MASTER

Industrializing Deep Reinforcement Learning for ASML's Service Network

van der Haar, Joost H.

Award date:
2023

Link to publication
Eindhoven University of Technology

Master Thesis

**Industrializing Deep Reinforcement Learning for ASML’s Service Network**

In partial fulfillment of the requirements for the degrees of Master of Science in Operations Management and Logistics and Master of Science in Data Science in Engineering

Veldhoven, September 24, 2023

J.F. (Joost) van der Haar
j.f.v.d.haar@student.tue.nl
1007432

_Supervisors_

**Operations Management and Logistics**
Dr. Ir. R.J.I. (Rob) Basten
Dr. W. (Willem) van Jaarsveld

**Data Science in Engineering**
Dr. P. (Pratik) Gajane
Prof. Dr. M. (Mykola) Pechenizkiy

**ASML**
Ir. T. (Tim) Kragten
Ir. T.H.P. (Tom) Vlassak, PDEng
Ir. J. (Joan) Stip, PDEng
Abstract

ASML’s customer service network aims to ensure timely availability of the materials needed for machine maintenance. Most operational decisions in this network are automated by the internally developed NORA algorithm, which bases its decisions on relatively simple decision rules. These decision rules are known to lead to good yet suboptimal decisions. More advanced methods exist, but they struggle with scalability. We propose a new deep reinforcement learning (DRL) method to take operational decisions in near real-time, and evaluate it through a series of studies. We contribute to academic literature by combining reward shaping, action shaping and global models to enable training a DRL model for up to 10,000 SKUs and 60 locations. Moreover, we find that this proposed DRL model outperforms NORA on real data in terms of cost while only having limited impact on service performance metrics, thereby highlighting our business contributions.
Executive Summary

Problem and Methodology

Machines sold by ASML require maintenance to maximize uptime and thereby throughput. The parts and tools needed to perform maintenance are managed on an operational level by the Customer Supply Chain Management department, which is tasked with meeting ASML’s material availability commitments to its customers as cost-efficiently as possible. It is aided in this task by the Network Oriented Replenishment Automation (NORA) algorithm, which automates most operational level decisions. NORA does so using relatively simple decision rules, which are known to lead to good yet suboptimal decisions.

More advanced methods for operational planning are available, but they struggle with scalability. We propose a new deep reinforcement learning (DRL) method to take operational decisions. DRL is a promising technique for many inventory management problems, and can take decisions in near real-time once trained. However, research has thus far failed to train DRL models for realistic problems of comparable size and complexity (Boute et al., 2022). This thesis therefore focuses on industrializing DRL for service logistics. We propose new methods to scale DRL to spare parts problems of the required size, in addition to showing how DRL can outperform NORA when faced with complexities such as time-bound demand in the form of service orders.

To assess scalability, we include all service locations and parts within our scope. To assess ability to handle different types of time-bound demand, we include routine, priority and emergency service orders within our scope. Furthermore, we include proactive lateral transshipments, contract performance and the emergency sourcing system (GES) in our scope. We exclude local sourcing, sales orders and demand for upgrades, installs and relocations (UIR) from our scope.

Experiments and Results

We perform three studies to evaluate our method, namely an ablation study, a benchmark study and a manual validation study. The ablation study demonstrates that our scaling techniques allow our DRL model to outperform NORA on a synthetic problem with 60 locations and 10,000 SKUs, thereby improving upon literature as summarized in Table 1.

<table>
<thead>
<tr>
<th>Study</th>
<th>Type</th>
<th>Problem</th>
<th>#SKUs</th>
<th>#locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dmitrochenko (2020)</td>
<td>Master thesis</td>
<td>Fully connected network</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Gijsbrechts et al. (2022)</td>
<td>Journal article</td>
<td>Distribution system</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Van Hezewijk et al. (2022)</td>
<td>Journal article</td>
<td>Capacitated lot sizing</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>Vanvuchelen et al. (2023a)</td>
<td>Journal article</td>
<td>Fully connected network</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Vanvuchelen et al. (2023b)</td>
<td>Preprint</td>
<td>Joint replenishment</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>This study</td>
<td>Master thesis</td>
<td>Fully connected network</td>
<td>10,000</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 1: Scale comparison with earlier studies using DRL for inventory management

Our benchmark study compares our DRL model performance to that of NORA by running simulation runs on real ASML data for 100 randomly selected SKUs, and finds that our DRL model is able to achieve cost savings of 5.14%. Moreover, we perform one-sided independent samples t-tests for each SKU separately. The results indicate that DRL significantly outperforms NORA for 48% of SKUs, while only being significantly outperformed for 6% of SKUs. Table 2 provides summary statistics of the results. Figure 1 plots the distribution of the cost ratios for the 100 SKUs, which are defined as the sum of DRL model costs divided by the sum of NORA costs.
We perform multiple analyses on the benchmark study results to gain insight into the performance of DRL and differences between actions proposed by DRL and NORA, leading to four main insights.

1. The DRL model makes use of proactive lateral transshipments, leading to reductions in service order waiting times.
2. Cost savings are achieved over all types of service orders, indicating robustness to changes in cost input parameters.
3. Location-based differences in service performance metrics between DRL and NORA are only minor, suggesting DRL does not consistently favor some locations over others.
4. The DRL model performs best for SKUs where replenishments arrive at the Netherlands’ central warehouse rather than the Korean central warehouse, and for SKUs where there are relatively many locations with demand.

The DRL model is manually validated in collaboration with a supply chain engineer from ASML by applying it to the data for a date in the past, such that it can be directly compared to NORA. We investigate decisions for 20 SKUs in detail, leading to two main observations.

1. DRL seems to follow a strategy that sources from nearby locations as much as possible, potentially proactively.
2. The order in which demand locations are prioritized differs between DRL and NORA, but arguments can usually be made for either order. Overall, the actions proposed by the DRL model appear to be reasonable, both in terms of overall strategy and local prioritization.

Conclusions and Recommendations

Based on the experimental results, we conclude that DRL is demonstrably cost-efficient, computationally efficient and able to meet material availability commitments. Our results underline our academic contributions to industrializing DRL for service logistics, while also demonstrating business benefits. However, our study is not without its limitations and opportunities for further improvement. Our scope excludes service tools, sales orders and UIR demand. Moreover, we rely on assumptions for some of the modelled processes and input parameters due to unavailability of data. We hence recommend implementation of DRL based on our experimental results, but anticipate that significant time and effort is still required to address limitations before implementation.

To overcome the remaining challenges while simultaneously reaping short-term benefits, we recommend a gradual move towards DRL implementation through the following short-term measures:

- Build a digital twin on the operational level to quantitatively evaluate and improve NORA decision rules in the short-term, and to train a DRL model on in the long-term.
• Further analyze and chart the demand fulfillment process to identify improvement opportunities in the short-term, and reduce the amount of required DRL assumptions in the long-term.
• Develop basic machine learning (ML) competency among NORA developers to enable simple analyses and facilitate communication with ML-specialized teams in the short-term, thereby building a knowledge base for long-term DRL implementation.

If these short-term measures are followed, implementation of a DRL model is only a relatively small step. To maximize performance, we recommend the following next steps during or after implementation:

• Investigate alternative methods to incorporate contractual commitments, for example to allow dynamically adapting to a location’s contract status.
• Extend the scope of the model to include service tools and GESA’s decision-making process.
• Optimize DRL model performance by increasing available computational power and improving data engineering.

We also see value in further academic research on topics related to those covered in this thesis. To this end, we suggest the following directions for future research:

• Explore incorporating contractual commitments through location-specific cost multipliers, thereby taking an approach akin to Lagrangian relaxation.
• Perform research into interpretability and explainability methods specifically tailored for applications of DRL to inventory management.
• Initiate case studies on monitoring and maintenance of DRL models for inventory management to identify frequently faced challenges and establish best practices.
Preface

The thesis that lies before you is the culmination of ten months of work and marks the end of my time as a student. It has been a great time, with a surprising trajectory from architecture to logistics and to data science. Although the path may not have always been straightforward, I thoroughly enjoyed the journey and am grateful to the people I shared it with. I would therefore like to take the opportunity to thank a few people who made this journey possible.

First of all, I would like to express my gratitude to those involved from university. Rob, thank you for your support, feedback and insights. Not just during this thesis, but also when discussing the possibility of choosing two Master programs, as a mentor during the honors program and as a supervisor for my internship in Leuven. Willem, thank you for our many interesting and helpful discussions on scoping, prioritization and technical details. Pratik, thank you for always being there with good advice on academic rigor and relevance, both when performing experiments and when writing. I would also like to express my gratitude to Tarkan for quickly getting me up to speed with Dynaplex, and to Mykola for completing my assessment committee.

Second, I am thankful to everyone involved from ASML. Tim, thank you for ensuring that I always kept the practical challenges, complexities and exceptions in mind, and for being there for my many questions. Tom, thank you for your insights and for your help whenever I hit a roadblock. Joan, thank you for your practical advice and guidance. Rutger, Tanya and Alard, thank you for our discussions and for your constructive feedback. I am also grateful to all other people from SBA and IFPP who welcomed me during my period here.

Finally, I would like to express my appreciation to my family and friends who supported me during my thesis and my studies. You were always there for me. Thank you for the joyful evenings, the chaotic trips and the many fun yet surprisingly productive work sessions. My time as a student would not have been the same without you.


Joost van der Haar

September 2023
## Contents

**Introduction**  
1 Background  
1.1 Strategic Landscape  
1.2 Tactical Planning  
1.3 Operational Execution  
1.4 Opportunities for Improvement  

2 Problem Definition  
2.1 Objective and Requirements  
2.2 Scope  
2.3 Approach  

3 Methodology  
3.1 Existing Methods  
3.2 Deep Reinforcement Learning  
3.3 Conclusion  

4 Conceptual Design  
4.1 Choice of Programming Language  
4.2 Choice of Algorithm  
4.3 Simulation Environment  
4.4 Scaling Deep Reinforcement Learning  

5 Detailed Design  
5.1 Implementation Architecture  
5.2 Markov Decision Process  
5.3 Evaluation Algorithm  
5.4 Base Agent  
5.5 Feature Engineering  
5.6 Approximations and Assumptions  

6 Ablation Study  
6.1 Data Generation  
6.2 Experimental Setup  
6.3 Results  

7 Benchmark Study  
7.1 Input Files and Scope  
7.2 Experimental Setup  
7.3 Results and Analyses  
7.4 Conclusion
Introduction

ASML is a leading producer of lithography systems. These systems form an essential link in the semiconductor manufacturing process. In light of the growing global demand for chips, ASML is ramping up production. The rapid increase in its production is accompanied by an increasing need for the parts and tools necessary to build the machines. Simultaneously, the same parts and tools are required to maintain the machines already out in the field. Making efficient use of the parts and tools available for maintenance is therefore crucial.

The department in ASML responsible for its day-to-day material availability in the field is the Customer Supply Chain Management department. This department is tasked with meeting ASML’s material availability commitments to its customers in as cost-efficient a manner as possible. It is aided in this task by the internally developed Network Oriented Replenishment Automation (NORA) algorithm, which automates most operational level decisions. NORA does so using relatively simple decision rules, which are known to lead to good yet suboptimal decisions.

More advanced methods for operational planning are available, but they struggle with scalability. We propose a new Deep Reinforcement Learning (DRL) method to take operational decisions. DRL is a potent technique for sequential decision-making problems, and can take decisions in near real-time once trained. However, research has thus far failed to train DRL models for realistic problems of comparable size and complexity (Boute et al., 2022).

This thesis improves upon the state-of-the-art by successfully training a DRL model for ASML’s service network, which contains more than 60 storage locations and 10,000 SKUs. Our academic contribution is threefold. First, we introduce a new variance reduction technique that approximates the expected reward in a given period. Second, we further establish the importance of action representation by proposing a novel way to formulate the action space. Third, we confirm the conjecture by Boute et al. (2022) that cross-SKU meta-learning can greatly improve scalability.

We find that the scaled DRL model meets business requirements, thereby highlighting our business contributions. Experiments demonstrate that our scaled DRL model is cost efficient, computationally efficient and able to meet material availability commitments. Furthermore, it can take decisions based on the data that would be available to it in practice, and is suitable for further scope expansions.

The thesis proceeds as follows. Chapter 1 elaborates on the context in which NORA operates, and delves into how it could be improved. Chapter 2 formalizes the investigated problem. The choice of method is motivated in Chapter 3, after which major design choices and our academic contributions are highlighted in Chapter 4. Chapter 5 provides a detailed overview of the proposed DRL method. Chapter 6 assesses our academic contributions through an ablation study, whereas Chapter 7 examines our business contributions through a benchmark study and Chapter 8 manually validates our method. Finally, Chapter 9 draws conclusions, discusses strengths and weaknesses, and gives recommendations for business follow-up and further research.

As this Master thesis is a joint thesis for the Data Science in Engineering and Operations Management and Logistics Masters, some terms may be unfamiliar to readers without a background in one of the two. We have therefore included Appendix A, which contains descriptions and explanations for terms and abbreviations used in this thesis. This appendix may also be used to look up terms and abbreviations specific to the investigated problem or to ASML, although these terms and abbreviations are also introduced in the main body of the thesis.
1 Background

This chapter provides a survey of the context in which NORA operates. Section 1.1 starts by examining the strategic landscape in which ASML operates, and the role played by Service Level Agreements (SLAs). Next, Section 1.2 describes how service is delivered through ASML’s service logistics network and tactical planning. This description is followed by an overview of the operational execution in Section 1.3. Finally, Section 1.4 discusses several potential improvement directions for NORA.

1.1 Strategic Landscape

1.1.1 Company Background

Lithography is a crucial step in the semiconductor manufacturing process. ASML designs, produces and services lithography systems that perform and support this step. It is a key supplier to all major semiconductor manufacturers, including TSMC, Samsung and Intel. ASML’s product portfolio consists of three main business lines, namely Deep UltraViolet (DUV) lithography systems, Extreme UltraViolet (EUV) lithography systems and so-called ‘applications’. The first two regard the different types of machines used for executing the lithography step. Applications on the other hand is an umbrella term for various machine types used for optimizing the lithography step.

