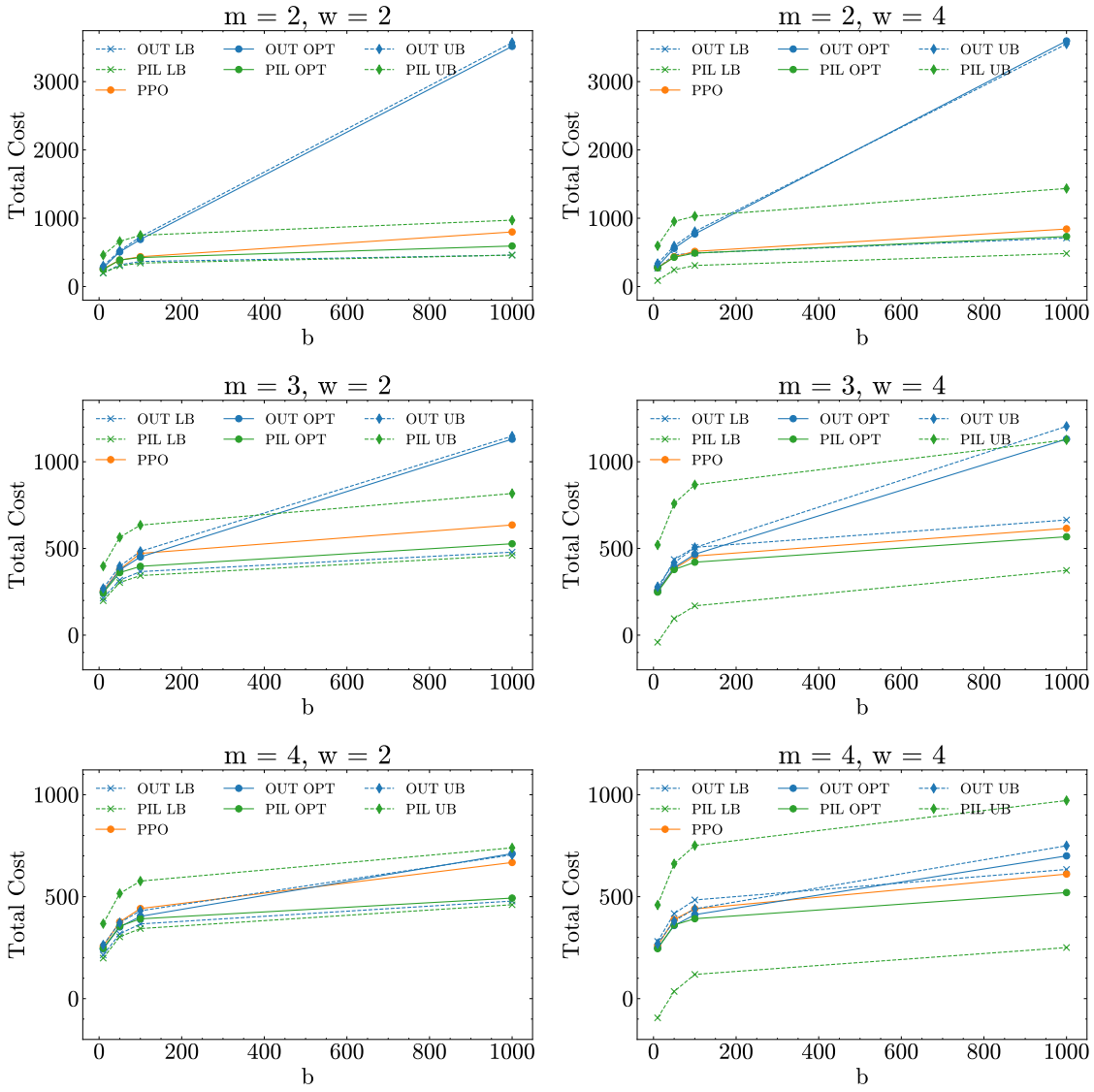


Figure 4.4: Average total cost for the PPO algorithm, with lower (LB), upper (UB), and optimal (OPT) values for the OUT and PIL policies. Demand noise is modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. Each subplot shows the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$.

slight decline in performance at the highest expiration and lost sales costs with the longest product lifetimes. This behavior is consistent with observations in the worst-case setting. The slight decline, while expected, may also be partially attributed to challenges in hyperparameter tuning.

The main *insight* from both the worst-case and balanced settings is that no single

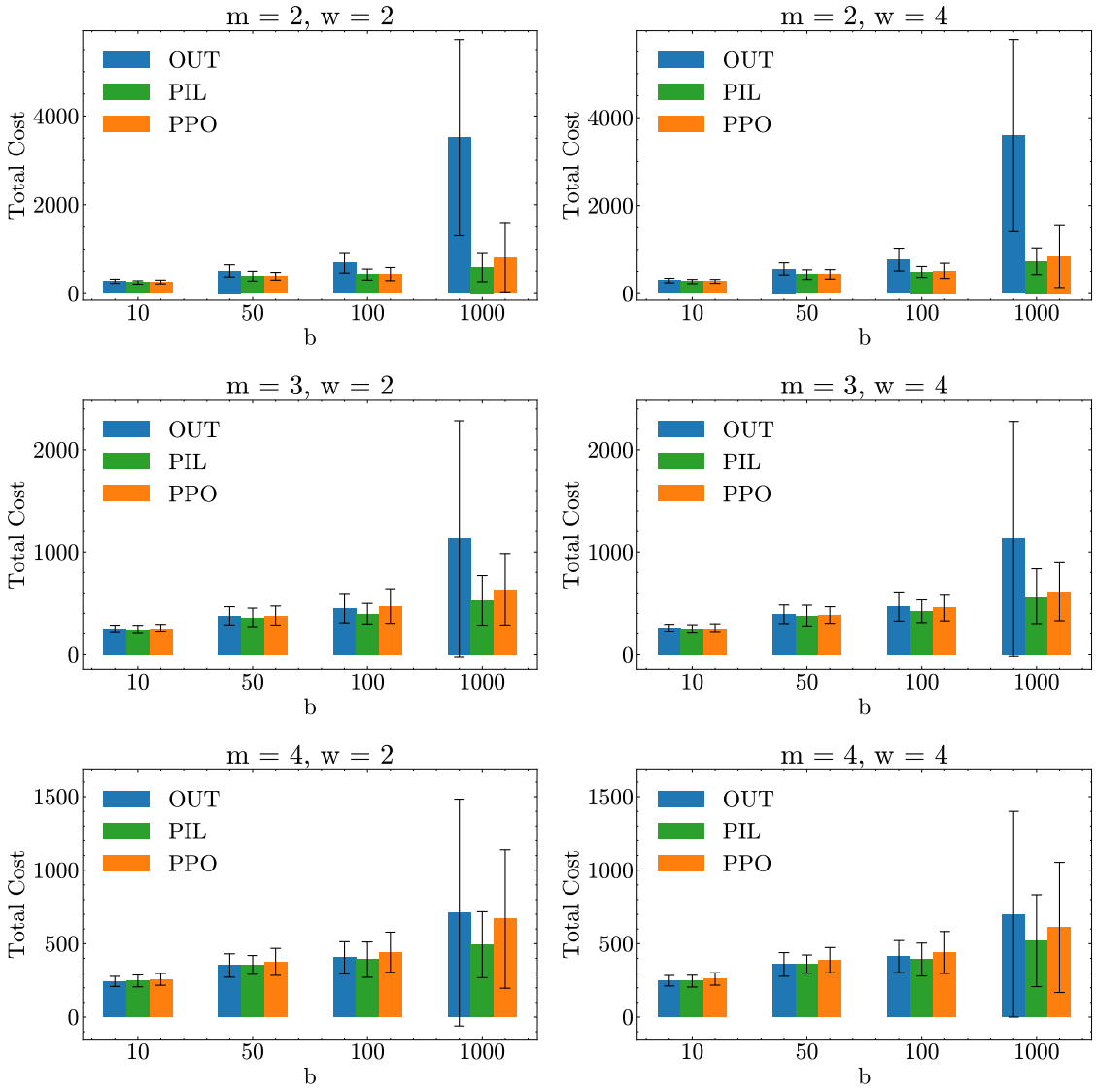


Figure 4.5: Bar plots representing the average total cost over 2000 simulated episodes, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. In each subplot, the bars show the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$.

policy consistently dominates across all experiments. Instead, each policy has experiments where it outperforms the others. Furthermore, the asymptotic optimality of the PIL policy, observed for non-perishable products, does not necessarily extend to perishable products.