Most of ASML’s activities take place on the Veldhoven campus, where more than half of its 39,000 employees work. Nonetheless, the company has a broad global presence with more than 60 locations worldwide. Its 2022 revenue was €21.2B (ASML Holding N.V., 2023). This revenue was generated by ASML’s two main business activities, namely sales of lithography systems and after-sales service and field options for the same systems. The former yielded €15.4B in net sales, whereas the latter led to €5.7B in net sales (ASML Holding N.V., 2023). Different kinds of after-sales contracts are offered to customers. Contract options include full service contracts, pre-paid service contracts and parts purchasing availability contracts.

Three types of service contracts can be distinguished, namely global, continental and local parts availability contracts. They differ both in terms of service level commitments and costs. Customers with Local Parts Availability (LPA) contracts are assigned to nearby local warehouses, whereas those with Continental Parts Availability (CPA) contracts are assigned to more distant local warehouses. Lastly, customers with Global Parts Availability (GPA) contracts are assigned to global warehouses. Service level commitments are captured in service level agreements.

1.1.2 Service Level Agreements

Customers opt for after-sales service and field options to ensure machine availability. Machines require maintenance from time to time, and the resulting downtime can lead to considerable opportunity costs. Material availability is necessary to quickly perform the required maintenance, which is where after-sales contracts come in. Each contract contains service level agreements (SLAs). The SLAs stipulate how service performance is measured, and what level of service performance ASML commits to. When they are not met, ASML has to pay considerable penalty costs to its customers.

Service performance is measured using three types of metrics, namely material availability, waiting time and delivery response time metrics. Material availability commitments can be found in LPA, CPA and GPA contracts. Waiting time commitments can be found in LPA and CPA contracts. Finally, delivery response time commitments can be found only in LPA contracts.
Material Availability

Material availability is measured using Customer Service Degree (CSD) and Non-Availability (NAV) metrics. The CSD metric measures the proportion of service material demand that can be fulfilled directly from local stock. It is therefore equivalent to the aggregate fillrate in literature, as introduced by Feeney and Sherbrooke (1965). Its complement is the NAV metric, which measures the proportion of non-availabilities: demand that cannot be fulfilled directly from local stock. CSD and NAV can be calculated empirically as follows:

\[
CSD = \frac{\text{Demand met locally}}{\text{Total demand}}, \quad \text{and} \quad NAV = 1 - CSD.
\]

Waiting Time

Although availability metrics are straightforward to calculate, they ignore the importance of downtime duration. Down waiting for materials (DWM) and downtime waiting for parts (DTWP) do take the downtime duration into account. DWM measures the time waiting for materials retrospectively and empirically, whereas DTWP can also be prospective and an estimate. Both are related to the aggregate mean waiting time as used in literature (Berg and Posner, 1985), but differ by measuring waiting time from a machine perspective rather than a service material perspective. Mathematically, DWM and DTWP are defined as follows:

\[
\text{DWM} = \frac{\text{Time down waiting for materials}}{\text{Total time}}, \quad \text{and} \quad \text{DTWP} = \frac{\text{Demand met locally} \times \mathbb{E}[\text{Local leadtime}] + \text{Non-availabilities} \times \mathbb{E}[\text{Emergency leadtime}]}{\text{Total time}}.
\]

Looking at only DWM and DTWP might however be misleading. Time waiting for service materials is only a small part of the total downtime, which also depends on other processes. Engineers first need to diagnose a problem, after which they compile a list of required materials and tools for maintenance. Once they arrive, defective parts are replaced and a recovery sequence is started.

Delivery Response Time

Another approach that takes downtime duration into account is the Delivery Response Time (DRT) approach, which measures performance in terms of eXtreme Long Downs (XLDs) and Hit Ratio. Long machine downs can lead to congestion issues in factories, as work-in-progress products keep arriving but can no longer be processed. Consequently, the output of other machines might need to be reduced due to finite buffer capacity (Missbauer and Uzsoy, 2020), leading to considerable opportunity costs. ASML customers hence prefer having many different short downs to having few long downs. The number of XLDs can be calculated using (Lamghari-Idrissi et al., 2020):

\[
\text{XLD} = \text{Non-availabilities} \times \mathbb{P}[\text{Emergency leadtime} > \text{Threshold}].
\]

where the threshold is location-dependent. XLD serves as a key building block for the Hit Ratio service measure, which can be seen as a type of leadtime-centered aggregate fillrate. Hit Ratio is defined as follows (Lamghari-Idrissi et al., 2022):

\[
\text{Hit Ratio} = \frac{\text{Total demand} - \text{XLD}}{\text{Total demand}}.
\]
1.2 Tactical Planning

1.2.1 Service Network

To meet the SLAs, service materials required for maintenance need to be at the right place at the right time. ASML’s Customer Supply Chain Management (CSCM) department hence maintains a global service logistics network, which services ASML systems at different customer semiconductor manufacturing plants (fabs) all over the world. These systems are collectively referred to as the installed base. The installed base is served through a network of two central warehouses (CWs) and many local warehouses (LWs). Service materials (re-)enter the service network at one of the CWs from manufacturing and repair plants. CWs subsequently use them to replenish LWs, although lateral transshipments from any LW to any other LW are also possible. Service materials leave the network if they are consumed during a maintenance activity. Figure 2 provides a graphical representation of the service logistics network.

Two main types of service materials can be distinguished in the network, namely service parts and service tools. Service parts are defined as materials that go into machines during maintenance activities, whereas service tools may be returned directly to stock after a maintenance activity has finished. Further subtypes of service parts and tools also exist, such as consumables and perishables. Consumable materials get depleted through usage. Perishable materials degrade over time, and might become unusable or require recalibration for quality or safety reasons.

Service materials can be sourced in different ways, each with their respective leadtimes. These leadtimes are uncertain. Leadtimes for replenishments and lateral transshipments generally depend on a combination of warehouse processing time and availability of means of transport. New-buy leadtimes are dependent on both the supplier leadtime and the transportation time. Similarly, repair leadtimes are dependent on both the repair leadtime and the transportation time. Supplier leadtimes and repair times can have high variance, whereas approximate transportation times are usually known assuming means of transport are available.
1.2.2 Tactical Decisions

Tactical decisions are taken in a two-step process. In the first step, forecasters use a combination of forecasting algorithms and expert knowledge to predict demand arrival rates under an assumption of Poisson demand. This assumption is reasonable when machine lifetimes are exponentially distributed, or when a local warehouse serves sufficient machines with independent and generally distributed lifetimes (Van Houtum and Kranenburg, 2015). Unsurprisingly then, Van den Oord (2021) found that the assumption is indeed often reasonable on a local level for the failure behavior of service parts at ASML.

In the second step, the forecasts are used to produce two types of tactical planning outputs. Reorder Points (ROPs) are a type of basestock level, and can be seen as a maximum inventory level. Safety Stock Levels (SSLs) are a type of holdback level, and can be seen as a target minimum inventory level. The ROPs are set to meet material availability and waiting time commitments, and are both used for capacity planning and operational planning. The SSLs are set to meet delivery response time requirements, and are solely used for operational planning. Both are initially produced by the SPartAn planning algorithm, after which they can be manually enriched by planners.

1.2.3 SPartAn

The SPartAn planning algorithm consists of three components. The core of the algorithm is proposed in Kranenburg and Van Houtum (2009). They analyze a system with two types of local warehouses, namely main local warehouses and regular local warehouses. Main local warehouses can send and receive reactive lateral transshipments, whereas regular local warehouses can only receive them. In their analysis, the authors assume use of a non-differentiated basestock policy, IID exponential replenishment leadtimes, FCFS replenishment and emergency sourcing in a fixed, location-specific order. Exact evaluation of this system is computationally too costly, Kranenburg and Van Houtum (2009) therefore propose a much faster approximate evaluation procedure.

The core idea of this procedure is to approximate the demand arrival process at main LWs by a Poisson process. It starts by calculating the fillrate of regular LWs exactly, after which the overflow demands towards the associated main LW are approximated using a Poisson distribution. Next, the procedure iteratively calculates approximations of the fillrates and overflow demands for the main LWs until the overflow rates between them stabilize. Remaining metrics can then be calculated using the fillrates of the main and regular LWs. For a more detailed description of the procedure, the reader is referred to Appendix B.1.

Van Aspert (2015) use this approximation procedure as the core of their greedy algorithm, where they use the results as input for their greedy value estimates. Their greedy optimization algorithm considers two echelons, namely the Central Warehouses (CWs) and the Local Warehouses (LWs). CWs are subject to fillrate constraints, whereas LWs are subject to waiting time constraints. The greedy algorithm can be summarized in the following steps:

1. Set initial ROPs based on business input.
2. Greedily increase CW stock until fillrate constraints are met for their direct customers.
3. Greedily increase CW and LW stock until no cost improvements are attained anymore.
4. Greedily increase LW stock until waiting time constraints are met, and calculate the resulting overflow demand towards the CWs.
5. Greedily increase CW stock until fillrate constraints are met for their direct customers, in case they were no longer met due to overflow demand from LWs.
The greedy algorithm by Van Aspert (2015) in turn serves as the core of the currently used algorithm by Lamghari-Idrissi (2021). Unlike its predecessors, Lamghari-Idrissi (2021) applies service differentiation through a strategy similar to critical level policies. Inventory is divided into a premium XLD-based pool (SSLs) and a non-premium pool (ROPs minus SSLs). Customers with a premium contract can access both pools, while other customers can only access the non-premium pool. The algorithm by Lamghari-Idrissi (2021) can be summarized in the following steps:

1. Set initial SSLs for premium storage locations based on business input.
2. Approximate the overflow demand of premium contract customers towards the non-premium pool using a Poisson distribution with a one-moment fit.
4. If XLD targets are met, stop. Else, greedily increase SSLs until they are met.
5. Re-approximate premium contract overflow demand towards the non-premium pool.
6. Run the algorithm by Van Aspert (2015) in reverse to decrease the non-premium pool in case the decrease in premium contract overflow demand allows it.

1.3 Operational Execution

1.3.1 Operational Processes

Operational processes tend to be more complex than tactical planning assumptions suggest. Demand for service materials can be split into three main groups. Service orders (SOs) specify that a material is needed at a certain ultimate need date (UND), and can differ on shipment priority. Sales orders involve the delivery of a material to a customer around the agreed upon new-buy leadtime. Lastly, upgrades, installs & relocations (UIR) demand is demand for service materials used for scheduled events. UIR demand is planned using event item objects on a regional basis through HERO, which is developed by the Service Automation and Engineering (SAE) team.

Holistic Event Reservation Overview (HERO) is a platform that provides an overview of all scheduled events, including the materials and tools necessary for performing them. Scheduled events require a pre-specified list of materials to be available before they start. Central UIR coordinators enter their material requirements in HERO. The HERO platform then ensures that the scheduled event demand is visible to both NORA and the field material availability teams, such that materials are made available when possible and shortages are known in advance when not. Simultaneously, it takes input from NORA to see if and how scheduled event demand is covered.

1.3.2 Operational Decisions

Decisions on how exactly service material demand is fulfilled are taken operationally. Materials can be sourced from different locations and with different priorities. Three types of shipment priorities can be distinguished. Regular shipments are the cheapest yet slowest option. Priority shipments are slightly faster, but also more expensive. Lastly, emergency shipments are the fastest but also most expensive option. Regular shipments are therefore often used for replenishment and order fulfillment, whereas emergency shipments are used when a service material is urgently needed. When service orders remain unfulfilled for too long, their priority level might be increased.

Shipment priority also affects the system that is used to fulfill demand. The Global Emergency Sourcing Automation (GESA) system is responsible for among others order fulfillment, replenishments and last-mile scheduling in case of automated emergency orders. Its focus is on fulfilling
emergency orders as fast as possible, while disregarding costs. It runs almost real-time, and was
developed by the IT department with the aim of running as quickly as possible. The SAE team
serves as a technical interface between business users and the IT department.

GESA works by looking for the nearest location where stock is available. Stock availability depends
on the SLA for the machine group where stock is needed and on the SSL for possible sourcing
locations. If stock is needed for a machine group with a DRT contract, any location with stock can
be sourced from. If stock is needed for a machine group without a DRT contract however, no SSL
stock can be accessed. This constraint might even result in emergency sourcing from the factory
instead, whereby a part is removed from a machine that is in the process of being built.

1.3.3 NORA

The Network Oriented Replenishment Automation (NORA) algorithm is responsible for the au-
tomation of non-emergency operational processes. NORA was first developed by Bakker (2016)
and has been developed further over the years by the SAE team. It starts every day by performing
a material availability check. If found to be necessary, it triggers material ordering, warehouse
replenishments and product movements. NORA’s activities also include among others preparing
worklists, providing dashboards and creating purchase orders.

NORA performs operational planning in a three-step process. Firstly, it creates a list of all available
supply and open demand sorted by priority. Secondly, it iteratively couples the highest priority
remaining supply and demand. Thirdly, NORA runs a simple optimization procedure on the created
supply and demand pairs to minimize out-of-region and out-of-continent shipments. Supply coupled
to service orders is reserved, and can only be shipped away by GESA and manual planners.

<table>
<thead>
<tr>
<th>Supply Prioritization</th>
<th>Demand Prioritization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Local availability</td>
<td>1. Service order - Emergency</td>
</tr>
<tr>
<td>2. ASML factory availability</td>
<td>2. Service order - Priority</td>
</tr>
<tr>
<td>3. Local excess - Within region</td>
<td>3. Korea CW zero-bin</td>
</tr>
<tr>
<td>4. ROP - CWs</td>
<td>4. SSL zero-bin - Based on NAV</td>
</tr>
<tr>
<td>5. Local excess - Within continent</td>
<td>5. SSL - Based on NAV</td>
</tr>
<tr>
<td>6. Local excess - Worldwide</td>
<td>6. UIR zero-bin - Based on NAV</td>
</tr>
<tr>
<td>7. CW incoming - ≤ 7 days</td>
<td>7. Service order - Routine</td>
</tr>
<tr>
<td>8. SSL - CWs</td>
<td>8. Sales order - Almost due/local excess</td>
</tr>
<tr>
<td>9. SSL/ROP - LWs</td>
<td>9. ROP - Based on NAV</td>
</tr>
<tr>
<td>10. CW incoming - ≥ 8 days</td>
<td>10. Sales order - Remaining</td>
</tr>
</tbody>
</table>

Table 3: NORA’s supply and demand prioritization criteria (non-exhaustive)

The supply and demand prioritizations are summarized in Table 3. Supply prioritization focuses on
sourcing from nearby locations whenever possible, as they are faster, cheaper and more reliable to
source from. Demand prioritization focuses on fulfilling the most urgent demand first, with priority
given to replenishing SSL stock. Ties for service and sales order prioritization are broken on UND.
Prioritization for replenishments is based on continental, regional and local NAV risk. The NAV
risk formula approximates the expected difference in NAVs given the current inventory level and
NAVs if all inventory were filled up to ROP, whereby the expected number of NAVs is calculated
using the Poisson distribution. Let \( I \) be the continental, regional or local inventory level, let \( S \) be
its ROP, let \( \lambda \) be its 14-day demand rate forecast and let \( m \) be a scaling constant of choice. The
NAV risk can then be calculated as follows:
\[
\hat{E}[\text{NAV}] = \left( \frac{\sum_{j=0}^{I} \frac{1}{j!} \lambda^j}{\sum_{j=0}^{S} \frac{1}{j!} \lambda^j} - \frac{\sum_{j=0}^{S} \lambda^j}{\sum_{j=0}^{S} \frac{1}{j!} \lambda^j} \right) \lambda m.
\]

After the priority lists have been compiled and supply has been coupled to demand, a first optimization loop starts running from highest-priority couple to lowest-priority couple. For each couple, it starts a subloop that iterates over all of the remaining couples. If the inner and outer loop couples can be broken and remade such that an out-of-region shipment can be prevented, the optimization loop breaks and remakes the couples. Once this first optimization loop has finished, a second loop is started that follows exactly the same procedure to prevent out-of-continent shipments.

### 1.4 Opportunities for Improvement

The introduction of NORA and its consequent further development has proven to be extremely valuable to ASML. Nonetheless, considerable opportunities for improvement are still present. Three improvement directions are therefore discussed here, namely the alignment between operational and tactical planning, the heuristic that is used to make the decisions, and the types of decisions that are considered.

#### 1.4.1 Operational and Tactical Alignment

The NORA supply and demand matching heuristic relies heavily on ROPs and SSLs, which are determined tactically under a set of simplifications and assumptions that are implicitly or explicitly used to maintain tractability. Kranenburg and Van Houtum (2009) do not only use an approximation approach, but also assume the use of basestock policies and FCFS replenishment on an operational level. Van Aspert (2015) does not only optimize greedily, but also neglect the possibility of proactive lateral transshipments. Lamghari-Idrissi (2021) does not only approximate the premium contract overflow demand, but also assume that premium and non-premium stock are split instead of using a critical level policy. Finally, the ROPs and SSLs are kept static over longer periods of time, even though input variables that led to them might have changed considerably.

These assumptions are reasonable and sometimes even necessary when planning over longer periods of time, as computational complexity would otherwise lead to difficulties with tractability (Topan et al., 2020). In operational planning, however, these assumptions and simplifications can be problematic. For example, Abouee-Mehrizi et al. (2014) find an average optimality gap of 25% when the FCFS material allocation policy is used for their two-echelon distribution system. Deviating from the ROPs and SSLs might therefore be necessary to obtain near-optimal policies.

#### 1.4.2 Decision Heuristic

NORA’s supply and demand matching heuristic, like any heuristic, is provably suboptimal under specific circumstances. The prioritization lists look at only one criterion at a time, and disregard waiting time and delivery response time metrics. CPA and LPA contracts cannot be met in the most efficient way possible as a result. It might happen that NORA has to decide between replenishing two warehouses with almost equal impact on NAV risk, but a very different impact on waiting time risk. For example, when one location is much more isolated than the other. The logical choice would then be to base the decision on waiting time risk, but NORA would only look at the NAV risk. Similar situations can arise wherever thresholds are applied, as each of these thresholds considers only one metric at a time.
The matching optimization heuristic can likewise lead to optimality gaps. It implicitly assumes that all service orders actually lead to demands, even though this assumption is frequently violated in practice. It also attempts to avoid unnecessary transportation costs by preventing out-of-region and out-of-continent shipments, but ignores the fact that within-region and within-continent shipment times and costs are not all equal. Moreover, the optimization loops are not well-equipped for minimizing out-of-region and out-of-continent shipments. By only optimizing on continent in the second loop, in-region shipment opportunities might be missed. In conclusion, significant improvement opportunities are present when it comes to matching optimizations.

1.4.3 Decision Space

A final improvement direction can be found in NORA’s decision space. NORA currently fulfills demand proactively by fulfilling service orders, but does not anticipate for future demand. Multiple investigations have, however, shown the value of proactive lateral transshipments. In her thesis on ASML’s service network, Pielage (2018) found that the use of proactive lateral transshipments could lead to cost reductions of approximately 7%. A study by Topan and van der Heijden (2020) led to similarly encouraging results, which suggest that considerable downtime reductions can be realized through proactive lateral transshipments, especially when tactically-set basestock levels are relatively low. Considering these results, it is likely that extending NORA’s decision space to include proactive lateral transshipments can lead to significant performance improvements.
2 Problem Definition

The aim of this chapter is to establish the thesis objective, provide requirements for the deliverables and to define the scope of the project. It starts by describing the objective and requirements in Section 2.1, after which it delineates the scope in Section 2.2 and elaborates on our approach to meeting the objective and requirements approach in Section 2.3.

2.1 Objective and Requirements

The objective of this thesis is to suggest a new method to improve NORA’s material allocation and lateral transshipment decisions, such that it can be implemented and applied in practice. This method should improve upon the current method in the trade-off between costs and service levels. It should be cost-efficient, computationally efficient and able to meet material availability commitments. In short, it should be demonstrably implementable and maintainable, leading to the following requirements:

- **Computational tractability**: The running time required for the proposed method should be such that the method can be run as part of the daily NORA cycle. It should remain tractable in case the number of SKUs and locations increases significantly, as might be the case in the near future.
- **Input availability**: The proposed method should be designed such that all information it needs to function is actually available within ASML.
- **Expandability**: The proposed method should be fit for further functional expansions, might the need arise at ASML. Possible future expansions such as local sourcing and running in near real-time should be possible, even if it requires significant alterations to the method.
- **Maintainability**: The implementation of the proposed method should be in line with maintainability principles. Its architecture should be modular to facilitate updates. Furthermore, used structures, functions and terminology should align with established standards in the field to increase clarity and ease interactions with external libraries.

2.2 Scope

To keep the thesis manageable, its bounds were delineated at the start of the thesis. The scope is decided upon based on a series of meetings with all ASML supervisors, all university supervisors and various other people from the Service Automation and Engineering team. During these meetings, we discussed scoping decisions in terms of the potential added value of inclusion and in terms of the amount of introduced complexity. Scoping decisions were finalized in a final scoping meeting with all ASML and university supervisors present. What follows is a point-by-point summary of our conclusions on what aspects of the problem context to include in the scope:

- **Service parts and tools**: Both service parts and service tools are important to ASML, but including both would widen the scope significantly. Both earlier external and internal research focused mainly on service parts, leading to more analytical results being available, the development of a simulation model for ASML’s parts network (Aerts, 2022) and the knowledge that an assumption of Poisson demand is reasonable for much of ASML’s parts demand (Van den Oord, 2021). The demand for both types of materials is approximately equal, and does hence not lead to a preference of one over the other. Considering the advantages of choosing service parts over service tools, the choice is made to focus on service parts.
• **Service logistics network**: The main reason improvements proposed earlier were not implemented is their lack of scalability. To prevent this problem, the full service logistics network including all of its locations and all served business lines should be taken into consideration from the start.

• **Proactive lateral transshipments**: The potential benefits of proactive lateral transshipments for unscheduled demand are considerable, as established by preceding Master theses (Pielage, 2018; Dmitrochenko, 2020). Furthermore, current processes could easily facilitate this type of transshipments as other types of lateral transshipments are already used. Proactive lateral transshipments are therefore included in the project scope.

• **Contract performance**: The purpose of the CSCM department is to enable ASML to meet its material availability commitments. Contract performance is hence a key consideration and should be accounted for in the proposed method. The exact manner it is accounted for is of secondary importance to CSCM, as long as it enables CSCM to meet ASML’s material availability commitments.

• **GESA**: The GESA system is developed by ASML’s IT department separately from NORA, and is built using an entirely different architecture. Implementing changes would therefore be time-intensive. Additionally, its main principle of fulfilling emergency shipments from the nearest location is sensible. Given these two considerations, it is chosen to leave GESA’s decision logic out of the scope. GESA does however have a strong effect on system outcomes. It should therefore be accounted for as a given.

• **Forecasting**: Including the process of forecasting in the scope is unlikely to add value, as integration possibilities are minimal. Furthermore, it would widen the scope considerably. Forecasts are hence considered as provided, although their accuracy remains a consideration.

• **Local sourcing**: Local warehouses receiving stock from a local source rather than from another warehouse is referred to as local sourcing. It is currently only used for a select number of service tools at a limited number of locations, although there are plans to widen the scope. Given both its limited scope and the further complexities it would introduce to the investigated problem, it is chosen not to include local sourcing in the project scope.

• **UIR demand**: Demand for upgrades, installs and relocations is a significant part of the total demand. Nonetheless, fulfilling UIR demand is conceptually quite similar to replenishments. UIR demand is prioritized based on NAV, and UIR demand is modelled as a temporary increase of ROPs. Given that replenishments are already included in the scope, UIR demand is not considered to be a prerequisite for the deliverable to be demonstrably implementable.

• **Sales orders**: Parts purchasing availability contracts are fulfilled through sales orders. They are generally only used for older systems, and account for less than 1% of orders handled by NORA. Considering their limited volume, they are not seen as a necessary component for a method to be demonstrably implementable.

### 2.3 Approach

The remainder of the thesis is dedicated to fulfilling our objective of proposing a demonstrably implementable and maintainable method. To this end, each of the remaining chapters is dedicated to a subobjective. Chapters 3, 4 and 5 address the *how* of the thesis in increasing level of detail, in line with the recommendations by Van Aken and Berends (2018). Particular attention is paid to the requirement of computational tractability, as previous theses were not able to meet this requirement (Pielage, 2018; Dmitrochenko, 2020). More concretely:
• **Chapter 3** reviews previous literature to select an appropriate optimization approach.

• **Chapter 4** identifies modelling requirements for the model to be demonstrably implementable, and proposes modelling techniques for the model to be computationally tractable.

• **Chapter 5** explains how the modelling requirements and techniques are translated into code.

Subsequently, Chapters 6, 7 and 8 examine whether the proposed model fulfills the thesis objective and requirements. They verify if the model is computationally tractable, ascertain if it leads to improvements compared to the current NORA, and provide insight into the inner workings and correctness of the method respectively.

• **Chapter 6** assesses the effectiveness of the scaling methods proposed in Chapter 4 by performing an ablation study.

• **Chapter 7** evaluates the performance of our method by benchmarking it against our NORA implementation and analyzing the results.

• **Chapter 8** validates our approach by applying our model on historical data, and examining its proposed actions with a supply chain engineer from the Service Automation and Engineering team.

We separate the ablation study from the benchmark study such that we have ample computing power available for the ablation study, and such that we can use real data for the benchmark study. Both are not possible at the same time, as ASML data may not leave ASML and servers available to the department wherein the thesis is written do not support usage of the required programming language. We separate the benchmark study from the manual validation such that we can use historical parameters for the former and historical scenarios for the latter.

Lastly, **Chapter 9** relates the findings and insights from Chapters 6, 7 and 8 to each other. It thereby draws conclusions on the degree to which the objective and requirements are met, in addition to elaborating on limitations of our study. Finally, it also provides recommendations both for future research and for applying the proposed model and obtained insights in practice.

Both ASML and university supervisors were involved in each phase of the thesis. Conclusions of the literature review were discussed with all supervisors. Conceptual design decisions on business logic were discussed with at least two ASML supervisors and one university supervisor. Conceptual design decisions on scaling techniques were discussed with all university supervisors and at least one ASML supervisor. Detailed design decisions were discussed with at least one ASML supervisor and one university supervisor. Decisions on the design of the ablation study were discussed with at least two university supervisors and one ASML supervisor. Decisions on the setup of the benchmark study were discussed with all supervisors. Finally, decisions on the setup of the manual validation were discussed with at least two ASML supervisors.
3 Methodology

This chapter assesses possible methods on their ability to meet the defined thesis objective and requirements. It starts by examining established methods in Section 3.1, leading to the conclusion that they fail to meet the thesis objective and requirements. Section 3.2 therefore examines the suitability of deep reinforcement learning as an alternative. Lastly, Section 3.3 weighs the advantages and disadvantages of the different methods, and argues that deep reinforcement learning is the most promising method for the investigated problem.

3.1 Existing Methods

Operational planning is closely related to tactical planning, but faces a distinct set of challenges. Topan and van der Heijden (2020) note three ways in which operational planning differs from tactical planning. (i) Decisions mainly influence short-term performance, (ii) various types of real-time information are available during the decision process and (iii) decisions depend on the state of the system at a certain moment in time rather than the steady-state. Consequently, we restrict our overview of existing methods to operational planning methods for service logistics. We examine the categories of methods identified in the literature review by Topan et al. (2020), namely the use of simple decision rules, the use of more complex analytical methods and the use of exact methods.