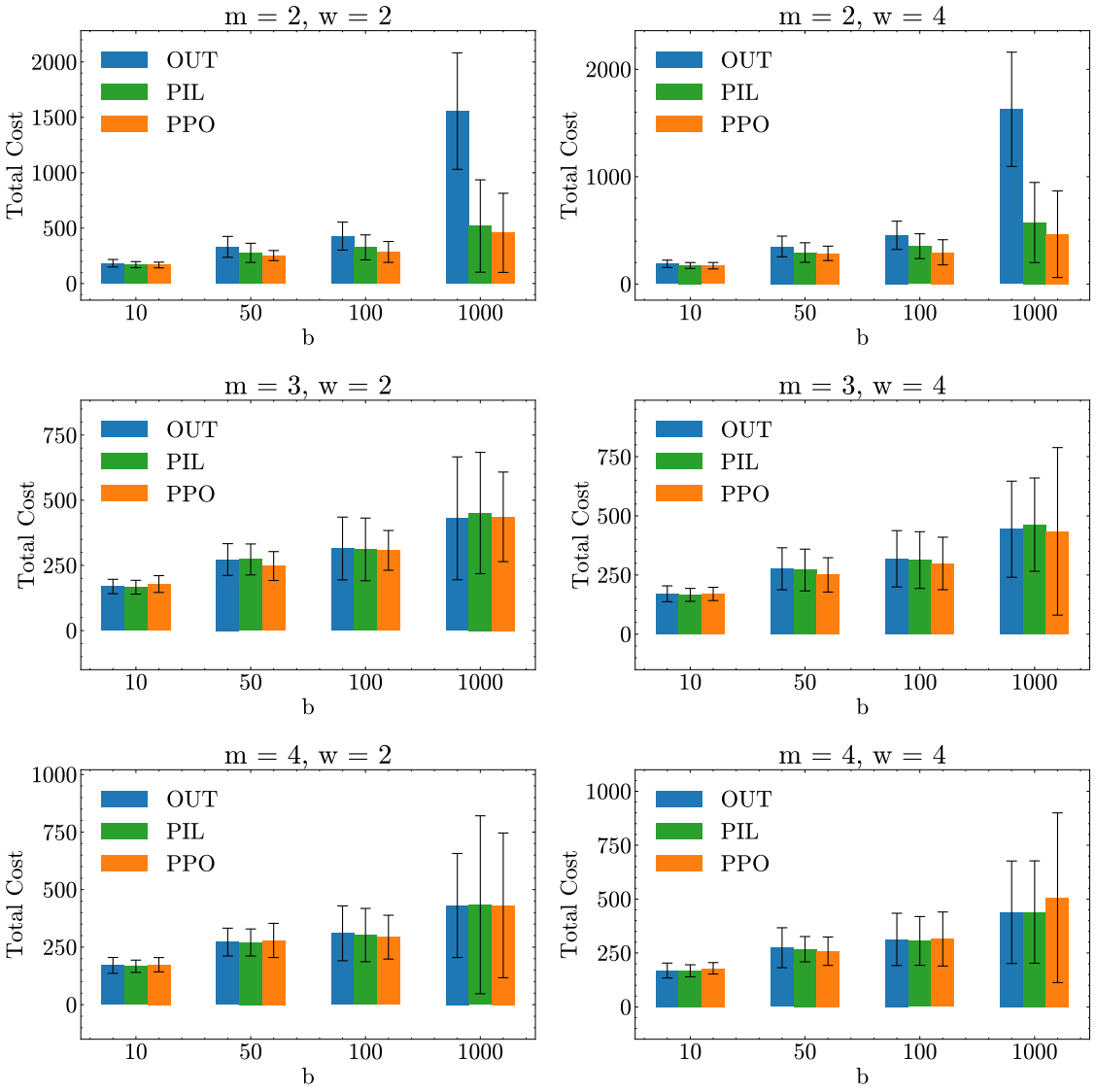


Figure 4.6: Bar plots representing the average total cost over 2000 simulated episodes, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, d_t \times 15\%)$. Each row corresponds to a different value of $m = \{2, 3, 4\}$, and each column to a different value of $w = \{2, 4\}$. In each subplot, the bars show the OUT, PIL, and PPO costs for $b = \{10, 50, 100, 1000\}$.

4.5.2 Second Scenario

In the second scenario, we evaluate the performance of the OUT, PIL, and PPO policies against the BMS baseline derived from *human expertise*. This human-driven policy is based on decisions made by a team of expert human planners at BMS with extensive knowledge of managing perishable pharmaceutical drugs, as discussed in Section 4.4.2. These decisions rely on the same information provided to the PPO algorithm through

Equation 4.41, ensuring a fair comparison.

A key distinction between DRL-based approaches, such as PPO, and the human-driven policy lies in adaptability. While human planners' decisions are built on expertise, they remain *static and are provided a priori*. Consequently, the human-driven policy cannot adjust dynamically in response to the actual state of the supply chain environment or react to unplanned demand fluctuations. In fact, these static decisions remain fixed throughout the simulated episodes, limiting their ability to manage the variability and uncertainty typically encountered in real-world conditions. In contrast, PPO offers a more flexible and responsive solution, *dynamically adapting* at each timestep based on the current state of the environment. Despite its simplicity, the human-driven policy serves as a valuable benchmark, highlighting its limitations and providing a reference point for evaluating advanced, data-driven policies. This comparison underscores the potential for significant improvements in efficiency and responsiveness, which are particularly relevant to pharmaceutical companies.

Table 4.2 summarizes the performance of the four implemented policies (OUT, PIL, PPO, and the BMS baseline) under various yield rates ($\hat{z} = 0\%, 5\%, 10\%$) and lost sales costs ($b = 100, 1000, 10000$) across nine different experiments. All experiments assume a lead time of $L = 12$ timesteps, a product lifetime of $m = 12$ timesteps, and an expiration cost of $\hat{w} = 3.0$. It is important to note that the human-driven policy was explicitly designed for a specific experiment (with $\hat{z} = 10\%$ and $b = 100$) and then applied without modification to the other experiments.

Table 4.2: Average total cost for OUT, PIL, PPO, and the human-driven policy over 2000 simulated episodes in the second scenario derived from real-world data, for values of $\hat{z} = \{0\%, 5\%, 10\%\}$ and $b = \{100, 1000, 10000\}$. Lower cost values indicate better performance.

$\hat{z}$	b	OUT	PIL	PPO	Human
0%	100	19392 ± 3438	21457 ± 3594	17594 ± 2981	120538 ± 758
	1000	28182 ± 10912	31824 ± 10403	25125 ± 9527	120538 ± 758
	10000	35580 ± 23507	40011 ± 32987	33829 ± 22952	120538 ± 758
5%	100	20853 ± 3228	23752 ± 3909	18861 ± 2796	100968 ± 1102
	1000	29796 ± 5478	34195 ± 11336	28069 ± 9410	100968 ± 1102
	10000	35869 ± 21999	41497 ± 22702	36070 ± 25478	100968 ± 1102
10%	100	22588 ± 3349	26349 ± 3852	18608 ± 2838	77477 ± 1170
	1000	30847 ± 9606	36755 ± 10183	29319 ± 9450	80613 ± 1890
	10000	37415 ± 10711	44803 ± 27323	37061 ± 33831	113766 ± 20326

When the yield rate is 0%, all implemented policies outperform the human-driven policy. Despite the lost sales cost, PPO consistently achieves the lowest costs and standard deviations across all experiments. Interestingly, the OUT policy incurs lower

costs than PIL, with the performance gap widening as b increases.

At a 5% yield rate, the costs of the human-driven policy decrease due to the higher production yield, which reduces accumulated stock compared to the $\hat{z} = 0\%$ case. Here, PPO continues to outperform PIL across all experiments, while OUT achieves the lowest costs in the most challenging experiment, where $b = 10000$.

When the yield rate extends to 10%, the performance gap between the human-driven policy and the others narrows. PPO remains the most cost-effective policy across all experiments. However, OUT performs comparably to PPO when $b = 10000$ and exhibits a significantly lower standard deviation than the other policies.

As previously discussed, the high total costs associated with the BMS baseline stem from its static nature, which prevents dynamic adaptation to changes in the supply chain environment. In fact, human planners rely on predetermined actions at each timestep, which remain unchanged throughout the simulated episodes. An analysis of the experiment explicitly designed for the human-driven policy reveals a *tendency to accumulate excess on-hand inventory*. On one hand, this approach almost completely avoids lost sales, particularly in experiments with reduced production yields. On the other hand, it results in significantly higher holding costs, explaining the observed cost gap. While the other policies achieve lower average total costs, they tend to exhibit higher standard deviations, primarily due to experiencing lost sales during the simulated episodes. In contrast, the human-driven policy nearly satisfies all demands.

According to BMS, the average total cost may not be the sole *evaluation criterion* for human planners. Instead, they aim to balance the risk of lost sales with product availability, trying to minimize overstocking and associated product expiration while adhering to ethical and legal constraints regarding potential stockouts.

Contrary to the results reported in the first scenario, PIL does not consistently outperform OUT and PPO across all experiments, with the performance gap widening as production yield increases. As demonstrated in [8], PIL achieves *optimality* when the product lifetime is restricted to a single period. In other experiments, PIL tends to underperform compared to OUT, as observed in this second scenario, where both the lead time and product lifetime are set to 12. Indeed, PIL does not always converge to an optimal solution, even as lost sales costs increase.

To investigate these findings further, we conduct a more detailed analysis of the real-world case study using varying key parameters identified in the first scenario. Specifically, we vary product lifetimes from the set $m = \{3, 6, 12, 24\}$ and lost sales costs from the set $b = \{10, 50, 100, 1000, 10000\}$ for each considered expiration cost $w = \{3, 6\}$, and lead times $L = \{3, 6, 12\}$, while maintaining the production yield $\hat{z}$ constant at zero. The results of these experiments provide deeper insights into the strengths and limitations of the implemented policies when applied across a more diversified and extended set of experimental conditions.

As shown in Table 4.3, PPO significantly outperforms OUT for shorter product lifetimes (i.e., $m = 3$ or $m = 6$), particularly when lost sales costs reach their highest value ($b = 10000$). In these experiments, PPO achieves a performance gap of approximately

Table 4.3: Percentage gap in performance between OUT vs. PPO and PIL vs. PPO for various values of $w = \{3, 6\}$, $m = \{3, 6, 12, 24\}$, and $b = \{10, 50, 100, 1000, 10000\}$, at lead times $L = \{3, 6, 12\}$. A positive percentage value indicates better performance by PPO.

			OUT vs. PPO					PIL vs. PPO				
L	w	m	b					b				
			10	50	100	1000	10000	10	50	100	1000	10000
3	3	3	12	28	30	57	315	14	45	52	52	32
		6	10	25	26	29	159	11	35	36	30	7
		12	7	24	23	20	-8	8	29	26	26	-3
		24	7	23	23	8	8	7	24	25	10	10
	6	3	15	33	42	67	279	18	58	74	76	43
		6	13	33	32	33	101	14	43	45	37	30
		12	8	31	29	29	4	10	36	33	35	8
		24	7	27	24	18	-9	7	28	25	20	-7
6	3	3	9	17	20	55	204	14	45	51	47	24
		6	5	17	23	53	46	6	22	30	39	14
		12	13	29	28	60	571	21	60	72	72	64
		24	5	21	26	37	-13	6	27	36	27	-30
	6	3	9	13	15	30	44	9	28	31	29	20
		6	7	21	20	9	-1	7	23	23	11	3
		12	11	22	24	47	53	12	41	47	52	39
		24	6	21	16	17	0	6	23	17	22	2
12	3	3	8	10	16	67	566	7	34	46	51	28
		6	4	9	7	53	113	-1	22	24	25	2
		12	6	10	10	12	5	1	19	22	27	18
		24	3	10	14	23	10	-4	10	15	14	-3
	6	3	11	14	20	89	499	15	46	62	69	44
		6	8	10	11	29	154	4	27	33	40	19
		12	0	6	10	-1	-2	-4	15	24	16	10
		24	3	7	17	8	21	-4	8	18	-1	8

500% when the lead time L is set to 12. As product lifetimes increase, OUT's performance improves accordingly, independent of expiration costs, and it slightly surpasses PPO in some experiments where $b = 1000$ or $b = 10000$.

PIL performs well when lost sales costs are at their lowest value ($b = 10$) and product lifetimes are longer (i.e., $m \geq 6$), especially when lead times are at their maximum

value ($L = 12$). PIL also occasionally outperforms PPO when $m = 12$ or $m = 24$ with $b \geq 1000$, although no clear trend emerges. However, as lost sales costs increase to a range between 50 and 100, PIL's performance tends to decline, consistently favoring PPO.

A critical factor influencing PIL's performance is the term $\mathbb{E}\left[\sum_{j=t}^{t+L-1} O_j - l_j\right]$, as defined in Equation 4.19. This term represents the difference between the expected expired quantity and lost sales over the lead time. For non-perishable products, incorporating only the expected expired quantity when computing q_t allows PIL to outperform OUT [6]. However, when PIL subtracts the expected lost sales term, this *non-zero mean term* can fluctuate between positive and negative values across timesteps, particularly under non-stationary demand. Such variability makes it challenging for PIL to determine optimal order quantities, leading to higher total costs than OUT, especially when lost sales costs are low (i.e., b equals 10, 50, or 100). In contrast, PIL outperforms OUT by more effectively accounting for expired and lost sales terms for short product lifetimes and high lost sales costs, mainly when $m = 3$ and $b = 10000$. In experiments with larger lead times and longer product lifetimes, both PIL and OUT tend to increase order quantities to reduce lost sales. However, PIL incorporates projected inventory levels by more accurately calculating the expected expired quantity and lost sales, often resulting in better decisions and lower total costs for both values of w. PPO demonstrates an even greater ability to adapt to these trade-offs, frequently outperforming both PIL and OUT. Nevertheless, as lead times and product lifetimes extend, the *dimensionality of PPO's state vector* increases accordingly, making optimization more challenging and reducing its performance gap over the other policies.

4.6 Conclusions

In this study, we evaluated the performance of different inventory control policies for a specific perishable pharmaceutical drug under non-stationary demand, considering factors such as random yields, perishability, positive lead times, and lost sales. This analysis was inspired by a real-world case study provided by BMS. By comparing the OUT policy, the PIL policy, and the PPO algorithm with a human-driven policy, we collected *managerial insights* into their cost-effectiveness and robustness across a diverse set of experimental conditions. The theoretical bounds derived for the OUT and PIL policies offered valuable direction into their optimal parameter values, while the MDP formulation of the supply chain environment was essential for effectively implementing the PPO algorithm.

Extensive numerical experiments revealed that, while simple to implement, the OUT policy exhibits limitations in experiments with short product lifetimes and high lost sales costs. However, its performance improves as these values increase, as observed in the second scenario based on real-world data. The PIL policy offers a robust and cost-effective solution across various experiments, demonstrating consistent performance without

divergence under any of the experiments considered. Notably, it proves particularly efficient in the first scenario, adapted from real-world conditions with reduced product lifetime and lead time values. Finally, as observed in the second scenario, the PPO algorithm demonstrated superior performance in more complex and variable experiments, particularly those involving higher lost sales costs and the inclusion of production yields. However, challenges related to hyperparameter tuning, the computational resources required for training, and the time needed for convergence can be significant, especially as the state vector grows with longer product lifetimes and lead times.

Another salient insight from this study is that none of the three policy types consistently outperforms the others across all experimental conditions. While all three implemented policies achieved lower average costs compared to the human-driven policy, they exhibited higher standard deviations. In contrast, the human-driven policy tends to maintain large inventory levels to avoid stockouts, resulting in higher holding costs but significantly lower cost variability. Therefore, decision-makers in pharmaceutical companies should evaluate policy choices based on specific operational contexts and objectives. Focusing solely on average total cost may not be sufficient, as ethical and legal constraints associated with potential stockouts must also be considered.

References

[1] Ehsan Ahmadi et al. "Intelligent inventory management approaches for perishable pharmaceutical products in a healthcare supply chain." In: *Computers & Operations Research* 147 (Nov. 2022), p. 105968. ISSN: 0305-0548. DOI: 10.1016/j.cor.2022.105968. URL: http://dx.doi.org/10.1016/j.cor.2022.105968.

[2] Joachim Arts et al. *Base-stock policies for lost-sales models: aggregation and asymptotics*. English. BETA publicatie : working papers. Technische Universiteit Eindhoven, Dec. 2015.

[3] Peter Berling and Danja R. Sonntag. "Inventory control in production–inventory systems with random yield and rework: The unit-tracking approach." In: *Production and Operations Management* 31.6 (June 2022), pp. 2628–2645. ISSN: 1937-5956. DOI: 10.1111/poms.13706. URL: http://dx.doi.org/10.1111/poms.13706.

[4] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[5] Greg Brockman et al. *OpenAI Gym*. 2016. DOI: 10.48550/ARXIV.1606.01540. URL: https://arxiv.org/abs/1606.01540.

[6] Rob A.C.M. Broekmeulen and Karel H. van Donselaar. "A heuristic to manage perishable inventory with batch ordering, positive lead-times, and time-varying demand." In: *Computers & Operations Research* 36.11 (Nov. 2009), pp. 3013–3018. ISSN: 0305-0548. DOI: 10.1016/j.cor.2009.01.017. URL: http://dx.doi.org/10.1016/j.cor.2009.01.017.

[7] Jinzhi Bu, Xiting Gong, and Xiuli Chao. "Asymptotic Optimality of Base-Stock Policies for Perishable Inventory Systems." In: *Management Science* 69.2 (Feb. 2023), pp. 846–864. ISSN: 1526-5501. DOI: 10.1287/mnsc.2022.4400. URL: http://dx.doi.org/10.1287/mnsc.2022.4400.

[8] Jinzhi Bu, Xiting Gong, and Huanyu Yin. *Managing Perishable Inventory Systems with Positive Lead Times: Inventory Position vs. Projected Inventory Level*. Available at SSRN: http://dx.doi.org/10.2139/ssrn.4638265. 2023. DOI: 10.2139/ssrn.4638265. URL: http://dx.doi.org/10.2139/ssrn.4638265.

[9] Xiuli Chao et al. "Approximation Algorithms for Capacitated Perishable Inventory Systems with Positive Lead Times." In: *Management Science* 64.11 (Nov. 2018), pp. 5038–5061. ISSN: 1526-5501. DOI: 10.1287/mnsc.2017.2886. URL: http://dx.doi.org/10.1287/mnsc.2017.2886.

[10] Xiuli Chao et al. "Approximation Algorithms for Perishable Inventory Systems." In: *Operations Research* 63.3 (June 2015), pp. 585–601. DOI: 10.1287/opre.2015.1386. URL: https://doi.org/10.1287/opre.2015.1386.

[11] Bram J. De Moor, Joren Gijsbrechts, and Robert N. Boute. "Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management." In: *European Journal of Operational Research* 301.2 (Sept. 2022), pp. 535–545. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.10.045. URL: http://dx.doi.org/10.1016/j.ejor.2021.10.045.

[12] Henri Dehaybe, Daniele Catanzaro, and Philippe Chevalier. "Deep Reinforcement Learning for inventory optimization with non-stationary uncertain demand." In: *European Journal of Operational Research* 314.2 (Apr. 2024), pp. 433–445. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.10.007. URL: http://dx.doi.org/10.1016/j.ejor.2023.10.007.

[13] Tjum van Dijck et al. "Inventory Planning in Capacitated High-Tech Assembly Systems Under Non-Stationary Demand." In: *SSRN Electronic Journal* (2024). ISSN: 1556-5068. DOI: 10.2139/ssrn.4843271. URL: http://dx.doi.org/10.2139/ssrn.4843271.

[14] Jingying Ding and Zhenkang Peng. "Heuristics for perishable inventory systems under mixture issuance policies." In: *Omega* 126 (July 2024), p. 103078. ISSN: 0305-0483. DOI: 10.1016/j.omega.2024.103078. URL: http://dx.doi.org/10.1016/j.omega.2024.103078.

[15] Joren Gijsbrechts et al. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems." In: *Manufacturing & Service Operations Management* 24.3 (May 2022), pp. 1349–1368. DOI: 10.1287/msom.2021.1064. URL: https://doi.org/10.1287/msom.2021.1064.

[16] David A. Goldberg, Martin I. Reiman, and Qiong Wang. "A Survey of Recent Progress in the Asymptotic Analysis of Inventory Systems." In: *Production and Operations Management* 30.6 (June 2021), pp. 1718–1750. ISSN: 1937-5956. DOI: 10.1111/poms.13339. URL: http://dx.doi.org/10.1111/poms.13339.

[17] Carlos Gorria, Mikel Lezaun, and F. Javier López. "Performance measures of nonstationary inventory models for perishable products under the EWA policy." In: *European Journal of Operational Research* 303.3 (Dec. 2022), pp. 1137–1150. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2022.03.018. URL: http://dx.doi.org/10.1016/j.ejor.2022.03.018.

[18] René Haijema and Stefan Minner. "Improved ordering of perishables: The value of stock-age information." In: *International Journal of Production Economics* 209 (Mar. 2019), pp. 316–324. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2018.03.008. URL: http://dx.doi.org/10.1016/j.ijpe.2018.03.008.

[19] Ashley Hill et al. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[20] Woonghee Tim Huh et al. "Asymptotic Optimality of Order-Up-To Policies in Lost Sales Inventory Systems." In: *Management Science* 55.3 (Mar. 2009), pp. 404–420. DOI: 10.1287/mnsc.1080.0945. URL: https://doi.org/10.1287/mnsc.1080.0945.

[21] Willem van Jaarsveld and Joachim Arts. "Projected Inventory-Level Policies for Lost Sales Inventory Systems: Asymptotic Optimality in Two Regimes." In: *Operations Research* (Apr. 2024). ISSN: 1526-5463. DOI: 10.1287/opre.2021.0032. URL: http://dx.doi.org/10.1287/opre.2021.0032.

[22] Illya Kaynov et al. "Deep Reinforcement Learning for One-Warehouse Multi-Retailer inventory management." In: *International Journal of Production Economics* 267 (Jan. 2024), p. 109088. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109088. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109088.

[23] Retsef Levi, Ganesh Janakiraman, and Mahesh Nagarajan. "A 2-Approximation Algorithm for Stochastic Inventory Control Models with Lost Sales." In: *Mathematics of Operations Research* 33.2 (May 2008), pp. 351–374. ISSN: 1526-5471. DOI: 10.1287/moor.1070.0285. URL: http://dx.doi.org/10.1287/moor.1070.0285.