3.1.1 Decision Rules

Operational systems and decision problems are usually modelled using simulation, which is used to evaluate proposed decision rules (Topan et al., 2020). Several simple decision rules have for example been proposed and evaluated for stock allocation in distribution systems. Abouee-Mehrizi et al. (2014) investigate several simple decision rules for stock allocation in a two-echelon distribution system. Although they find that first-come-first-serve (FCFS) performs poorly, they identify several other decision rules that perform well, including the generalized multilevel rationing (GMR) priority rule introduced in the paper. However, what decision rule performs best is heavily dependent on the testbed. Rong et al. (2017) similarly look at distribution systems and compare their proposed allocation rule to FCFS. Experiments lead them to conclude that FCFS is better in some cases, while their allocation rule is better in others. They note that a part of the differences between experiments appears to be caused by whether the basestock optimization procedure complements the choice of allocation rule.

We observe that the simple decision rules proposed in both of these examples can lead to better-than-baseline results, but their performance tends to be sensitive to the context in which they are applied. This observation is in line with the findings of the literature review by Topan et al. (2020), who note that models are often system specific and fail to consider interventions in a more unified and holistic manner. They state that it is often unclear which interventions are complementary and which are substitutable. Without a unified and holistic approach, opportunities might be missed for complementary interventions, whereas unnecessary costs and complexities might be introduced by substitutable interventions.

3.1.2 Analytical Methods

The category of more complex analytical methods is used by Topan et al. (2020) to refer to methods such as “myopic/greedy algorithms, marginal analysis and decomposition” (p. 5). As marginal analysis and decomposition are often used as components of a solution rather than as a solution in
and by themselves, we discuss them as such. We follow this discussion by an examination of greedy algorithms using Topan and van der Heijden (2020) as a basis. Lastly, we discuss the application of stochastic programming by Dehghani et al. (2021) to optimize proactive lateral transshipment decisions.

The key idea behind decomposition is to split a problem in smaller subproblems that are easier to analyze and solve, possibly by using an approximate approach. For example, Howard et al. (2015) apply decomposition to analyze operational decisions in their two-echelon spare parts network with emergency replenishments. They approximate this network by decomposing it into easier-to-analyze single-echelon systems, after which exact solutions are derived for these systems. The key idea behind marginal analysis is to estimate the sum of the marginal benefits and costs of a decision. For instance, Gerrits et al. (2022) use marginal analysis to derive decision rules for their two-echelon spare parts network with expediting and emergency replenishments. They estimate the marginal returns of decisions using approximations for the expected backorders, which they use to formulate decision rules that stipulate that the decision with the highest estimated marginal return should be chosen. Their approach thereby reduces to a greedy algorithm. Its performance was however meagre, as it was not able to achieve significant Pareto improvements compared to a purely reactive policy.

Topan and van der Heijden (2020) propose a greedy algorithm for a two-echelon spare parts network very similar to ours, with reactive lateral transshipments, proactive lateral transshipments, reactive emergency replenishment and proactive emergency replenishment. They define their cost function as the sum of transportation costs and a conservatively yet arbitrarily set downtime penalty, allowing them to greedily optimize on item-level. Their greedy algorithm is compared to an exact mixed integer programming (MIP) formulation, which leads the authors to conclude that the performance gap is relatively large for this operational problem. Given the strong performance of greedy algorithms on tactical planning problems (e.g., Van Houtum and Kranenburg, 2015), they hypothesize that this gap is caused by the algorithm's myopic nature.

Dehghani et al. (2021) investigate the use of stochastic programming for a two-echelon blood supply network with proactive lateral transshipments and reactive emergency replenishments, which contains four locations and one SKU. The authors generate a set of scenarios using a quasi-Monte Carlo sampling method, after which they use a mixed-integer linear programming (MILP) two-stage stochastic programming model to optimize their decisions. The first stage regards the decision at hand and its immediate consequences, whereas the second stage models the problem’s future dynamics under the assumption that the \((R, S)\) policy will be used and hence that no further lateral transshipments will take place in the future. Although they only benchmark their model against the previously used \((R, S)\) policy, their results appear promising given the considerable cost improvements when compared to this simple decision rule. Even so, the authors concede that multi-stage stochastic programming would have been more suitable for the problem, but use two stage stochastic programming instead to keep it computationally tractable. They also simplify the problem by ignoring blood type substitution and crossmatching rejection rates to maintain tractability.

3.1.3 Exact Methods

Topan et al. (2020) name dynamic programming, linear programming (LP) and MIP as exact methods. Researchers that wish to investigate optimal policies often use Markov Decision Processes (MDPs). If the assumption is made that the time horizon is infinite, steady-state analysis can be
used to characterize and identify optimal policies using for example dynamic programming. Finding the optimal policy using this approach does however quickly become intractable for more complex problems due to the curse of dimensionality Topan et al. (2020). LP is generally only exact when decision variables can be fractional, which leaves MIP as the last of the three options.

Aside from their greedy algorithm, Topan and van der Heijden (2020) also propose a MIP model for their two-echelon spare parts network, with reactive lateral transshipments, proactive lateral transshipments, reactive emergency replenishments and proactive emergency replenishments. Their network consists of approximately 30 locations and 6,000 parts. Similar to Dehghani et al. (2021), they approximate the future consequences. They do so by assuming that the probability of future backorders changes by its expectation, rather than with variation. The results do however appear to be promising, with significant cost savings when compared to a NORA-like baseline on the artificial testbed they introduce. Nonetheless, their method is at the edge of what can be considered tractable as it requires half a second per SKU per period. Furthermore, the current speed is dependent on taking an item approach rather than a system approach, even though several papers have already shown that an item approach can lead to considerably higher costs (Basten and van Houtum, 2014).

3.1.4 Discussion

We discussed each category of operational planning methods discussed by Topan et al. (2020) using examples from recent literature, and found that they all either have some distinct disadvantages or are unable to form a full solution by themselves. Simple decision rules fail to consider interventions in a holistic and unified manner, in addition to their quality often being system specific (Topan et al., 2020). Decomposition and marginal analysis could be part of a solution, but do not provide a full solution by themselves. Greedy algorithms were found to perform poorly for operational problems, possibly due to their myopic nature (Topan and van der Heijden, 2020). In conclusion, simple decision rules and greedy algorithms lead to subpar performance. Decomposition and marginal analysis do not suffice as a solution on their own.

Stochastic programming and MIP yield more promising results. The performance of stochastic programming can lead to significant cost savings (Dehghani et al., 2021), but the technique struggles with computational tractability. Despite only looking at a problem with four locations and one SKU, Dehghani et al. (2021) state that they need several undesirable simplifications to keep their model computationally tractable. The use of MIP similarly appears promising at first glance. Despite considerable simplifications on future consequences of decisions and despite using an item approach however, the MIP model by Topan and van der Heijden (2020) can only barely be considered computationally tractable. 0.5 second per SKU per period would translate to approximately 1.5 hours per 10,000 SKUs per period. Meaningful future improvements are unlikely, as Koch et al. (2022) found that the past twenty years of research has failed to significantly improve the overall solvability of mathematical programming instances. Stochastic programming and MIP can thus both be concluded to struggle with tractability and require undesirable assumptions.

We hence conclude that each of the operational planning methods identified by Topan et al. (2020) has some significant disadvantages. To truly solve the problem to near-optimality then, we need to look further than these established methods to avoid their disadvantages. The following section therefore examines a technique that has the potential to achieve exactly this aim, and explores ways in which these and other potential disadvantages could be avoided.
3.2 Deep Reinforcement Learning

Many problems encountered in inventory management are sequential decision-making problems. For such problems, the value of a decision depends on subsequent decisions that will be taken in the future. This property often makes it computationally hard to quantify the value of a decision, leading researchers to make dubious assumptions on future events and decisions. We saw examples of such simplifications while discussing Topan and van der Heijden (2020) and Dehghani et al. (2021) in Section 3.1. These simplifications can be shortsighted. For example, if we follow Dehghani et al. (2021) in assuming that no proactive lateral transshipments will be possible in the future, we might want to execute them unnecessarily while it is still possible to execute them later if need be.

Deep reinforcement learning (DRL) is an approximate technique that can explicitly account for future decisions and events, which makes it a potent technique for a wide range of inventory management problems. Gijsbrechts et al. (2022) demonstrate the potential of DRL as a general-purpose technique for inventory management. They do so by successfully applying it to the lost sales problem, the dual sourcing problem and the optimization of distribution systems. This success leads them to conclude that problem-agnostic DRL models can achieve performance on par with state-of-the-art heuristics that took years to develop. Vanvuchelen et al. (2020), Oroojlooyjadid et al. (2022) and Liu et al. (2023) reach similar conclusions on different types of inventory management problems. From these successes, we can conclude that DRL could potentially be used to improve NORA’s material allocation and lateral transshipment decisions.

A second important advantage of DRL is that most of its computations are required during training instead of when deployed. If a DRL model can be trained for a problem of the desired scale, all operational decisions can hence often be taken in near real-time. DRL algorithms are, however, known to require much computational power to train and tune (Boute et al., 2022; Gijsbrechts et al., 2022). Even so, research is successful at making DRL increasingly scalable for inventory problems. Examples can be found in Van Jaarsveld (2020), De Moor et al. (2022). Van Hezewijk et al. (2022) and Vanvuchelen et al. (2023b). We therefore conclude that DRL can meet our requirement of computational tractability when deployed, but that scaling its training process can be a challenge.

Previous literature also describes various characteristics of DRL for inventory management that do not directly affect our objective and requirements. Vanvuchelen et al. (2020) apply the Proximal Policy Optimization (PPO) DRL algorithm to the joint replenishment problem, where they find that its learned policy strongly resembles the optimal policy, as visualized in Figure 3. Oroojlooyjadid et al. (2022) use DRL to address the beer distribution game, where their DRL model performs well despite the system being only partially observable. Based on these papers, we conclude that DRL has the potential to approach the optimal policy in realistic inventory problems.

![Figure 3: Optimal policy, DRL and (Q,S|T) actions (adapted from Vanvuchelen et al., 2020)](image)

---

1 Readers unfamiliar with mentioned inventory problems can find descriptions of these problems in Appendix A.
Other studies looked at the interface between humans and DRL algorithms. Gijsbrechts et al. (2022) and Van Hezewijk et al. (2022) note that DRL’s black-box nature might hamper model verification efforts and lead to reduced trust among end-users. However, multiple methods to achieve interpretability are available for DRL, as summarized in reviews by Puiutta and Veith (2020) and Heuillet et al. (2021). Some have already been successfully applied to logistical problems (Van Hezewijk et al., 2022). Liu et al. (2023) compare their DRL model’s performance on a multi-period lost sales setting to that of human planners. They find that their model can account for fluctuations in leadtimes and supplier fillrates, thereby preventing overreactions from human planners to potential shortages. From these studies, we can deduce the importance of considering the human element when implementing DRL algorithms in practice.

3.3 Conclusion

We discussed all categories of methods for operational planning defined by Topan et al. (2020). This discussion led to the conclusion that each of them has significant disadvantages. Exact methods struggle with large problem sizes, and therefore fail to meet our requirement of computational tractability. Decision rules are unlikely to meet our objective of improving NORA’s performance. Lastly, analytical methods either require too much computational power or dubious assumptions, which are likely to affect performance.

In contrast, we found that deep reinforcement learning can likely achieve our objective of improving on NORA’s performance while maintaining computational tractability. Computational tractability of model training is a concern, but one that we can overcome as we will see in later sections. It can also meet our other requirements. Input availability and maintainability depend more on the problem formulation and code implementation than on the method of choice. Expandability can in most cases simply be achieved by adapting the simulation model used for training. Considering that existing methods either fail to meet our objective or our requirements, and given that deep reinforcement learning could potentially meet them, our method of choice is deep reinforcement learning.
4 Conceptual Design

The objective of this chapter is to highlight the most important design decisions. It provides both a motivation for these decisions and a high-level overview of their implementation. Section 4.1 explains our choice of programming language, and Section 4.2 delves into our choice of deep reinforcement learning (DRL) algorithm. Decisions with major business relevance are discussed in Section 4.3. Finally, we introduce our academic contributions in Section 4.4.

4.1 Choice of Programming Language

The choice of programming language can have profound influence on both the speed and maintainability of code. Compared to higher-level languages, lower-level languages such as C++ tend to be substantially faster, but programs built in lower-level languages require more effort and expertise to maintain. The opposite holds for higher-level languages such as Java and Python. Furthermore, programming languages vary considerably in terms of available libraries. Code for lower-level languages with relevant libraries might be more maintainable than code in a higher-level language where everything is built from scratch. To select the best language for our purposes, we evaluate them on their ability to meet our objectives and requirements.

Two of our requirements are mostly independent of programming language. Input availability depends on the design of the proposed method, as almost all languages are flexible enough to model the situation in whatever manner is desired. The same holds for expandability, which depends more on the choice of method than on the choice of programming language. Both input availability and expandability are therefore left out of our considerations. With regards to computational tractability and maintainability, we find that our needs differ depending on the functionality of the code. Hence, we split our discussion in two parts, namely the simulation environment and the model pipeline around it.

The simulation environment forms the core of the DRL training procedure, and determines to a large extent whether our method is scalable enough. Computational tractability is therefore a main concern for the simulation functionality, and would suggest picking a lower-level programming language such as Rust, C# or C++. Nonetheless, maintainability remains important. C++ has an advantage over other lower-level languages here, as it allows usage of the DynaPlex (Akkerman et al., 2023) and PyTorch (Paszke et al., 2019) libraries. DynaPlex allows us to define our MDP using only a few functions, has many built-in optimizations and provides an interface to PyTorch. PyTorch allows us to train a DRL agents. Considering these advantages in terms of computational tractability and maintainability, we choose to build our simulation environment in C++.

We define our model pipeline to consist of the code to load data, pre-process data, initialize a neural network, initiate model training, initiate model evaluation and store both the model and its results. Although not unimportant, speed is a secondary concern for these functionalities. We therefore base our choice of model pipeline language on maintainability, and consequently consider higher-level programming languages such as Java and Python. Ideally, the language of choice would have a wide availability of data preprocessing libraries, have an interface to PyTorch and have good documentation for these libraries. Given that Python can be considered the standard language for data science, we believe that Python best fulfills these criteria and hence choose it as the language for our model pipeline.
4.2 Choice of Algorithm

The recent popularity of DRL has given rise to many applications of DRL algorithms to inventory management problems. Value-based algorithms such as Deep Q-Networks (DQNs, Mnih et al., 2013) were applied to inventory management problems by Oroojlooyjadid et al. (2022) and De Moor et al. (2022). This type of DRL algorithm uses a neural network to estimate the value for each possible action in each possible state, and subsequently bases its decisions on these values. However, value-based algorithms like DQNs struggle with the stochasticity inherent to many inventory management problems, as it leads to unstable value estimates.