[24] Steven Nahmias. "Optimal Ordering Policies for Perishable Inventory—II." In: *Operations Research* 23.4 (Aug. 1975), pp. 735–749. ISSN: 1526-5463. DOI: 10.1287/opre.23.4.735. URL: http://dx.doi.org/10.1287/opre.23.4.735.

[25] Steven Nahmias. "The Fixed-Charge Perishable Inventory Problem." In: *Operations Research* 26.3 (June 1978), pp. 464–481. ISSN: 1526-5463. DOI: 10.1287/opre.26.3.464. URL: http://dx.doi.org/10.1287/opre.26.3.464.

[26] Purushottaman Nandakumar and Thomas E. Morton. "Near Myopic Heuristics for the Fixed-Life Perishability Problem." In: *Management Science* 39.12 (Dec. 1993), pp. 1490–1498. ISSN: 1526-5501. DOI: 10.1287/mnsc.39.12.1490. URL: http://dx.doi.org/10.1287/mnsc.39.12.1490.

[27] Afshin Oroojlooyjadid et al. "A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization." In: *Manufacturing & Service Operations Management* 24.1 (Jan. 2022), pp. 285–304. DOI: 10.1287/msom.2020.0939. URL: https://doi.org/10.1287/msom.2020.0939.

[28] Dennis Prak, Ruud Teunter, and Aris Syntetos. "On the calculation of safety stocks when demand is forecasted." In: *European Journal of Operational Research* 256.2 (Jan. 2017), pp. 454–461. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2016.06.035. URL: http://dx.doi.org/10.1016/j.ejor.2016.06.035.

[29] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. DOI: 10.48550/ARXIV.1707.06347. URL: https://arxiv.org/abs/1707.06347.

[30] Jing-Sheng J Song. *Research handbook on inventory management*. Edward Elgar Publishing, 2023.

[31] Danja Sonntag and Gudrun P. Kiesmüller. "Disposal versus rework – Inventory control in a production system with random yield." In: *European Journal of Operational Research* 267.1 (May 2018), pp. 138–149. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2017.11.019. URL: http://dx.doi.org/10.1016/j.ejor.2017.11.019.

[32] Danja Sonntag and Gudrun P. Kiesmüller. "The shape of the yield and its impact on inventory decisions." In: *4OR* 14.4 (Apr. 2016), pp. 405–415. ISSN: 1614-2411. DOI: 10.1007/s10288-016-0317-z. URL: http://dx.doi.org/10.1007/s10288-016-0317-z.

[33] Francesco Stranieri, Edoardo Fadda, and Fabio Stella. "Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem." In: *International Journal of Production Economics* 268 (Feb. 2024), p. 109099. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109099. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109099.

[34] Francesco Stranieri, Edoardo Fadda, and Fabio Stella. "Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem." In: *International Journal of Production Economics* 268 (Feb. 2024), p. 109099. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109099. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109099.

[35] Ou Tang, Robert W. Grubbström, and Simone Zanoni. "Planned lead time determination in a make-to-order remanufacturing system." In: *International Journal of Production Economics* 108.1–2 (July 2007), pp. 426–435. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2006.12.034. URL: http://dx.doi.org/10.1016/j.ijpe.2006.12.034.

[36] Tarkan Temizöz et al. *Deep Controlled Learning for Inventory Control*. 2020. DOI: 10.48550/ARXIV.2011.15122. URL: https://arxiv.org/abs/2011.15122.

[37] Joost Floris Van der Haar et al. *Industrializing Deep Reinforcement Learning for Operational Spare Parts Inventory Management*. Available at SSRN: http://dx.doi.org/10.2139/ssrn.4999374. 2024. DOI: 10.2139/ssrn.4999374. URL: http://dx.doi.org/10.2139/ssrn.4999374.

[38] Lotte Van Hezewijk, Nico Dellaert, and Willem Van Jaarsveld. *Scalable Deep Reinforcement Learning in the Non-Stationary Capacitated Lot Sizing Problem*. Available at SSRN: http://dx.doi.org/10.2139/ssrn.4846298. 2024. DOI: 10.2139/ssrn.4846298. URL: http://dx.doi.org/10.2139/ssrn.4846298.

[39] Gideon JJ Van Zyl. *Inventory control for perishable commodities*. Tech. rep. North Carolina State University. Dept. of Statistics, 1963.

[40] Nathalie Vanvuchelen, Joren Gijsbrechts, and Robert Boute. "Use of Proximal Policy Optimization for the Joint Replenishment Problem." In: *Computers in Industry* 119 (Aug. 2020), p. 103239. ISSN: 0166-3615. DOI: 10.1016/j.compind.2020.103239. URL: http://dx.doi.org/10.1016/j.compind.2020.103239.

[41] Huanan Zhang, Cong Shi, and Xiuli Chao. "Technical Note—Approximation Algorithms for Perishable Inventory Systems with Setup Costs." In: *Operations Research* 64.2 (Apr. 2016), pp. 432–440. ISSN: 1526-5463. DOI: 10.1287/opre.2016.1485. URL: http://dx.doi.org/10.1287/opre.2016.1485.

Chapter 5

Conclusions

This thesis thoroughly explores the applicability and transformative power of deep reinforcement learning (DRL) in the field of supply chain inventory management (SCIM), aiming to enhance the efficiency and robustness of modern supply chains. Through *three interconnected research studies*, it investigates how DRL addresses scalability and computational challenges; outperforms traditional inventory policies in multi-echelon systems under dynamic and uncertain conditions; combines with traditional optimization methods to improve production and logistics decisions; and demonstrates its effectiveness and interpretability in real-world contexts, such as pharmaceutical supply chains. Collectively, these findings underscore the significant contributions of DRL to advancing SCIM and establishing a solid foundation for future research and innovation in this domain.

Performance of Deep Reinforcement Learning Algorithms in Two-Echelon Inventory Control Systems [16]

The first study considers a sequential decision-making problem in a capacitated two-echelon system consisting of a factory at the first echelon and multiple local warehouses at the second echelon. The local warehouses face stochastic, seasonal demand for diverse product types, with orders delivered after a constant, positive lead time. Excess demand that cannot be immediately satisfied from available stock is back-ordered. The primary goal is to determine optimal production and shipment quantities over a given time horizon. This study highlights the inherent complexity of finding an optimal solution due to the curse of dimensionality arising from the interplay among the number of product types and local warehouses, the length of lead times, and the stochastic nature of the problem.

Focusing on the first research question:

> *How can DRL improve performance in multi-echelon systems under seasonal and stochastic demand scenarios, varying the number of local warehouses, product types, and the length of lead times?*

By mathematically formulating the problem as a Markov decision process (MDP) and focusing on the action vector, this study aims to improve scalability and reduce computational complexity by exploiting a continuous action space with independently determined bounds for each action value. For instance, the lower bound is zero for local warehouses, while the upper bound corresponds to their maximum capacity. This approach scales linearly with the size of the problem, enabling DRL to adapt to diverse experimental conditions–including varying numbers of product types and local warehouses–while maintaining robust performance as lead times increase. According to the results, DRL algorithms achieve convergence across all experiments, demonstrating their capacity to manage these complexities effectively under stochastic demand scenarios.

Regarding the second research question:

> *Can we design an allocation rule for selecting feasible actions and preventing backorders in the first echelon?*

In response, the study proposes a balanced allocation rule designed for continuous action spaces. Since each action is independent of the others, it is inherently challenging to enforce interdependency constraints. This rule ensures the central warehouse does not ship more products than available, thereby maintaining non-negative inventory levels and preventing backorders at the first echelon. It also provides fair and balanced product distribution among local warehouses, preventing allocation imbalances where some warehouses receive minimal or no products. By constraining the continuous action space to consider coherent and feasible actions, this allocation rule enhances the practical applicability of DRL, improves training times, and facilitates algorithmic convergence.

In relation to the third research question:

> *How does the performance of DRL compare to traditional inventory policies in terms of cost minimization and training time?*

The study uses a data-driven method via Bayesian optimization to determine optimal parameter values for static inventory policies, applying independent bounds for each parameter similarly to the approach used with DRL. DRL algorithms, particularly the proximal policy optimization (PPO) algorithm, consistently outperform traditional inventory policies in terms of cost minimization while achieving comparable training times. Although the (s, Q)-policy offers commendable computational efficiency and a balanced trade-off between simplicity and performance, it does not match the cost minimization achieved by PPO. Additionally, the deterministic linear programming policy, which replaces stochastic demand with its average value, performs significantly worse than DRL algorithms. These findings underscore the importance of addressing the stochastic nature of the problem, which static and deterministic models fail to capture effectively.

In summary, the PPO algorithm emerges as the best-performing algorithm across all experiments. Although the (s, Q)-policy provides a viable alternative by balancing simplicity and computational efficiency, it does not reach the performance level of PPO. This study advances innovations in exploiting a continuous action space and a balanced allocation rule, offering a robust framework for addressing the scalability and dimensionality challenges inherent in modern supply chains. Significantly, all implemented state-of-the-art DRL algorithms achieve convergence with training times comparable to static inventory policies. These results validate the practicality of DRL for large-scale, complex multi-echelon systems, encouraging decision-makers to consider these insights when selecting an inventory policy.

Combining Deep Reinforcement Learning and Multi-Stage Stochastic Programming to Address the Supply Chain Inventory Management Problem [15]

By combining DRL with multi-stage (MS) stochastic programming, the second study introduces a novel heuristic to optimize production and logistics decisions in multi-echelon systems, tackling the computational and scalability challenges inherent in complex sequential decision-making problems. Similar to the supply chain analyzed in the first study, this work focuses on a two-echelon system. However, it involves a single item type, no lead times, production limits, and fixed and variable logistics costs, defined as costs per vehicle and unit shipped, respectively. The MDP formulation builds upon the environment established in the first study, employing a continuous action space with independent bounds and a balanced allocation rule. As demonstrated previously, the proposed heuristic relies on the PPO algorithm due to its superior performance compared to other state-of-the-art DRL algorithms.

In alignment with the first research question:

> *How can combining DRL with MS stochastic programming optimize production and logistics decisions in two-echelon systems?*

This approach decomposes the problem into two components. DRL determines the number of batches to produce at each decision period, while MS stochastic programming calculates the number of batches to ship. By combining the complementary strengths of both methods, this hybrid approach effectively manages the complexity of production and logistics decisions. Specifically, DRL excels in long-horizon production planning through its simulation-optimization capabilities, while MS stochastic programming uses its model-based formulation to handle short-term logistics decisions. In practice, two distinct agents operate sequentially. First, DRL is trained over a sequence of episodes to learn an effective policy, and the resulting actions are used to fix production variables. Given these fixed values, the MS stochastic programming model focuses exclusively on logistics decisions. By decomposing the problem, the heuristic minimizes total costs while avoiding the dimensionality challenges of solving all variables simultaneously.

Addressing the second research question:

How can the hybrid heuristic adapt to changes in supply chain parameters, such as seasonal and stochastic demand patterns, production limits, warehouse capacity constraints, and fixed and variable logistics costs, while mitigating computational complexity?

During the training phase, DRL learns a policy that accounts for seasonal and stochastic demand patterns while respecting production limits. This results in proactive behaviors, such as preserving stock to anticipate future demand and thus preventing myopic decisions. Once training is complete, DRL selects actions instantaneously based on the current state of the environment. Concurrently, MS stochastic programming complements this adaptability through its mathematical formulation, which models key supply chain parameters, including warehouse capacity constraints and fixed and variable logistics costs, integrating their effects directly into the objective function. By using DRL-determined actions and reducing the number of MS stochastic programming stages, the heuristic mitigates computational complexity and avoids the curse of dimensionality typical of MS stochastic programming. While the primary computational effort occurs during the DRL training phase, once completed, the heuristic can rapidly compute decisions, making it a practical, scalable solution for large-scale supply chains.

In answering the third research question:

Can the hybrid heuristic improve performance compared to standalone DRL and MS stochastic programming methods?

Numerical experiments demonstrate the superiority of the hybrid heuristic in terms of scalability and performance compared to standalone methods. In small-scale scenarios, its performance closely approaches that of exact solutions. In larger-scale scenarios, where obtaining optimal solutions is computationally infeasible, the heuristic consistently outperforms benchmarks such as the PPO algorithm, the (s, Q)-policy, and the deterministic linear programming policy. Sensitivity analyses conducted by varying demand parameters further validate its robustness, underscoring its potential as a decision-making tool capable of delivering effective solutions in practical applications.

In summary, the proposed heuristic combines the complementary strengths of DRL and MS stochastic programming to address production and logistics decisions in complex supply chains effectively. By overcoming the limitations of traditional optimization methods, this hybrid approach provides a scalable, computationally efficient, and adaptable solution. Its ability to compute decisions rapidly after the DRL training phase makes it particularly valuable to decision-makers, ensuring high-quality solutions while preserving computational tractability.

Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity [17]

Managing inventory in pharmaceutical supply chains presents multifaceted challenges due to lost sales instead of backorders, the critical nature of perishable products, and

the need to consider significant lead times. This study focuses on a single-echelon supply chain modeled on a real-world case, connecting a third-party factory to a storage warehouse with unlimited capacity. The factory produces a single product type, subject to limits on batches per shipment and potential losses due to production yield variability. The storage warehouse faces non-stationary demand, supported by demand forecasts directly provided by the pharmaceutical company.

In response to the first research question:

> *How should DRL and traditional inventory policies be adapted to provide robust solutions for managing non-stationary demand while considering product perishability, production yields, batch limits, lead times, and lost sales?*

To address this, both traditional inventory policies and DRL algorithms are adapted to account for the complexities of the pharmaceutical supply chain. For traditional inventory policies, the order-up-to (OUT) policy is made non-stationary by linking the target inventory level to demand forecasts. Additionally, a projected inventory level (PIL) policy is developed as a variant of the OUT policy. This variant considers expected inventory levels over lead times, provides a more accurate approximation of expired stock by incorporating lost sales and production yields, and uses the true demand distribution rather than deterministic approximations. Bounds-based procedures for optimizing the parameters of the OUT and PIL policies are derived and validated. For DRL, the environment is formalized as an MDP, with a state representation that includes expected inventory levels for each age category and demand forecasts. This novel state representation empirically outperforms classical representations, enabling PPO to learn effective policies that manage the inherent complexities of the considered supply chain.

Considering the second research question:

> *How does DRL compare to human-driven policies and traditional inventory policies in managing a real-world pharmaceutical product?*

The study evaluates the performance of the adapted OUT and PIL policies and the PPO algorithm by comparing their decisions to those of expert human planners. By varying critical parameters, the results indicate that PPO demonstrates superior performance in more complex and variable scenarios, particularly those characterized by higher lost sales costs and production yields, as observed in the real-world case. However, it does not consistently outperform traditional inventory policies across all experiments. The computational resources required for PPO training and convergence remain significant, especially as the state vector grows with longer product lifetimes and lead times. In contrast, the adapted OUT and PIL policies provide a practical alternative, balancing performance and ease of implementation.

Finally, exploring the third research question:

> *How do human-driven policies compare to data-driven methods, and what managerial insights can be drawn from their performance across varying experimental conditions?*

The analysis reveals that the OUT and PIL policies and the PPO algorithm significantly outperform human-driven policies in terms of cost minimization. However, all data-driven methods exhibit greater performance variability and occasionally fail to prevent lost sales during simulations, reducing their robustness. In contrast, human-driven policies nearly satisfy all demand, increasing holding costs due to overstocking but consistently ensuring product availability. This approach reflects the priorities of human planners, who aim to minimize the risk of lost sales given the ethical and legal constraints associated with potential stockouts. These findings suggest that industries should evaluate inventory policies not only based on average total cost but also by considering operational constraints, strategic priorities, and the broader implications of stockouts.

In summary, while the PPO algorithm demonstrates superior performance in complex, variable scenarios, the challenges associated with its implementation suggest that simpler policies, like the adapted OUT and PIL policies, may provide an acceptable trade-off between performance and ease of implementation. Furthermore, the findings indicate that no single policy is universally optimal, underscoring the importance of aligning policy choice with the specific needs of each operational context. These insights offer practical guidance for decision-makers in pharmaceutical industries, emphasizing the importance of evaluating inventory policies based on cost and their ability to balance robustness with ethical and legal constraints. This consideration is especially critical in the pharmaceutical domain, where product availability has profound implications.

5.1 Collective Findings

In conclusion, the *findings* of this thesis demonstrate the capability of DRL to optimize multidimensional decision-making processes, underscoring its potential to enhance operational efficiency, reduce costs, and improve performance in dynamic and uncertain environments. By carefully designing a continuous action space with a balanced allocation rule and by combining DRL with traditional optimization methods, this research addresses critical scalability and computational challenges, enabling the effective management of increasingly complex multi-echelon systems as the problem size grows. In industries like pharmaceuticals, where perishability and ethical and legal constraints are essential, the methodologies and strategies presented here offer a robust framework for balancing cost efficiency with interpretability considerations embedded in real-world data.

The development of an open-source Python library for simulating and analyzing diverse supply chain contexts–integrating both DRL algorithms and traditional inventory policies–provides researchers and practitioners with a flexible tool for conducting rigorous experimentation and facilitating real-world applications. By *bridging the gap* between theoretical understanding and practical implementation, this tool drives innovation and advancements in SCIM through its capacity to model varying supply chain

parameters and experimental conditions.

Collectively, this thesis successfully answers its specific research questions and makes significant *contributions*, establishing a solid foundation for future research and progress in the field of SCIM. These insights trace a clear path toward the development of modern, robust, and efficient data-driven supply chains that align with the transformative principles of artificial intelligence (AI) and industry 4.0.

5.2 Future Research

Future research could focus on implementing *action masking* within the context of SCIM to address the challenges posed by high-dimensional action spaces in decision-making processes. DRL algorithms explore an action space to identify the optimal action for a given state. However, traditional approaches often fail to adequately prevent infeasible actions in environments where the set of feasible actions varies based on the state. For example, in a factory within a multi-echelon system, the number of possible shipments depends on inventory levels. Existing solutions, such as penalizing infeasible actions with negative rewards or remapping them to feasible alternatives, often compromise solution quality [9, 14]. Action masking provides a promising alternative by dynamically modifying artificial neural network outputs to assign zero probability to infeasible actions, ensuring they are never selected [19]. This technique has demonstrated its effectiveness in accelerating convergence and achieving superior performance in domains like video games [23, 20, 12]. Extending action masking to DRL applications in multi-echelon systems has the potential to significantly enhance training efficiency, making it a promising area for further exploration [5].

DRL algorithms often require extensive exploration to learn effective policies, a process that can be both time-consuming and computationally expensive, particularly in dynamic and uncertain environments. Techniques that rely on learning from scratch may fail to fully exploit SCIM domain knowledge, resulting in slower convergence and suboptimal policies [1]. Another avenue for future research involves exploring the application of *reward shaping* techniques to improve the efficiency and effectiveness of DRL [11, 8]. By incorporating guidance from a teacher policy, reward shaping adjusts the rewards observed by DRL algorithms, directing them toward better decisions early in the training phase [4]. For example, novel reward functions can penalize discrepancies in key metrics, such as expired items and inventory levels, accelerating convergence and improving decision-making. Moreover, dynamic reward-shaping techniques can further refine this process by closely aligning rewards with teacher policies while progressively reducing reliance on them. These advancements highlight the potential of reward shaping as a key area for future research, particularly in addressing the complexities of large-scale supply chains [2, 13].

Another promising direction for future research in SCIM is the integration of large language models (LLMs), such as ChatGPT, into DRL implementations [21, 7, 10]. These

advanced AI models, which are a subset of generative AI, are built on sophisticated deep learning architectures trained on vast amounts of textual data and represent a domain of machine learning focused on creating new content–including text–by learning patterns from existing data [18, 3]. By recognizing and interpreting patterns in language, LLMs can analyze linguistic structures, identify context, and generate contextually relevant responses to support decision-making processes [22]. For example, LLMs could assist in action masking by identifying infeasible actions based on state descriptions and contextual information. Moreover, they can act as teacher policies during the early training phases of DRL algorithms, guiding them toward impactful actions before these algorithms independently develop effective policies. Consequently, LLMs can contribute to shaping reward functions by suggesting informative actions, accelerating the training phase, and reducing the costly trial-and-error process inherent in purely data-driven methods. By assuming this dual role as both action maskers and teacher policies, LLMs represent a promising avenue to improve DRL algorithms, advancing the development of more robust and efficient solutions in SCIM.