An alternative would be to use actor-critic algorithms such as A3C (Mnih et al., 2016) and Proximal Policy Optimization (PPO, Schulman et al., 2017). This type of algorithms consists of two components. The actor is a neural network that estimates the best action probabilistically. The critic evaluates the actor’s actions, after which its evaluation is used for updating the weights of both the actor and critic neural networks. Actor-critic algorithms partially avoid the issue of unstable value estimates through probabilistic action selection, and have already been widely applied to inventory management (e.g., in Gijsbrechts et al., 2022; Madeka et al., 2022; Vanvuchelen et al., 2023b). Nonetheless, both the actor’s action probabilities and the critic’s value estimates can suffer from instability due to stochasticity.

In response to the challenge posed by stochastic logistical problems, Van Jaarsveld (2020) proposes the Deep Controlled Learning (DCL) algorithm. DCL uses two key ideas to maximize its performance on operations research problems. The first idea is to transform sequential decision-making problems into supervised learning problems using policy iteration. Samples are generated by simulating the consequences of possible actions taken now, under the assumption that future decisions are taken by another agent. This agent is a pre-specified base agent in the first policy iteration, and the agent trained in the previous iteration afterwards. Using this approach has multiple advantages. First, it avoids using unstable value approximations caused by the stochasticity inherent to many operations research problems (Van Jaarsveld, 2020). Second, sample generation can easily be parallelized (Van Jaarsveld, 2020). Third, base agents can serve as teachers to considerably speed up convergence (De Moor et al., 2022).

The second key idea behind the DCL algorithm is variance reduction, which helps an agent to efficiently estimate the quality of a given action in stochastic environments. The algorithm reduces variance through two methods, namely roll-out policies and common random numbers. Roll-out policies simulate multiple trajectories (i.e., roll-outs) from a given initial state. Having multiple roll-outs for a single decision enables averaging the reward of these roll-outs, making the reward signal much more stable. Common Random Numbers (CRNs) build on roll-out policies by ensuring that the sequence of encountered events is the same for all of the allowed decisions. As a consequence, no additional variance arises due to differences in event streams. Considering both these theoretical arguments and its strong benchmarks, we choose DCL as our DRL algorithm.

4.3 Simulation Environment

For our proposed method to be demonstrably implementable, key characteristics of our simulation environment need to match those of the real world. Simultaneously, it is impossible to account for all real world processes. The following sections motivate our main choices on what characteristics to include, and explain our implementation of them. Section 4.3.1 describes the order of events within our model. Our implementation of service orders is discussed in Section 4.3.2. Finally, Section 4.3.3 delves into the way our model handles contractual commitments.
4.3.1 Decision Moments

Motivation

The key idea behind the temporal design of our simulation environment is that decisions are taken at discrete intervals, while the environment changes continuously. Our motivation for this design choice is to match the current situation as well as possible. The NORA run takes place at discrete intervals, although demand can arrive at any moment. When demand takes the form of routine or priority orders, no new decision moments are scheduled. When demand takes the form of emergency orders, it is the GESA system that acts. Given that the decisions made by GESA are out of our scope, all decisions in scope take place at the discrete intervals of the NORA run. We can thus model the system as a discrete-time Markov Decision Process (MDP). Figure 4 provides an example of what an interval in our discrete-time MDP might look like.

Implementation

We implement the order of events in our MDP accordingly. Decision moments are assumed to take place at times \( t \in \mathbb{N}_0 \), and the MDP is assumed to go through three steps at each decision moment. First, shipments with leadtime \( L \in \mathbb{N} \) placed at time \( t - L \) arrive. Second, shipment decisions are made. Third, the status of unfulfilled service orders (SOs) might be updated under conditions that are specified in the following chapter. Several SOs can arrive between two subsequent decision moments. Routine and priority SOs are automatically added to the system backlog, whereas emergency SOs are fulfilled instantaneously using either local stock or GESA. As a result, the moment of arrival does not matter for routine and priority SOs. In contrast, the order in which emergency SOs arrive affects the availability of local stock and stock available to GESA.

4.3.2 Service Orders

Motivation

Service orders are the instantiation of demand, and hence crucial for modelling NORA’s demand fulfillment process. To model them in a convincing manner, we choose to incorporate them in as much detail as possible. To this end, we base our design on real data and business rules where possible, and fall back to the should-be situation where not.

Implementation

Concretely, we implement six types of SOs. The implemented types are routine SOs from non-DRT customers, routine SOs from DRT customers, priority SOs from non-DRT customers, priority SOs from DRT customers, emergency SOs from non-DRT customers and emergency SOs from DRT customers. The time between the moment a SO is created and its ultimate need date (UND) is
bound according to the Customer Service department’s decision rules, which stipulate a range of [4, 14] days for priority SOs and [15, 42] days for routine SOs. In our simulation, we assume these times to UND follow a uniform distribution over the stipulated ranges. Furthermore, we assume the proportion of routine, priority and emergency SOs is constant over time, locations and SKUs. Neither of these assumptions matches the exact situation in practice, but together with all three supervisors from ASML both were concluded to be necessary in absence of better data.

Stock reservations are implemented as in the current NORA logic. This logic prescribes that stock can be reserved for service orders with a UND in the coming 21 days, and that reservations are confirmed 4 days before UND. Reserved stock can only be accessed by GESA for emergency sourcing purposes, or locally for service orders with nearer UND. Confirmed stock can no longer be accessed for any reason. Neither type of stock is counted towards the SSL or re-order point (ROP). We assume that stock is consumed when fulfilled at UND, and that replenishments are triggered at UND even if the service order was not fulfilled.

4.3.3 Contract Performance

Motivation

The ability to meet contractual requirements in a cost-effective manner is a key requirement for the designed DRL model. However, meeting contractual requirements in a cost-effective manner typically involves restricting the action space or quantifying the costs of not meeting them. We choose to use a combination of both. In line with current business practice, we implement the distinction between SSL and ROP stock in our base agent, in our GESA implementation and in our restrictions on the action space. We quantify the costs of not meeting contractual requirements by estimating downtime costs.

We model unmet service orders past their UND as backordered until fulfilled, whereby the type of service order determines the costs for each hour a service order remains unmet. This approach brings three advantages. (1) It is similar to a Lagrangian relaxation approach, which is optimal under strong duality and near-optimal when using perturbations on the Lagrangian multipliers. (2) Hourly downtime costs are easy to interpret. (3) It enables decomposition of our 10,000 SKU problem into 10,000 single SKU problems, thereby enabling the application of global models.

Implementation

Contract performance is incorporated through the base agent, through restrictions on the action space and through hourly downtime costs. The base agent is made to prioritize in the same manner as NORA. The GESA implementation can only source from SSL stock when fulfilling an emergency service order with access to SSL stock. The action space is restricted to prohibit replenishment of ROP stock from SSL stock, and to prohibit fulfilment of routine service orders from the SSL stock of other locations.

Hourly downtime costs are estimated based on downtime costs incurred by customers, where one hour of machine downtime can cost up to €72,000 (ASML Holding N.V., 2014). We set the cost of the worst-case unfulfilled service orders to be equal to this amount, namely emergency service orders from DRT customers. For the remaining service order types, we halve the downtime costs for each priority type lower. This approach leads to rough estimates, but was agreed to be the best method to obtain estimates in meetings with all three ASML supervisors given the lack of better data. It results in the following overdue costs:
• Emergency service order from DRT customer: €72,000/hour.
• Emergency service order from non-DRT customer: €36,000/hour.
• Priority service order from DRT customer: €18,000/hour.
• Priority service order from non-DRT customer: €9,000/hour.
• Routine service order from DRT customer: €4,500/hour.
• Routine service order from non-DRT customer: €2,250/hour.

4.4 Scaling Deep Reinforcement Learning

Existing DRL methods do not scale well enough for inventory management problems the size of ASML’s service network (Boute et al., 2022). To industrialize a DRL approach, we therefore need to introduce new techniques that significantly increase scalability. We propose three such techniques. Global models allow us to scale over many SKUs. Approximate evaluation enables DRL to better handle stochastic and sparse problems like NORA’s. Lastly, action shaping enables our model to more efficiently explore the action space.

4.4.1 Global Models

Motivation

In order to scale a DRL method for ASML’s service network, it needs to be able to handle more than 10,000 SKUs. Considering all SKUs at the same time in the same model would be intractable, as the state and action spaces would explode. One option would be to train a model separately for each SKU, which could possibly be sped up using transfer learning. Boute et al. (2022) build on this idea by suggesting a meta-learning approach that trains DRL models for multiple SKUs in similar circumstances, by using one or multiple base models. They conjecture that the resulting reduction in state and action space could significantly improve scalability of multi-SKU DRL models. However, to the best of our knowledge this idea has not yet been implemented.

A similar idea has taken root in the forecasting community. Global models are trained on data from multiple SKUs, whereby SKU-specific features are included such that they can still yield predictions relevant to a SKU of interest. They have shown strong empirical performance over many different studies (Petropoulos et al., 2022). Conceptually, they work by learning cross-SKU patterns in both the feature and label data. Cross-learning can achieve superior forecasting results by fully utilizing available data (Makridakis et al., 2022a,b), while simultaneously reducing computational costs (Makridakis et al., 2022a).

Implementation

Considering the benefits and past performance of global models, and considering the conceptual arguments made by Boute et al. (2022), we propose creating a global DRL model. We generate the data required for our global model as follows. Each time the DCL algorithm initializes a simulation thread, it is passed a set of parameters associated to a randomly selected SKU. The thread can then start generating samples using these parameters. After 100 samples have been generated, the thread is terminated, and a new thread is starting for a new randomly selected SKU. This continuous re-initialization allows the DCL algorithm to generate data for any number of SKUs, even when the number of available threads is limited.
We encourage cross-learning by carefully engineering our features. We explicitly choose to not include SKU-identifiers in the feature space to prevent overfitting. Furthermore, we standardize all input features by expressing them using fillrates. The idea behind this choice is to help the model recognize that locations with lower fillrates are generally at risk, whereas locations with higher fillrates could be used to alleviate this risk. Using fillrates can also help give the model a sense of scale, by including features which express the fillrate in case the number of units in stock would be one unit higher or lower. The model can then recognize low-demand locations by looking for locations where the difference between fillrates is relatively large, and recognize high-demand locations by looking for smaller differences in fillrates. An exact description of the utilized features is given in Section 5.5.

4.4.2 Approximate Evaluation

Motivation

Reward shaping is the process of artificially altering or augmenting a reward function. If applied properly, it can speed up training while still allowing for convergence to the optimal solution (Ng et al., 1999). It is especially useful for problems with sparse rewards (Sutton and Barto, 2018), as it can be used to increase reward density, thereby making the training process more efficient. Proof of its potential effectiveness in inventory control is provided by De Moor et al. (2022), who use reward shaping to warm-start their DRL model training procedure.

A related concept is that of variance reduction, which aims to reduce stochasticity in reward signals. It is hence especially effective for highly stochastic problems. The algorithm by Van Jaarsveld (2020) uses two types of variance reduction. First, the author uses a roll-out policy that performs multiple different actions on the same starting state. Second, they use a common random number approach such that the sequence of events that follows is the same for all roll-outs. Van Jaarsveld (2020) prove the effectiveness of their variance reduction schemes by benchmarking their algorithm on the classical lost sales problem, where they find that it significantly outperforms benchmark heuristics and model-free DRL methods.

Service logistics problems tend to have rewards with higher sparsity and stochasticity than most inventory control problems, due to the comparatively low demand rates and high service rates. As reward shaping and variance reduction have proven to be effective remedies in earlier studies, they are promising options to make our deep reinforcement learning model more scalable. We therefore suggest implementing them as follows.

Implementation

Rather than using the realized reward $R_t$ for a given period $t$, it is possible to use an approximation of the expected reward $\hat{E}[R_t]$ for this period $t$. This approximated expectation would provide a much more consistent and dense reward than the realized reward $R_t$. The exact expectation is not used as it is too computationally expensive to compute each period. We hence propose obtaining estimates $\hat{E}[R_t]$ using Algorithm 3, for which a detailed description is given in Section 5.3.

The key idea is to calculate upper bounds on the fillrates encountered by demand arrivals, and to assume that all unmet demand overflows to the next emergency sourcing location. It is inspired by the approach of Kranenburg and Van Houtum (2009), who obtain lower bounds for the encountered fillrates by assuming that the distribution of overflow demand is Poisson with a constant rate. Our approach further differs by looking at a finite time interval without replenishments, and by considering prioritized demands with a critical level policy.
4.4.3 Action Shaping

Motivation

The size and shape of the action space can drastically impact the scalability of DRL models, as already briefly touched upon in our discussion of global models in Section 4.4.1. Previous work on DRL for inventory management has therefore already examined means to formulate the action space in a more scalable way. For example, Vanvuchelen et al. (2023b) approach the joint replenishment problem using a DRL model with continuous action representation. More specifically, they let their model output a real number, which they then map to a feasible discrete action. This change allows them to reduce the action space to a single decision per SKU, thereby decoupling the size of the problem from the size of the action space.

Van Hezewijk et al. (2022) also seek to reduce their action space, but take a different approach. They address a multi-item stochastic capacitated lot sizing problem with DRL using a feasibility and eligibility mask. This mask hides actions that lead to infeasible or low-quality solutions from the DRL model, thereby preventing the model from wasting computational resources on exploring these solutions. The DRL models by Vanvuchelen et al. (2023b) and Van Hezewijk et al. (2022) were both subjected to numerical experiments, from which the respective authors conclude that action space reduction approaches lead to more efficient model training.

Implementation

We propose shaping the action space by splitting decisions into multiple subdecisions. First, we only allow shipping one unit at a time. Second, we first select the location to ship towards, and then the location to ship from. The necessity for splitting the action space is given by its growth rate if we would not split it. If shipping more than one unit at a time is allowed, the size of the action space would grow exponentially in the number of units on stock. If the location to ship from and towards is chosen at the same time, the size of the action space would grow quadratically in the number of locations. With the current setup, it grows only linearly in the number of locations.

Neither split affects convergence properties, as all possible actions can still be reached through combinations of remaining actions. Conceptually, the first split helps simplify the learning process for our algorithm by facilitating cross-learning between actions. The state space does not increase as a result, variables such as inventories and inventory positions remain in the same range. Hence, our algorithm effectively has to learn to decide between a considerably reduced amount of actions without needing to process a more complex input. The second split simplifies the learning process for the agent by leveraging correlation in the quality of actions. For example, if we already know that shipping towards a certain location is not necessary, then we do not need to waste computational power on exploring options whereby we ship to that location from other locations.
5 Detailed Design

This chapter explains our design choices and implementation in detail. It starts by providing a high-level overview of our implementation in Section 5.1. Next, Section 5.2 discusses how our Markov Decision Process (MDP) evolves over time. Section 5.3 explains the implementation of our approximate evaluation algorithm. Our NORA-like base agent implementation is summarized in Section 5.4. Section 5.5 describes our feature engineering. Finally, Section 5.6 gives an overview of all approximations and modelling assumptions from the preceding sections. Where applicable, these sections contain references to objects and functions from the code such that a reader could use this chapter as high-level documentation.

5.1 Implementation Architecture

We implement our model using the DynaPlex library (Akkerman et al., 2023) and its modular design principles to improve maintainability. Consequently, our implementation consists only of the model pipeline and the simulation engine. The architecture of these components and the way they relate to each other is summarized in Figure 5. On the left, it visualizes the five steps of the model pipeline. Three of these steps, namely data loading & preprocessing, model tuning & training and model evaluation depend on the experiment and are hence discussed in Chapters 6, 7 and 8. MDP initialization merely consists of creating the DRL_MDP and DRL_BaseAgent objects and performing precomputations for both. Samples are collected using the simulation engine, which we discuss in the remainder of this chapter.

Figure 5: Architecture of the proposed implementation
The simulation engine consists of one main loop of five steps in addition to five functions that can be called from within this loop, whereby steps and functions that define our MDP are colored grey in the figure. The engine is built within the DynaPlex library and hence capable of running multiple simulation trajectories in parallel. Each simulation trajectory is initialized by selecting an SKU uniformly at random, after which inventories are initialized at their re-order point (ROP) levels. We re-initialize every pre-specified number of actions, such that samples can be collected for more SKUs than there are CPU threads.

Once a simulation trajectory has been initialized, the first action can be selected. The set of feasible actions is determined using an action masking function described in Section 5.2.1. Section 5.4 explains the procedure followed by our NORA-like base agent to select actions in the DCL algorithm’s first generation. The state as observed by our agent is given by our feature extraction method, for which the feature engineering is discussed in Section 5.5. Once an action has been selected, it is processed using the function outlined in Section 5.2.2. If another action can be taken, the simulation engine moves back to the action selection function.

If not, the simulation engine proceeds to the event generating function described in Section 5.2.3. The procedure to process these events is given in Section 5.2.4. Whenever an event is such that an emergency order cannot be met locally, our implementation of the GESA system is called. The function that models this system is detailed in Section 5.2.5. Lastly, Section 5.3 outlines the algorithm we use to calculate an estimate of the expected reward during the event processing function. After event processing has finished, the simulation loop returns to the action selection step for as long as the maximum sample collection time has not been exceeded.

5.2 Markov Decision Process

5.2.1 Action Masking

The DRL\textsubscript{MDP}.\texttt{IsAllowedAction(state, action)} function is used to determine whether actions are allowed or not. Prohibited actions are masked from both the training data and trained model, making it impossible for the model to choose these actions. In accordance with the action split discussed in Section 4.4.3, we distinguish two types of actions. Firstly, our model chooses what warehouse to allocate a single SKU to. Secondly, our model chooses what warehouse to source this single SKU from. When faced with either of these action types, our model can also choose to stop the cycle of (re-)allocating SKUs at a given decision moment when choosing the receiving or sending location.

We always allow our model to stop (re-)allocating SKUs. Determining whether a location can be sourced from depends on the availability of safety stock level (SSL) and non-SSL stock. Any location with non-SSL stock can be sourced from for fulfillment of SOs, replenishment of SSL stock at LWs or replenishment of SSL zero-bins at CWs. Locations with SSL stock can only be sourced from to fulfill a priority SO or to replenish SSL stock from the replenishment location. The process to determine whether a location is allowed to receive stock is shown in Figure 6. In order to improve the quality of generated samples, we prohibit selection of receiving locations with demand types for which no appropriate inventory would be available. This design decision prevents creating samples on a choice between receiving locations for which no sending location would be available. These samples would be useless, because each action would then lead to the same outcome, namely stopping the (re-)allocation cycle and not sending any further shipments.
5.2.2 Processing Actions

After the agent selects one of the allowed actions, this action is processed using the `DRL_MDP.ModifyStateWithAction(state, action)` function. The function consists of three parts, namely one for processing the decision where to ship towards, one for where to ship from and one for the abort action. If it is processing a decision on where to ship towards, it updates the `state` object to indicate that a decision on where to ship from should now be taken. If it is processing a decision to abort the decision-making process for this decision moment, it updates the `state` object to indicate that simulation should proceed to the `DRL_MDP.GetEvent(rng)` function.

The part processing a decision on where to ship from is designed under the assumptions that reserving stock for priority SOs is always better than replenishing SSL, that replenishing SSL is always better than reserving stock for routine SOs, and that reserving stock for routine SOs is always better than replenishing ROP. These assumptions are in line with current business requirements, and allow us to limit the action space to only selecting a receiving warehouse, rather than also selecting what the warehouse should use the shipment for. The pseudocode for this part of the function is given in Algorithm 1.

In accordance with the design assumptions, it consists of three cases. First, if a priority SO can be fulfilled, it is fulfilled. Second, if the receiving location’s inventory position is such that a routine SO can be fulfilled, it is fulfilled. In either of these two cases, reserved stock arriving before the confirmation date is stored such that our GESA function can access it, and costs for late fulfillment are incurred if the leadtime is longer than the time until the ultimate need date (UND). Third, if neither a priority SO nor a routine SO could be fulfilled, the shipped unit is simply entered into the receiving location’s pipeline. For all three cases, corresponding transportation costs are incurred and the shipped unit is subtracted from the sending location’s inventory.
5.2.3 Generating Events

Events are generated using the \texttt{DRL.MDP.GetEvent(rng)} function, which produces a series of common random numbers (see Section 4.2). It returns these random numbers in a demand event object, which allows re-use of the generated numbers over different rollout trajectories. The \texttt{DRL.MDP.ModifyStateWithEvent(state, demand)} function takes this object as input, and maps the random numbers to demand characteristics based on the properties of the SKU associated to the state object. Characteristics include the location of demands, the type of demands (routine, priority or emergency), their ability to access SSL stock and their UND.

5.2.4 Processing Events

The \texttt{DRL.MDP.ModifyStateWithEvent(state, demand)} function takes the demand event generated by the \texttt{DRL.MDP.GetEvent(rng)} function and updates the MDP’s state accordingly. The function starts by updating the status of SOs and their reserved stock. SOs past UND are added to the overdue SO count, and reservations with time to UND below 4 are confirmed. Where possible, SOs without reservations are also confirmed. Next, the function successively processes new emergency SOs, new priority SOs, new routine SOs and existing routine SOs crossing the 21-days reservation time threshold. Replenishments are triggered at the UND for each of these SO types, even if they are not fulfilled at this date.

Emergency SOs are processed in order of arrival, as they are fulfilled instantaneously from inventory or through GESA. Routine and priority SOs are processed in order of UND, as they are fulfilled at decision moments. They are fulfilled through on hand stock reservations or pipeline stock reservations when possible, and added to a vector of unfulfilled SOs when no on hand or pipeline inventory of the appropriate type (e.g., non-SSL) is available.
Algorithm 2 DRL.MDP.ModifyStateWithEvent(state, demand)

1: for all Locations do
2:   Overdue SOs += Amount(SOs at UND)
3:   Delete (Reservations where Time to UND < Confirmation time)
4:   If possible, confirm (SOs without reservations)
5:   Costs ← DRL.MDP.Evaluate(state)
6: for all Emergency SOs in order of arrival do
7:   Trigger replenishment
8:   if Local stock available then
9:     Fulfill locally
10: else
11:   Fulfill using DRL.MDP.GESA(state, location, isDRT)
12: for all Priority SOs in order of UND do
13:   Trigger replenishment at Time to UND
14:   if Local stock available then
15:     Reserve stock locally, confirm if possible
16:   else if Local pipeline stock available then
17:     Reserve pipeline stock locally
18:   else
19:     Create (Priority SO at Location)
20: for all Routine SOs in order of UND do
21:   if Time to UND ≤ Max. reservation time then
22:     Trigger replenishment at Time to UND
23:   if Local stock available then
24:     Reserve stock locally
25:   else if Local pipeline stock available then
26:     Reserve pipeline stock locally
27:   else
28:     Create (Routine SO at Location)
29: else
30:   Create (Future routine SO at Location)
31: for all Future routine SOs do
32:   if Time to UND ≤ Max. reservation time then
33:     Trigger replenishment at Time to UND
34:   if Local stock available then
35:     Reserve stock locally
36:   else if Local pipeline stock available then
37:     Reserve pipeline stock locally
38:   else
39:     Create (Routine SO at Location)
40: Delete (Future routine SO at Location)
41: return Costs
5.2.5 Emergency Sourcing

The GESA system is implemented through the DRL.MDP.GESA(state, location, isDRT) function, which can be called from the event-processing function described in Section 5.2.4. If the boolean isDRT is set to true, it iteratively tries to source from both the SSL and non-SSL stock of locations specified in the GESA sourcing sequence for location. If isDRT is set to false instead, the function iteratively tries to source from non-SSL stock of locations specified in location’s GESA sourcing sequence.

For the choice of location, the GESA implementation does not distinguish between reserved stock and ordinary stock, it simply follows the GESA sourcing sequence. However, within a location it will first attempt to take ordinary stock. If not possible, it will source from stock reserved for the routine service order with the furthest UND, if any such stock exists. If neither ordinary stock nor stock reserved for routine service orders is available, it will source from stock reserved for the priority service order with the furthest UND.

5.3 Evaluation Algorithm

To calculate the expected reward for a given period, the action-processing function described in Section 5.2.2 calls the DRL.MDP.Evaluate(state) function. This function outputs an estimate of the expected emergency sourcing costs for the coming period, which in our case corresponds to one day. The motivation and main idea behind the evaluation function were discussed in Section 4.4.2. What follows here is an overview of used notation, assumptions, calculations and bounds in Section 5.3.1, and a step-by-step discussion of the pseudocode in Section 5.3.2.

5.3.1 Preliminaries

Loop Invariants and Cost Estimates

We maintain a cost estimate loop invariant in the outer loop. This invariant contains an estimate of costs incurred at all locations that incur demand locally and have so far been looped over. Emergency shipments are assumed to have a fixed cost per unit, dependent on the location that is sourced from and the location that is sourced towards. Let J be the set of locations, let j ∈ J be the receiving location, and let l ∈ σj be the sending location. We then denote the emergency shipment costs of sending a unit from l to j by $c_{l,j}^m$.

Four loop invariants are maintained in the inner loop. First, we maintain an estimate of the expected overflow demands in $\hat{E}[O_1]$ and $\hat{E}[O_2]$, whereby $\hat{E}[O_1]$ refers to the non-DRT demand stream and $\hat{E}[O_2]$ to the DRT demand stream. Second, we maintain counters for the amounts of accessible stock that needs to be depleted to arrive at a zero-bin for both streams. $P_2$ contains a counter for all stock, whereas $P_1$ only counts local stock and stock above SSL.

Local Fillrate Calculations

To ensure computational tractability, we precompute as many values as possible and ensure that as many calculations as possible can be precomputed. In accordance with this principle, we precompute one-period fillrates $\beta_1(I)$ and $\beta_2(I)$ under purely local non-DRT and DRT demand respectively, whereby I denotes the on-hand inventory at the start of the given period. To prevent notational burden, we leave out indices related to location and period in this subsection.

Let k denote the SSL, let $\lambda_1$ denote the non-DRT emergency service order demand rate and let $\lambda_2$ denote the DRT emergency service order demand rate. We distinguish between two cases to
calculate the fillrates $\beta_1(I)$ and $\beta_2(I)$. When $I \leq k$ at the start of a given period, non-DRT demand arrivals will encounter a zero-bin and hence $\beta_1(I) = 0$. Following our derivations in Appendix B.2, the fillrate $\beta_2(I)$ is then given by:

$$\beta_2(I) = \frac{I}{\lambda_2} - e^{-\lambda_2} \sum_{x=0}^{I-1} \frac{(I-x)(\lambda_2)^{x-1}}{x!}$$

When $I > k$ at the start of the given period, both DRT and non-DRT demand can access stock at the start of this period. According to our derivations in Appendix B.2, the fillrates $\beta_1(I)$ and $\beta_2(I)$ are then given by:

$$\beta_1(I) = \frac{I-k}{\lambda_1 + \lambda_2} - e^{-(\lambda_1+\lambda_2)} \sum_{x=0}^{I-k-1} \frac{(I-k-x)(\lambda_1 + \lambda_2)^{x-1}}{x!}$$

$$\beta_2(I) = 1 + \left( \frac{I-k}{\lambda_1 + \lambda_2} + \frac{k}{\lambda_2} - 1 \right) \left( 1 - e^{-\lambda_2} \sum_{n=0}^{I-k-1} \frac{(\lambda_2)^n}{n!} \right)$$

Bounding Fillrates

While the derived fillrates hold for systems without lateral emergency sourcing, demand can overflow from one location to another in our system. Calculating exact fillrates for this type of system is computationally intensive. Hence, we use upper bounds on the fillrates instead. Let $j$ denote a location, and let $\hat{\beta}_{j,1}(I_j)$ and $\hat{\beta}_{j,2}(I_j)$ denote our upper bounds for non-DRT and DRT fillrates respectively. These fillrates are a function of demand, which consists of local demand and overflow demand. Fillrates computed under purely local demand therefore form an upper bound. We use these upper bounds as our estimate for the fillrates encountered by local demand. The same estimates can be used for the fillrates encountered by overflowing demand. However, the upper bound is weak when overflow demand is high compared to local demand. In contrast, the bound is tight when demand rates and overflow demand are low.