Finally, a wider range of structural attributes and operational factors could be incorporated into the existing open-source Python library. For example, it could model an arbitrary number of echelons and warehouses, each defined by its own parameters. Moreover, leveraging advanced computational frameworks like JAX and parallelization techniques, such as Picard iterations for batched computations or segmenting simulations to optimize GPU utilization, could significantly accelerate large-scale simulations [6]. By continuously updating and refining this library to include new methodologies and strategies, it may evolve into a comprehensive tool that facilitates experimentation, drives innovation, and enables researchers and practitioners to address increasingly complex, modern SCIM challenges.

References

[1] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[2] Eleonora Brambatti et al. "Application and Development of Reward Shaping Techniques to Deep Reinforcement Learning for Perishable Inventory Management." MA thesis. University of Milano-Bicocca, 2024.

[3] Pawan Budhwar et al. "Human resource management in the age of generative artificial intelligence: Perspectives and research directions on ChatGPT." In: *Human Resource Management Journal* 33.3 (July 2023), pp. 606–659. ISSN: 1748-8583. DOI: 10.1111/1748-8583.12524. URL: http://dx.doi.org/10.1111/1748-8583.12524.

[4] Bram J. De Moor, Joren Gijsbrechts, and Robert N. Boute. "Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management." In: *European Journal of Operational Research* 301.2 (Sept. 2022), pp. 535–545. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.10.045. URL: http://dx.doi.org/10.1016/j.ejor.2021.10.045.

[5] Giovanni Fantoni, Fabio Stella, and Francesco Stranieri. "Inventory Management: Can Action Masking Help?" Bachelor's thesis. University of Milano-Bicocca, 2023.

[6] Vivek Farias et al. *Speeding up Policy Simulation in Supply Chain RL*. 2024. DOI: 10.48550/ARXIV.2406.01939. URL: https://arxiv.org/abs/2406.01939.

[7] Samuel Fosso Wamba et al. "Are both generative AI and ChatGPT game changers for 21st-Century operations and supply chain excellence?" In: *International Journal of Production Economics* 265 (Nov. 2023), p. 109015. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109015. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109015.

[8] Abhishek Gupta et al. "Unpacking reward shaping: understanding the benefits of reward engineering on sample complexity." In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2024. ISBN: 9781713871088.

[9] Shengyi Huang and Santiago Ontañón. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms." In: *The International FLAIRS Conference Proceedings* 35 (May 2022). ISSN: 2334-0762. DOI: 10.32473/flairs.v35i.130584. URL: http://dx.doi.org/10.32473/flairs.v35i.130584.

[10] Ilya Jackson et al. "Generative artificial intelligence in supply chain and operations management: a capability-based framework for analysis and implementation." In: *International Journal of Production Research* 62.17 (Jan. 2024), pp. 6120–6145. ISSN: 1366-588X. DOI: 10.1080/00207543.2024.2309309. URL: http://dx.doi.org/10.1080/00207543.2024.2309309.

[11] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping." In: *Proceedings of the Sixteenth International Conference on Machine Learning*. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287. ISBN: 1558606122.

[12] OpenAI et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. DOI: 10.48550/ARXIV.1912.06680. URL: https://arxiv.org/abs/1912.06680.

[13] Marta Privitera et al. "Design and Comparison of Potential Functions for Reward Shaping in Deep Reinforcement Learning." MA thesis. University of Milano-Bicocca, 2024.

[14] Roland Stolz et al. *Excluding the Irrelevant: Focusing Reinforcement Learning through Continuous Action Masking*. 2024. DOI: 10.48550/ARXIV.2406.03704. URL: https://arxiv.org/abs/2406.03704.

[15] Francesco Stranieri, Edoardo Fadda, and Fabio Stella. "Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem." In: *International Journal of Production Economics* 268 (Feb. 2024), p. 109099. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2023.109099. URL: http://dx.doi.org/10.1016/j.ijpe.2023.109099.

[16] Francesco Stranieri, Fabio Stella, and Chaaben Kouki. "Performance of deep reinforcement learning algorithms in two-echelon inventory control systems." In: *International Journal of Production Research* 62.17 (Mar. 2024), pp. 6211–6226. ISSN: 1366-588X. DOI: 10.1080/00207543.2024.2311180. URL: http://dx.doi.org/10.1080/00207543.2024.2311180.

[17] Francesco Stranieri et al. *Classical and Deep Reinforcement Learning Inventory Control Policies for Pharmaceutical Supply Chains with Perishability and Non-Stationarity*. 2025. DOI: 10.48550/ARXIV.2501.10895. URL: https://arxiv.org/abs/2501.10895.

[18] Anjana Susarla et al. "The Janus Effect of Generative AI: Charting the Path for Responsible Conduct of Scholarly Activities in Information Systems." In: *Information Systems Research* 34.2 (June 2023), pp. 399–408. ISSN: 1526-5536. DOI: 10.1287/isre.2023.ed.v34.n2. URL: http://dx.doi.org/10.1287/isre.2023.ed.v34.n2.

[19] Nathalie Vanvuchelen, Bram J De Moor, and Robert N Boute. "The use of continuous action representations to scale deep reinforcement learning for inventory control." In: *IMA Journal of Management Mathematics* (Nov. 2024). ISSN: 1471-6798. DOI: 10.1093/imaman/dpae031. URL: http://dx.doi.org/10.1093/imaman/dpae031.

[20] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." In: *Nature* 575.7782 (Oct. 2019), pp. 350–354. DOI: 10.1038/s41586-019-1724-z. URL: https://doi.org/10.1038/s41586-019-1724-z.

[21] Taylor Webb, Keith J. Holyoak, and Hongjing Lu. "Emergent analogical reasoning in large language models." In: *Nature Human Behaviour* 7.9 (July 2023), pp. 1526–1541. ISSN: 2397-3374. DOI: 10.1038/s41562-023-01659-w. URL: http://dx.doi.org/10.1038/s41562-023-01659-w.

[22] Jason Wei et al. *Emergent Abilities of Large Language Models*. 2022. DOI: 10.48550/ARXIV.2206.07682. URL: https://arxiv.org/abs/2206.07682.

[23] Tom Zahavy et al. “Learn what not to learn: action elimination with deep reinforcement learning.” In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 3566–3577.

Appendix A

Supplementary Material of Chapter 2

A.1 Hyperparameters Selection

The process of selecting appropriate values for hyperparameters in ANNs is widely acknowledged to be complex and time-consuming, as extensively discussed in the relevant literature [3, 5]. Moreover, hyperparameters play a crucial role in the context of DRL algorithms since they can significantly influence training, convergence, and, consequently, their relative performance [2].

In our experimental setup, each DRL agent was trained for a specific number of episodes, which varied depending on the number of local warehouses.

For one product type:

- 25000 episodes with 1 and 2 local warehouses.
- 50000 episodes with 3 local warehouses.

For two product types:

- 50000 episodes with 1 and 2 local warehouses.
- 75000 episodes with 3 local warehouses.

Using the Population-Based Bandit (PB2) algorithm [4], the DRL agents determined the best hyperparameter values from those we selected, which are presented in Table A.1. PB2 adapts hyperparameters during training, leading to improved performance and convergence of the agents. In our implementation, DRL agents accessed demand values from the preceding two time steps (i.e., $\tau = 2$). However, preliminary results suggest that comparable performances can be achieved when agents access demand values from the last or the last three time steps.

To identify the optimal static inventory policy parameters – S_j^i for the BS policy and s_j^i and Q_j^i for the sQ policy – we employed a data-driven method using Bayesian optimization via Ax [1], an open-source Python platform designed for optimizing simulation

Table A.1: The selected hyperparameters for the DRL algorithms. For hyperparameters not included in the table, we used the default values provided by Ray.

Hyperparameter	Values		
	PPO	PG	A3C
Learning Rate	{7e-4, 1e-5, 3e-5, 5e-5, 7e-5}	{1e-5, 3e-5, 7e-5, 1e-6}	{1e-5, 3e-5, 5e-5, 7e-5}
Neurons per Hidden Layer	{(32, 32), (64, 64), (128, 128)}	{(32, 32), (64, 64), (128, 128)}	{(32, 32), (64, 64), (128, 128)}
Gamma	0.99	0.99	0.99
Train Batch Size	{2000, 4000, 8000}	{200, 600, 800}	{200, 800, 1400}
Gradient Clipping	-	-	{20, 40, 60}
Stochastic Gradient Descent Iterations	{5, 15, 25}	-	-
Stochastic Gradient Descent Mini-Batch Size	{128, 256, 512}	-	-

experiments. As delineated in Section 2.3.2 of the original manuscript, we opted to make reordering decisions independently, and as for the DRL algorithms, we designated the same lower and upper bound for each parameter.

In our experimental setup, each static inventory policy was optimized for a specific number of training iterations based on the number of local warehouses.

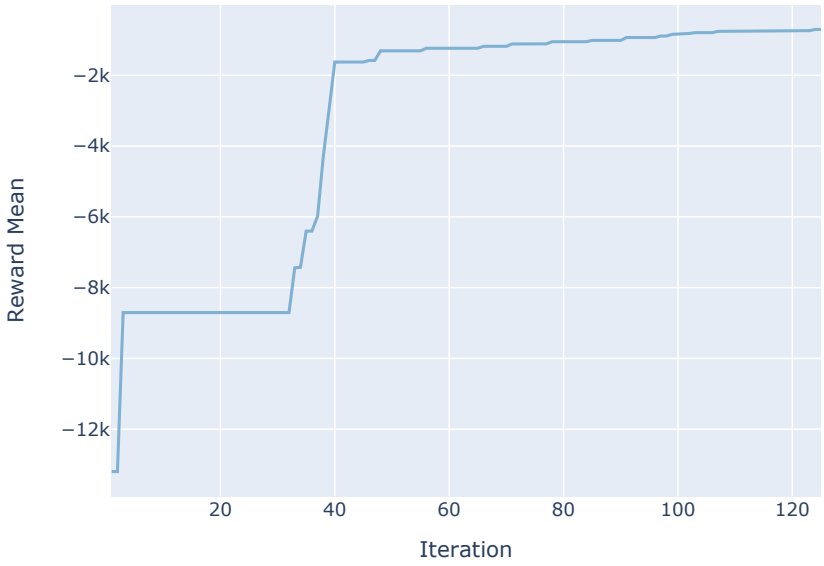
For one product type:

- 50 training iterations with 1 local warehouse.
- 75 training iterations with 2 local warehouses.
- 100 training iterations with 3 local warehouses.