Let $l$ denote a location to which demand is overflowing, let $\hat{\beta}_{l,1}^c(I_l)$ and $\hat{\beta}_{l,2}^c(I_l)$ denote its fillrates under local and overflow demand, and let $\beta_{l,1}$ and $\beta_{l,2}$ denote its fillrates under only local demand. Notice that fillrates are the complement of the probability that previous demand has already depleted all accessible stock at the moment of a demand arrival. Bounding using fillrates under local demand is equivalent to bounding by the probability that previous local demand has already depleted all stock. We can also bound using the probability that all stock up to this point has been depleted purely by demand from location $j$, by using the fillrates $\beta_{j,1}(P_1)$ and $\beta_{j,2}(P_2)$. As these two probabilities are independent and either would lead to a zero-bin by itself, we can multiply the two bounds to obtain a stronger bound. Concretely, we have that $\hat{\beta}_{l,1}^c \leq \beta_{j,1}(P_1)\beta_{l,1}(I_l)$ and $\hat{\beta}_{l,2}^c \leq \beta_{j,2}(P_2)\beta_{l,2}(I_l)$. Using this combined bound helps avoid inaccuracies that could otherwise arise due to overflow demand.
5.3.2 Pseudocode

The pseudocode for our evaluation function $\text{DRL\_MDP.Evaluate(state)}$ is given in Algorithm 3. Before the algorithm is started, we precompute fillrates $\beta_1(I_j)$ and $\beta_2(I_j)$ under local demand for each location and for all possible inventory levels while initializing the $\text{DRL\_MDP}$ object. Let $j \in J$ be a local warehouse and $S_j$ its ROP, then the range of inventory levels over which we precompute fillrates is $I = \{0, \ldots, \sum_{j \in J} S_j\}$. The precomputed fillrates are stored in the $\text{DRL\_MDP}$ object and are hence readily available each time $\text{DRL\_MDP.Evaluate(state)}$ is called.

The $\text{DRL\_MDP.Evaluate(state)}$ function starts by initializing the cost estimate loop invariant $\hat{E}[C]$ and looping over all locations $j \in J$ in steps (1) and (2) respectively. Next, it initializes the remaining loop invariants $\hat{E}[O_1]$, $\hat{E}[O_2]$, $P_1$ and $P_2$ in steps (3)-(6). Expected overflow demand rates are estimated using the fillrates under local demand. Counter $P_2$ is initialized using the local inventory, while counter $P_1$ is initialized such that $\beta_{j,1}(P_1)$ gives the proportion of demand arrivals at $j$ that can be fulfilled locally at location $l$.

Algorithm 3 $\text{DRL\_MDP.Evaluate(state)}$

1: Initialize $\hat{E}[C] \leftarrow 0$
2: for all $j \in J$ do
3: Initialize $\hat{E}[O_1] \leftarrow (1 - \beta_{j,1}(I_j))\lambda_{j,1}$
4: Initialize $\hat{E}[O_2] \leftarrow (1 - \beta_{j,2}(I_j))\lambda_{j,2}$
5: Initialize $P_1 \leftarrow \max(I_j, k_j)$
6: Initialize $P_2 \leftarrow I_j$
7: for all $l \in \sigma_j$ do
8: Update $P_1 \leftarrow P_1 + [I_l - k_l]^+$
9: Update $P_2 \leftarrow P_1 + I_l$
10: Estimate $\hat{\beta}_{t,1} \leftarrow \beta_{j,1}(P_1)\beta_{t,1}(I_l)$
11: Estimate $\hat{\beta}_{t,2} \leftarrow \beta_{j,2}(P_2)\beta_{t,2}(I_l)$
12: Update $\hat{E}[C] \leftarrow \hat{E}[C] + \hat{E}[O_1]\hat{\beta}_{t,1}c_{t,j}^m$
13: Update $\hat{E}[C] \leftarrow \hat{E}[C] + \hat{E}[O_2]\hat{\beta}_{t,2}c_{t,j}^m$
14: Update $\hat{E}[O_1] \leftarrow (1 - \hat{\beta}_{t,1})\hat{E}[O_1]$
15: Update $\hat{E}[O_2] \leftarrow (1 - \hat{\beta}_{t,2})\hat{E}[O_2]$
16: Update $\hat{E}[C] \leftarrow \hat{E}[C] + \hat{E}[O_1]c_{0,j}^m$
17: Update $\hat{E}[C] \leftarrow \hat{E}[C] + \hat{E}[O_2]c_{0,j}^m$
18: Return $\hat{E}[C]$

The inner loop in steps (7) to (15) iterates over all emergency sourcing locations $l$ in the GESA sourcing sequence $\sigma_j$ for location $j$. It starts by adding newly passed inventory to the counters $P_1$ and $P_2$ in steps (8) and (9). Next, it estimates the fillrate encountered by overflow demand from location $j$ to $l$ in steps (10) and (11) for non-DRT and DRT demand respectively. These fillrate estimates are used for updating the cost estimate loop invariant in steps (12) and (13), after which they are used to update the expected overflow demand estimates in steps (14) and (15). All remaining overflow demand after the inner loop has finished is assumed to be sourced from Veldhoven, which we denote by location 0. The costs associated to this sourcing option are added in steps (16) and (17). Finally, the function ends by returning the cost estimate in step (18).
5.4 Base Agent

We use an adapted version of NORA as our base agent (see Section 4.2). This agent is modelled using a DRL BaseAgent object, which has two functions. The DRL BaseAgent.GetAction(state) function selects an action to take according to the base agent’s logic. The DRL BaseAgent.PreCompute() function precomputes non-availability (NAV) risks used in the DRL BaseAgent.GetAction(state) function when initializing the MDP, and is included for computational efficiency. As our action space is split following the logic discussed in Section 4.4.3, decisions by the base agent are taken accordingly. Section 5.4.1 describes how it selects the next receiving location, and Section 5.4.2 describes how it selects the next sourcing location.

5.4.1 Receiving Location

The base agent’s decision tree for selecting the receiving location is designed such that it gives a reasonable indication as to the quality of a decision, but simultaneously creates a diverse set of training samples. The latter is necessary for the trained DRL agent to perform well when it inevitably encounters out-of-sample states. To give a reasonable indication as to the quality of a decision, the base agent is made to prioritize exactly like the real NORA. To create a diverse set of training samples, shipments are only made when the replenishment CW has excess stock, unlike in the real NORA. The decision tree is visualized in Figure 7.

As the decision tree is too elaborate to discuss in detail, we instead draw attention to three points. First, because of the excess stock requirement at the replenishment hub, a sending location will always be available for whatever receiving location is chosen. Second, service orders are prioritized based on which one has the nearest UND. Third, ROP and SSL replenishments are prioritized
based on NAV risk (see Section 1.3), whereby the algorithm looks at the highest local, regional and continental NAV risk. Figure 8 describes the NAV-based prioritization rules, which are the same for both NORA and our base agent. Thresholds $\tau_i, i = 1, \ldots, 5$ are left out due to confidentiality.

Figure 8: Breaking ties for receiving location selection

Figure 9: Overview of sending location selection
5.4.2 Sending Location

The decision tree for the base agent’s decision on the sending location is detailed in Figure 9. It is the same as for the actual NORA implementation, albeit with one simplification. NORA does not allow shipping the last unit in a region, with the underlying idea that this would never be worth it. We leave the decision on whether it is worth it to our DRL agent. When the decision tree in Figure 9 leads to multiple eligible sourcing locations, ties are broken based on the region, continent and NAV risk of these locations. This tie-breaking procedure is given in Figure 10.

![Decision Tree](image)

Figure 10: Breaking ties for sending location selection

<table>
<thead>
<tr>
<th>Type</th>
<th>Subtype</th>
<th>Amount</th>
<th>Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Inv. on hand - 1</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Inv. on hand</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Rv. inv. on hand - 1</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Rv. inv. on hand</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Inv. position</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Fillrate (SSL &amp; ROP)</td>
<td>Inv. position + 1</td>
<td>$2 \times #\text{locations}$</td>
<td>Range $[0, 1]$</td>
</tr>
<tr>
<td>Service order</td>
<td>Routine, SSL</td>
<td>#locations</td>
<td>Range $[-1, 1]$</td>
</tr>
<tr>
<td>Service order</td>
<td>Routine, ROP</td>
<td>#locations</td>
<td>Range $[-1, 1]$</td>
</tr>
<tr>
<td>Service order</td>
<td>Priority, SSL</td>
<td>#locations</td>
<td>Range $[-1, 1]$</td>
</tr>
<tr>
<td>Service order</td>
<td>Priority, ROP</td>
<td>#locations</td>
<td>Range $[-1, 1]$</td>
</tr>
<tr>
<td>Transp. costs</td>
<td>Routine</td>
<td>#locations</td>
<td>Div. by 500</td>
</tr>
<tr>
<td>Transp. costs</td>
<td>Priority</td>
<td>#locations</td>
<td>Div. by 1000</td>
</tr>
<tr>
<td>Transp. leadtimes</td>
<td>Routine</td>
<td>#locations</td>
<td>Div. by 21</td>
</tr>
<tr>
<td>Transp. leadtimes</td>
<td>Priority</td>
<td>#locations</td>
<td>Div. by 14</td>
</tr>
<tr>
<td>Repl. leadtime</td>
<td>New-buys &amp; repairs</td>
<td>1</td>
<td>Div. by 500</td>
</tr>
<tr>
<td>Repl. hub</td>
<td>New-buys &amp; repairs</td>
<td>1</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Table 4: Extracted and engineered features
5.5 Feature Engineering

The DRL.MDP.ExtractFeatures(state, featureList) function takes the state of the MDP and an empty list of features as input, extracts features from the state and adds them to the list of features. It extracts four types of features, namely fillrate-based features, service order-based features, transportation-based features and replenishment-based features. An overview of these features is given in Table 4.

Fillrate-based features capture information on reserved and non-reserved inventory (on hand and position), on ROP, on SSL and on demand rates. Using fillrates allows representing information on these four types of variables using just two features each, namely one fillrate for demand with access to SSL stock and one without. Furthermore, it scales each of these features to the same [0,1] range, which allows the DRL agent to learn from the similarities between SKUs in a global model-like manner (see Section 4.4.1). For example, it can learn that low fillrates tend to make locations suitable locations to ship towards, but unsuitable to source from.

Concretely, we implement them as follows. We add features based on the inventory on hand, and the inventory on hand minus one to incorporate the consequences of sending a unit on hand away. We also add features based on the inventory position, and the inventory position plus one to incorporate the consequences of receiving one unit. No features on pipeline inventory are included, as ETAs on arrival times are considered too unreliable in practice, if available at all. Moreover, the difference between the inventory on hand and the inventory position already accounts for pipeline inventory implicitly.

Service order-based features provide information on whether locations have an unfulfilled routine or priority service order, and if yes, with what urgency it should be fulfilled based on its UND. For each type of service order, we only extract information on the unfulfilled service order with the nearest UND of at least the same importance, if any exists. Information on the amount of other service orders and their UND is redundant, as the single-unit shipment that is decided upon when selecting a single action would only fulfill this first service order. We determine the values of the service order-based features as follows. If there are no unfulfilled service orders of a certain type, we set the feature to \(-1\). If there are, we scale the time to UND to the [0,1] range, whereby 0 indicates the unfulfilled service order has the maximum time to UND and 1 indicates that there is an overdue service order.

Transportation-based features provide information on transportation costs and leadtimes towards a selected receiving location from all possible sending locations. Values are set to 0 when selecting the receiving location. These feature serve two purposes. First, they help the model distinguish between selection of receiving and sending locations. Second, they aid in recognizing suitable sending locations. If a sending location having low transportation costs and leadtimes make it a suitable candidate for one receiving location, then it is likely to also be a suitable candidate for another receiving location towards which its transportation costs and leadtimes are low.

Lastly, replenishment-based features provide information on the replenishment process. The replenishment leadtime gives a rough indication of when stock in-transit to the replenishment location will arrive, whereas the one-hot encoded replenishment hub feature indicates at which one of the two central warehouses replenishments will arrive.
5.6 Approximations and Assumptions

Several approximations and assumptions were introduced in the preceding sections. The aim of this section is to provide an overview of all of them in a single location, and to refer to the locations where their motivation and implementation were discussed in more detail. The following simplifications and approximations are used in our implementation:

- **Local Prioritization:** Within one location, reserving stock for priority SOs is always better than replenishing SSL, replenishing SSL is always better than reserving stock for routine SOs, and reserving stock for routine SOs is always better than replenishing ROP. See Section 5.2.2.

- **Base Agent:** Our NORA implementation is a simplified version of the real NORA. It differs in terms of scope (e.g., no special case for UIR demand, no distinction between LW/CW demands) and in that it is only able to take one action at a time. See Sections 1.3.3 and 5.4.

- **Approximate Evaluation:** We approximate the expected reward in a given period using our approximate evaluation algorithm. See Sections 4.4.2 and 5.3.

- **Reserved Stock:** When extracting features, we do not distinguish between different types of reserved stock to limit the number of extracted features. See Section 5.5.

- **Min. SSL:** Our functions to mask actions, process events and check if the base agent has stock to ship use the SSL of central warehouses, instead of min. SSL like in practice. This exception was left out in consultation with ASML supervisors. We expect this simplification to only have a minor impact on the results given the often limited differences between the min. SSL and the SSL.

We further use the following modelling assumptions:

- **Poisson Demand:** Demand arrivals are assumed to follow a Poisson process, with demand rate equal to the demand rate forecast. See Sections 1.2.2 and 5.2.3.