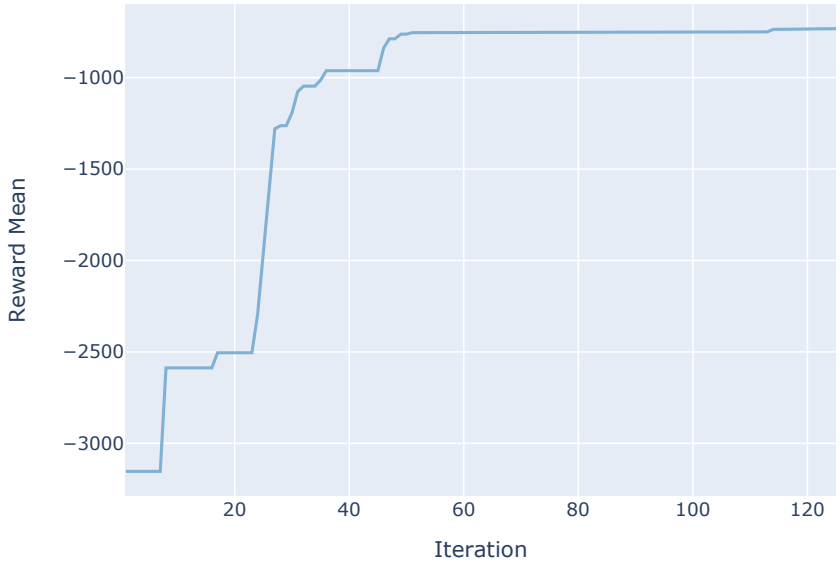
For two product types:

- 75 training iterations with 1 local warehouse.
- 100 training iterations with 2 local warehouses.
- 125 training iterations with 3 local warehouses.

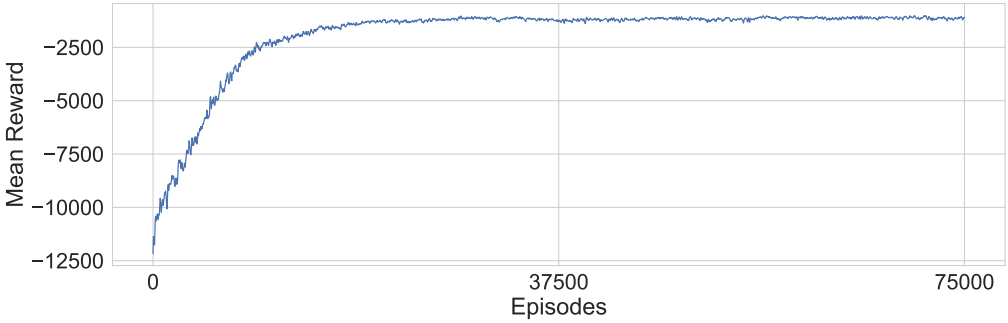
By adjusting the number of episodes and training iterations according to the size and complexity of the experiment, we ensure that each agent strikes a balance between computational resources and time. This approach promotes accurate and reliable results while guaranteeing consistent convergence for each agent, as shown in Figure A.1.



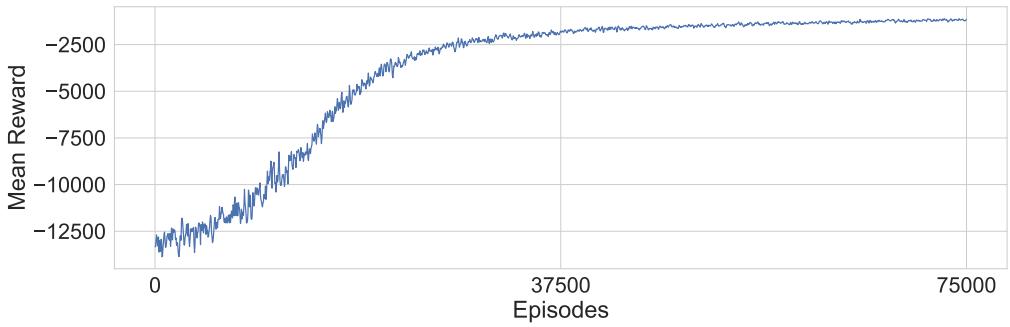
(a) sQ policy.



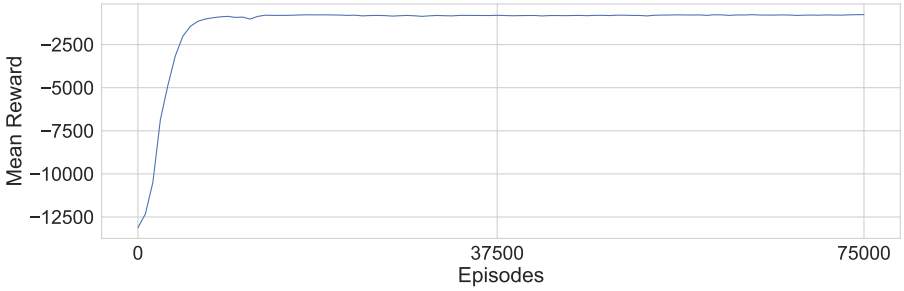
(b) BS policy.



(c) PPO.



(d) PG.



(e) A3C.

Figure A.1: Convergence trajectories during training of sQ policy (Figure A.1a), BS policy (Figure A.1b), PPO (Figure A.1c), PG (Figure A.1d), and A3C (Figure A.1e) in the most challenging experiment, which involves two product types, three local warehouses, and lead times of three. Plots related to other experiments conducted can be found in the GitHub repository associated with our open-source library (available at https://github.com/frenkowski/SCIMAI-Gym).

A.2 Training Times

Table A.2 presents the training times for the various agents implemented, calculated as averages across the two scenarios considered (i.e., with $L = 1$ and $L = 3$) for each combination between the number of product types and local warehouses. In particular, PG typically converges more rapidly than A3C and PPO, leading to shorter training durations. When examining static inventory policies, the (s, Q) policy usually takes a longer time to converge compared to the BS policy, primarily due to the increased number of parameters that need optimization. Ultimately, the training durations for DRL algorithms and static inventory policies are reasonably comparable.

Table A.2: The average training times (in minutes) for DRL algorithms and static inventory policies across the two values of lead times considered for one product type (Table A.2a) and two product types (Table A.2b) and for each number of local warehouses.

	PPO	PG	A3C	sQ	BS
$J = 1$	1.27 ± 0.18	1.18 ± 0.18	1.57 ± 0.46	1.06 ± 0.65	0.49 ± 0.10
$J = 2$	1.29 ± 0.17	1.16 ± 0.16	1.40 ± 0.17	1.78 ± 0.38	1.29 ± 0.17
$J = 3$	3.01 ± 0.55	2.70 ± 0.49	3.73 ± 0.59	3.30 ± 0.74	1.88 ± 0.50

(a) $I = 1$.

	PPO	PG	A3C	sQ	BS
$J = 1$	2.87 ± 0.59	2.52 ± 0.50	3.27 ± 0.79	1.79 ± 0.36	1.41 ± 0.10
$J = 2$	3.50 ± 0.50	2.83 ± 0.48	3.96 ± 0.63	2.68 ± 0.36	2.01 ± 0.43
$J = 3$	5.89 ± 1.47	5.31 ± 0.87	7.49 ± 2.03	3.71 ± 0.47	3.02 ± 0.48

(b) $I = 2$.

References

[1] Eytan Bakshy et al. "AE: A domain-agnostic platform for adaptive experimentation." In: *Conference on Neural Information Processing Systems*. 2018, pp. 1–8.

[2] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[3] Matthias Feurer and Frank Hutter. "Hyperparameter Optimization." In: *Automated Machine Learning*. Springer International Publishing, 2019, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1. URL: https://doi.org/10.1007/978-3-030-05318-5_1.

[4] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. "Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 17200–17211. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c7af0926b294e47e52e46cfebe173f20-Paper.pdf.

[5] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice." In: *Neurocomputing* 415 (Nov. 2020), pp. 295–316. DOI: 10.1016/j.neucom.2020.07.061. URL: https://doi.org/10.1016/j.neucom.2020.07.061.

Appendix B

Supplementary Material of Chapter 3

B.1 SCIM Environment Parameters

Table B.1 outlines the parameters defining the SCIM environment for small and large settings. For small experiments, the indication of two values is associated with $\epsilon \sim \mathscr{B}(0.5)$ and ϵ as in Equation (3.16), respectively. For large experiments, the exhibition of two values corresponds to $J = 5$ and $J = 10$ distribution warehouses, respectively. This comparative methodology enables the evaluation of the impact of distinct configurations on the overall performance and efficiency of our proposed techniques.

More specifically, in small settings, the demand amplitude remains constant at 5, whereas in large ones, it is set at 2. Production costs are consistent across experiments, with a value of 1. Vehicle capacities differ between the two settings. Small experiments accommodate a capacity of 3, with a fixed cost per vehicle of 0.7 and a variable cost per batch of 0.03. Conversely, large experiments allow for a capacity of 2, with fixed costs randomly ranging between 0.7 and 1 and variable costs that exhibit random variation within the range of 0.01 and 0.07.

Maximum production capacity also exhibits different values, with bounds set at 8 and 15 batches for small settings and 13 and 25 batches for large ones. Furthermore, factory capacities display differences, with values of 10 and 20 batches in small settings and 18 and 36 batches in large ones. Distribution warehouse capacities are lower than factory capacities, with 5 and 10 batches for small settings and large ones that consistently maintain a capacity of 6 batches. It is crucial to highlight that these configurations are strategic and designed to encourage agents to preserve stocks in advance and effectively avoid myopic behavior.

Concerning storage costs, factory storage costs are set to 0.1 in small experiments, while they randomly vary between 0.01 and 0.1 in large ones. In contrast, distribution warehouse storage costs remain constant at 1 for small settings and exhibit random variations between 1 and 2 for large settings. Finally, backorder costs remain fixed at 10 across all experiments.

Table B.1: Parameters defining the SCIM environment for small and large settings. In small settings, when two values are listed, the first is associated with the experiment with $\epsilon \sim \mathscr{B}(0.5)$, while the second refers to ϵ as in Equation (3.16). For the large settings, the provided two values correspond to experiments with $J = 5$ and $J = 10$ distribution warehouses, respectively.

Parameters	Small Settings	Large Settings
Maximum Demand Value	5	2
Production Costs	1	1
Vehicles Capacities	3	2
Logistic Costs Fixed	0.7	random(0.7, 1)
Logistic Costs Variable	0.03	random(0.01, 0.07)
Maximum Production	{8, 15}	{13, 25}
Factory Capacities	{10, 20}	{18, 36}
Distribution Warehouses Capacities	{5, 10}	{6, 6}
Factory Storage Costs	0.1	random(0.01, 0.1)
Distribution Warehouses Storage Costs	1	random(1, 2)
Backorder Costs	10	10

B.2 Hyperparameters Selection

Selecting appropriate values for hyperparameters in neural networks is complex and time-consuming, as extensively discussed in the relevant literature [2, 5]. Hyperparameters play a crucial role in the context of DRL algorithms since they can significantly influence training and, consequently, relative performance [1]. Our choice of hyperparameters is based on the Ray documentation and in observance of the discussions presented in Gijsbrechts et al. [3] and Stranieri and Stella [4].

Table B.2 lists the selected hyperparameters for the underlying multilayer perceptron with two hidden layers, along with their corresponding values. Through a grid search, the PPO algorithm was trained for 75k episodes for the two experiments belonging to the small settings, 100k episodes for the experiment of the large setting comprising 5 distribution warehouses (i.e., $J = 5$), and 200k episodes for the experiment of the large setting involving 10 distribution warehouses (i.e., $J = 10$). The results presented in the paper refer to the best combination of hyperparameters identified, for which we replicated the training and testing procedures.

Moreover, we assess the PPO agent to access the demand values of the previous timesteps (i.e., referring to Equation (3.3), we set $\tau = 2$ in small experiments and $\tau = 3$ in large ones). This choice reflects a balance between providing the PPO agent with enough information to make informed decision-making while not overwhelming it with excessive historical data, which might introduce unnecessary complexity and longer training times.

Table B.2: Hyperparameters of the PPO algorithm selected for tuning, along with their corresponding values. Through a grid search, we search for the best combination of hyperparameters for each experiment.

PPO Hyperparameters	Values
Neurons per Hidden Layer	{(16, 16), (32, 32), (64, 64), (128, 128)}
Learning Rate	{5e-4, 5e-5, 5e-6}
Train Batch Size	{200, 400, 800, 2000, 4000, 8000}
Stochastic Gradient Descent Mini-Batch Size	{64, 128, 256, 512}
Stochastic Gradient Descent Iterations	{15, 30, 45}

The parameters governing the (s, Q)-policy are determined using a data-driven approach based on Bayesian optimization, as presented in Stranieri and Stella [4].

Finally, DRLBD exploits the best combination of PPO hyperparameters as previously described, while to generate the scenario tree it employs a Monte Carlo technique known as moment matching.

References

[1] Robert N. Boute et al. "Deep reinforcement learning for inventory control: A roadmap." In: *European Journal of Operational Research* 298.2 (Apr. 2022), pp. 401–412. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.07.016. URL: http://dx.doi.org/10.1016/j.ejor.2021.07.016.

[2] Matthias Feurer and Frank Hutter. "Hyperparameter Optimization." In: *Automated Machine Learning*. Springer International Publishing, 2019, pp. 3–33. DOI: 10.1007/978-3-030-05318-5_1. URL: https://doi.org/10.1007/978-3-030-05318-5_1.

[3] Joren Gijsbrechts et al. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems." In: *Manufacturing & Service Operations Management* 24.3 (May 2022), pp. 1349–1368. DOI: 10.1287/msom.2021.1064. URL: https://doi.org/10.1287/msom.2021.1064.

[4] Francesco Stranieri and Fabio Stella. "Comparing Deep Reinforcement Learning Algorithms in Two-Echelon Supply Chains." In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer Nature Switzerland, 2025, pp. 454–469. ISBN: 9783031746406. DOI: 10.1007/978-3-031-74640-6_37. URL: http://dx.doi.org/10.1007/978-3-031-74640-6_37.

[5] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice." In: *Neurocomputing* 415 (Nov. 2020), pp. 295–316. DOI: 10.1016/j.neucom.2020.07.061. URL: https://doi.org/10.1016/j.neucom.2020.07.061.

Appendix C

Supplementary Material of Chapter 4

C.1 Proof of Lemma 1

Proof. We know that $q_t = 0$ for $t = T - L + 1, \dots, T$, and $\sum_{i=1}^{m-1} x_{T+1,i} = (Y_T - D_T)^+ - (X_{T,1} - D_T)^+$. Additionally, the following relations hold: $q_{t-L} = (Y_{t+L-1,1} - D_{t+L-1})^+ - (Y_{t+L-1} - D_{t+L-1})^+ + Y_{t+L}$, and $Y_{t+L} = (Y_{t+L} - D_{t+L})^+ - (D_{t+L} - Y_{t+L})^+ + D_{t+L}$.

Substituting these relations, we can express the total cost C^π as follows:

$$\begin{aligned} C^\pi = &\sum_{t=1}^{T-L} \left(\hat{c}(X_{t+L-1,1} - D_{t+L-1})^+ - \hat{c}(Y_{t+L-1} - D_{t+L-1})^+ + \hat{c}Y_{t+L}\right) + \\ &\sum_{t=L+1}^{T} \left(\hat{\theta}(X_{t,1} - D_t)^+ + \hat{h}(Y_t - D_t)^+ + \hat{b}(D_t - Y_t)^+\right) - \\ &\hat{c}\left((Y_T - D_T)^+ - (X_{T,1} - D_T)^+\right). \end{aligned} \tag{C.1}$$

Rearranging terms, we obtain three key relations:

$$\sum_{t=1}^{T-L} \hat{c}\left((Y_{t+L} - D_{t+L})^+ - (Y_{t+L-1} - D_{t+L-1})^+\right) - \hat{c}(Y_T - D_T)^+ = 0 \tag{C.2}$$

$$\sum_{t=1}^{T-L} \hat{c}(X_{t+L-1,1} - D_{t+L-1})^+ + \hat{c}(X_{T,1} - D_T)^+ = \sum_{t=L+1}^{T} \hat{c}(X_{t,1} - D_t)^+ \tag{C.3}$$

$$\sum_{t=1}^{T-L} \left(\hat{c}D_{t+L} - \hat{c}(D_{t+L} - Y_{t+L})^+\right) = \sum_{t=L+1}^{T} \hat{c}\left(D_t - (D_t - Y_t)^+\right) \tag{C.4}$$

These relations hold because the inventory level is depleted at timestep $t = 1$ and remains so until timestep $t = L$. By combining (C.2), (C.3), and (C.4), we establish the result stated in Lemma 1. □

C.2 Proof of Lemma 2

Proof. From the expression for the expired quantity at timestep t and the order of events, we have:

$$\begin{aligned} O_t &= \left(s - \sum_{j=t-L-m+1}^{t} D_j + \sum_{j=t-m-L+1}^{t-1} l_j - \sum_{j=t-m-L+1}^{t-1} O_j \right)^+ \\ &\geq \left(s - \sum_{j=t-m-L+1}^{t} D_j \right)^+ - \sum_{j=t-m-L+1}^{t-1} O_j. \end{aligned} \tag{C.5}$$

Taking the expectation of the left- and right-hand sides yields:

$$\mathbb{E}\left[O_t\right] \geq \frac{\mathbb{E}\left[\left(s - \mathscr{D}_{m+L}\right)^+\right]}{m+L}. \tag{C.6}$$

For any $t > L$, we can write:

$$\left(\sum_{i=1}^{m} x_{t,i} - D_t \right)^+ \geq \left(s - \sum_{j=t-L}^{t} \xi_j \right)^+ - \sum_{j=t-L}^{t-1} O_j, \tag{C.7}$$

and

$$\left(D_t - \sum_{i=1}^{m} x_{t,i} \right)^+ \geq \left(\sum_{j=t-L}^{t} \xi_j - s \right)^+ - \sum_{j=t-L}^{t-1} l_j. \tag{C.8}$$

This leads to:

$$\begin{cases} \mathbb{E}\left[\left(\sum_{i=1}^{m} x_{t,i} - D_t\right)^+\right] \geq \mathbb{E}\left[\left(s - \mathscr{D}_{L+1}\right)^+\right] + \frac{L\mathbb{E}\left[\left(s-\mathscr{D}_{m+L}\right)^+\right]}{m+L} \\ \mathbb{E}\left[\left(D_t - \sum_{i=1}^{m} x_{t,i}\right)^+\right] \geq \frac{\mathbb{E}\left[\left(\mathscr{D}_{L+1}-s\right)^+\right]}{L+1} \end{cases}. \tag{C.9}$$

□

C.3 Bounds Value Analysis

Table C.1 presents the lower and upper bounds for both the OUT and PIL policies, calculated under the worst-case setting described in Section 4.4, along with the corresponding optimal values obtained.

Both policies exhibit similar trends. The upper bounds consistently align with or slightly exceed the optimal values, confirming their reliability. Conversely, the lower bounds remain equal to or below the optimal values, providing a conservative estimate. Interestingly, while the lower bounds are identical for both policies, the upper bound for the PIL policy is generally lower than that of the OUT policy. This observation suggests that the PIL policy offers more constrained bounds, possibly contributing to its cost-effectiveness in certain scenarios.

Table C.1: Lower (LB), upper (UB), and optimal (OPT) values for the OUT and PIL policies, with demand noise modeled as $\xi_t \sim \mathcal{N}(0, \bar{d} \times 15\%)$, where $\bar{d} = \max_t d_t$. The results are shown for parameter values $m = \{2, 3, 4\}$, $w = \{2, 4\}$, and $b = \{10, 50, 100, 1000\}$.

		OUT LB				OUT OPT				OUT UB				PIL LB				PIL OPT				PIL UB			
m	w	b				b				b				b				b				b			
		10	50	100	1000	10	50	100	1000	10	50	100	1000	10	50	100	1000	10	50	100	1000	10	50	100	1000
2	2	2	4	5	7	2	5	6	8	4	6	7	9	2	4	5	7	3	4	5	7	3	5	6	8
	4	1	4	4	7	2	5	5	8	4	6	7	9	1	4	4	7	2	4	5	7	2	5	5	8
3	2	2	4	5	7	3	5	6	8	4	6	7	9	2	4	5	7	3	5	6	8	3	5	6	8
	4	1	4	5	7	3	5	6	8	4	6	7	9	1	4	5	7	3	5	6	8	3	5	6	8
4	2	2	4	5	7	3	5	6	8	4	6	7	9	2	4	5	7	3	6	6	8	3	5	6	8
	4	2	4	5	7	3	5	6	8	4	6	7	9	2	4	5	7	3	6	6	8	3	5	6	8

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was LuaLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.