- **Basestock Policy:** Consumed materials are assumed to immediately lead to the order of a new or repaired material. Consequently, the inventory position of our system is constant. For routine and priority service orders, we assume that materials are consumed at UND. See Sections 1.2.3, 4.3.2 and 5.2.4.

- **Emergency Sourcing:** All emergency orders are assumed to be handled immediately by our GESA implementation, as they would be in practice. See Sections 2.2 and 5.2.5.

- **Shipment Arrivals:** Shipments are assumed to always arrive just before a decision moment. See Sections 4.3.1 and 5.2.4.

- **Time to UND:** The time between the creation date of a service order and its UND are assumed to be fully dependent on the type of service order. The distribution of this time to UND (in days) is modelled using the $\text{Uniform}(4,14)$ distribution for priority SOs and the $\text{Uniform}(15,42)$ distribution for routine SOs. See Sections 4.3.2 and 5.2.4.

- **Certainty of SOs:** All SOs are assumed to lead to the consumption of the requested part at UND if fulfilled, and at fulfillment if fulfilled after the UND. See Section 5.2.4.

- **Bound on Inventory Positions:** ROPs are used as an upper bound on the inventory positions. See Section 5.2.1.

- **Constant Leadtimes:** Replenishment and transportation leadtimes are assumed to be constant for each combination of SKU and shipping lane, in accordance with NORA’s current assumptions. See Section 5.2.4.
- **Constant Costs**: Transportation costs are assumed to be constant for each combination of SKU, shipping lane and shipment type. Waiting time costs are assumed to be constant for each type of service order and contract type. Incorporating variation in costs would not change mean cost outcomes due to linearity of expectation, but would make the system more stochastic. See Sections 5.2.2 and 5.2.4.

- **Local Fulfillment**: If demand can be met from local stock of the appropriate type (e.g., non-SSL), it will be met from this stock. See Section 5.2.4.

- **No Manual Interventions**: Our model dynamics implicitly assume that all actions follow a fixed set of rules, whereby no manual interventions take place. See Section 5.2.4.

- **Static Parameters**: Input parameters such as ROPs, SSLs and demand rates are assumed to be static, even though they are likely to change over time in practice. This assumption is common for inventory management problems due to unpredictability of future parameter changes.

- **Absence of Local Excess**: Due to a combination of the ‘Certainty of SOs’, ‘Bound on Inventory Positions’, ‘No Manual Interventions’ and ‘Static Parameters’ assumptions, it is impossible to have excess stock at local warehouses.
6 Ablation Study

This chapter assesses the effectiveness of our proposed scaling methods, and thereby highlights our academic contributions to scalability. To ensure reproducibility, we generate publicly shareable data according to the procedures detailed in Section 6.1 and Section 6.2 describes our experimental setup. Lastly, Section 6.3 provides our experimental results and discusses them.

6.1 Data Generation

We generate data for five distinct scenarios of varying scales, such that we can examine the effectiveness of the scaling methods proposed in Section 4.4. These scenarios are generated with a dual purpose. First, the scenarios should be close enough to the one encountered by ASML to convincingly state that if our method scales for these scenarios, it should also scale for ASML’s. Second, they should be relevant and reproducible for other researchers.

The settings for each scenario are given in Table 5. Scenarios 1 to 4 are used to show scalability benefits of our proposed methods for reward shaping and action splitting. Scenario 5 is used to show the scalability benefits of our proposed method for applying global models. For each scenario, we generate all parameters from scratch. Section 6.1.1 describes how we generate demand rates, replenishment leadtimes and ROPs. Section 6.1.2 elaborates on our procedure to generate regions, continents, transportation costs, transportation leadtimes and GESA sourcing sequences.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>#SKUs</th>
<th>#locations</th>
<th>#regions</th>
<th>#continents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>12</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>30</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>60</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>10,000</td>
<td>60</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5: Scenarios used in the ablation study

6.1.1 Planning Parameters

We start the planning parameter generation by generating demand rates per day and replenishment leadtimes in days. Let $|U|$ denote the number of SKUs and $|L|$ the number of locations. First, we use the $\text{Loguniform}(10^{-9}, 10^{-3})$ distribution to generate a vector of length $|U|$. This vector represents the SKU-dependent part of the demand rate, which is among others related to the SKU’s degradation rate. Second, we use the $\text{Loguniform}(10^{0}, 10^{3})$ distribution to generate a vector of length $|L|$. This vector represents the location-dependent part of the demand rate, which is among others related to the number of machines at a given location. We obtain demand rates for each SKU and location combination by taking the outer product of the two vectors, yielding a $|U| \times |L|$ matrix of demand rates. Replenishment leadtimes are generated using the $\text{Loguniform}(20, 500)$ distribution, whereby we round the generated values up to the nearest integer.

ROPs are generated by applying the greedy algorithm described in Kranenburg and Van Houtum (2009) on each SKU separately, such that we can assess our methods’ ability to deal with different service levels while only requiring a single testbed. We disregard SSLs and non-emergency service orders to ensure our experiments are not only reproducible, but also relevant for others. The greedy algorithm takes the previously generated demand rates and replenishment leadtimes as input. Furthermore, it takes as input holding costs, emergency sourcing leadtimes, waiting time
constraints and emergency sourcing lists. Holding costs are assumed to be equal for all locations. Emergency sourcing leadtimes are set at 25% of (routine) transportation leadtimes. Sourcing from factory is assumed to cost an additional 48 hours. Waiting time constraints (in hours) are generated using the $\text{Uniform}(0.25, 4)$ distribution.

6.1.2 Geographical Parameters

To generate geographical parameters, we first split the world into 3 continents and 6 regions. This split is in line with geographical splits present in many organisations, which contain divisions for the Americas, Europe-Middle-East-Africa (EMEA) and Asia-Pacific. We create our data such that each continent contains 2 regions, and such that each region contains an equal number of locations. The $x$- and $y$-coordinates of locations are generated uniformly at random over the area of a region. They serve as the basis for the transportation leadtimes, transportation costs and GESA sourcing sequences. One location from the upper middle region is randomly selected to serve as the central warehouse. A visualization of locations, regions and continents is given in Figure 11.

![Figure 11: Plot of locations (points), regions (brightness) and continents (colors)](image)

GESA sourcing sequences are determined such that a location always sources from the nearest location with available stock, if any such location exists. For the remaining parameters, we first calculate the Euclidian distances between all locations. Transportation leadtimes are obtained by multiplying distances by $\frac{14}{\sqrt{3^2+2^2}}$ and rounding them up to the nearest integer, thereby scaling the leadtimes to the $[1, 14]$ days range. Transportation costs are obtained by multiplying the distances with a factor 100 for routine shipments and 300 for emergency shipments.

6.2 Experimental Setup

All ablation study experiments are performed on a single thin compute node of the Snellius supercomputer, which has 256 GiB of RAM and 2 AMD Rome 2.6 GHz CPUs with a total of 128 CPU cores. Training and evaluation runs are performed on independently generated data, and hyperparameter tuning is performed manually. Tuned hyperparameters can be found in Appendix D. The trained DRL agent is benchmarked against our benchmark NORA implementation, which can be found in Appendix E.

Training time is regulated by a timeout on sample collection, which is set to the minimum amount of time our DRL model needs to match NORA’s performance. Evaluation runs are performed as a continuous run with a warm-up length of 10,000 days to reach the steady-state, periods of length 1,000 days for them to be independent of each other and a total of 100 periods to obtain tight confidence bounds. This set-up leads to a total simulation length of 110,000 days per SKU the model is evaluated on. 95% confidence bounds are calculated using the following formula:

$$40$$
Mean (± 95%-CI half-width) := \( \hat{\mu} \left( \pm t_{99,0.975} \frac{s}{\sqrt{100}} \right) \)

where \( \hat{\mu} \) is the average daily cost in one of the 1,000-day periods, \( t_{99,0.975} \) is the 0.975th quantile of a Student t-distribution with 99 degrees of freedom, and \( s \) is the sample standard deviation of the average daily cost in a 1,000-day period.

6.3 Results

6.3.1 Experiment 1: Scaling over locations

We start our experiments by evaluating the effectiveness of our proposed methods for reward shaping and action splitting on scenarios 1, 2, 3 and 4. To this end, four different algorithms are evaluated. The ‘DCL’ algorithm is a standard implementation of the DCL algorithm (Van Jaarsveld, 2020) without reward shaping and without splitting receiving and sending location selection actions. The ‘DCL-RS’ model extends this implementation through reward shaping, and ‘DCL-AS’ does so using action splitting. Finally, we refer to our proposed model with both reward shaping and action splitting as the ‘DCL-RS-AS’ algorithm. Experimental results are given in Table 6.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Algorithm</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DCL</td>
<td>0.40</td>
<td>25.52</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DCL-RS</td>
<td>0.27</td>
<td>22.27</td>
<td>14.48</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DCL-AS</td>
<td>0.82</td>
<td>19.84</td>
<td>9.38</td>
<td>109.92</td>
</tr>
<tr>
<td></td>
<td>DCL-RS-AS</td>
<td>0.27</td>
<td>19.36</td>
<td>5.91</td>
<td>106.79</td>
</tr>
</tbody>
</table>

Table 6: Average costs per day for single-SKU scenarios

The results show that reward shaping and action splitting significantly improve scalability. Comparing DCL to DCL-AS and DCL-RS to DCL-RS-AS suggests that action splitting leads to considerable performance improvements, although an outlier is present for DCL-AS on scenario 1. Similarly, comparing DCL to DCL-RS and DCL-AS to DCL-RS-AS suggests that reward shaping leads to consistent performance improvements, although to a lesser extent. Lastly, we note that DCL-RS-AS does not perform significantly better or worse than NORA for any scenario. It outperforms NORA by an insignificant margin for scenarios 2, 3 and 4, and achieves exactly the same result for scenario 1. Based on these results, we conclude that reward shaping and action splitting enable scaling over up to 60 locations, whereby both reward shaping and action splitting contribute significantly to this scalability.

6.3.2 Experiment 2: Scaling over SKUs

Our second experiment evaluates the effectiveness of our proposed application of global models on scenario 5. We train two DCL-RS-AS models for this dataset, and evaluate them on 100 randomly selected SKUs spread out over 60 locations. One model is trained and evaluated on the 77.91% of SKUs with the lowest (\( \leq 0.005/\text{location/day} \)) demand rates, and one model on the 22.09% of SKUs with the highest (\( > 0.005/\text{location/day} \)) demand rates, as we find that splitting the dataset in this manner leads to better results for fast movers than when using a single model. Results are given in Table 7 and Figure 12.
<table>
<thead>
<tr>
<th>Statistic</th>
<th>Slow Movers</th>
<th>Fast Movers</th>
</tr>
</thead>
<tbody>
<tr>
<td>% DRL sign. better</td>
<td>49%</td>
<td>42%</td>
</tr>
<tr>
<td>% neither sign. better</td>
<td>48%</td>
<td>37%</td>
</tr>
<tr>
<td>% NORA sign. better</td>
<td>3%</td>
<td>21%</td>
</tr>
<tr>
<td>DRL/NORA total cost ratio</td>
<td>97.76%</td>
<td>94.98%</td>
</tr>
<tr>
<td>DRL/NORA avg. cost ratio</td>
<td>91.46%</td>
<td>98.53%</td>
</tr>
</tbody>
</table>

Table 7: Performance of global models compared to the benchmark

![Histograms of cost ratios for the 100 randomly selected SKUs](image)

Figure 12: Histograms of cost ratios for the 100 randomly selected SKUs

Table 7 provides a selection of descriptive statistics, and Figure 12 provides an overview of how performance differs from SKU to SKU. The statistics are calculated as follows. The percentage of SKUs for which DRL performs significantly better than NORA and vice versa are determined using a one-sided independent samples t-test with $\alpha = 0.05$. Total cost ratios are calculated by summing DRL algorithm costs for all 100 SKUs, and dividing them by the sum of NORA algorithm costs. Finally, the average cost ratio is determined by first dividing the DRL costs by NORA costs on a per SKU basis, after which the average of this statistic is calculated over all SKUs.

The results in Table 7 and Figure 12 demonstrate that our application of global models allows the DRL model to outperform NORA on an arbitrary number of SKUs and 60 locations. DRL performs significantly better for both slow and fast movers, leading to average cost reductions of 2.24% and 5.02% respectively. Furthermore, the number of SKUs for which DRL significantly outperforms NORA is considerably higher than the number of SKUs for which DRL is significantly outperformed by NORA for both slow and fast movers. We hence conclude that our DRL model is able to outperform NORA on a dataset with 60 locations and 10,000 SKUs, and thereby meets the requirement of computational tractability.
7 Benchmark Study

The purpose of this chapter is to examine the effectiveness of our method in a real-world business context. It achieves this purpose by benchmarking our method against our NORA implementation on a test set from practice, and analyzing what factors influence its performance. Section 7.1 provides an overview of this test set. Section 7.2 describes our experimental setup. Experimental results and analysis thereof are given in Section 7.3. Finally, Section 7.4 interprets the results and discusses their implications.

7.1 Input Files and Scope

Data is collected from various sources, with a dual aim. First, the data should be as close to the data encountered by NORA as possible. Second, irrelevant, missing and low-quality data should be avoided where possible. We fulfill this dual aim as follows. Data is sourced directly from NORA and SAP to ensure applicability. However, many locations and parts are not relevant. For example, including office locations with just a few SKUs in scope would not add much value. We therefore only consider locations and parts when they are in scope both for NORA and for SPartAn to ensure relevance. Table 8 provides a summary of the used input files. Our preprocessing steps, which are aimed at ensuring data completeness and quality, are discussed in Appendix C.1.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Source</th>
<th>Day/Period</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPORT_RBA04.xlsx</td>
<td>SAP</td>
<td>Q2, 2023</td>
<td>GESA-lists</td>
</tr>
<tr>
<td>LOCALPLANTS.txt</td>
<td>SPartAn</td>
<td>Q2, 2023</td>
<td>Scope locations</td>
</tr>
<tr>
<td>*forecast_parts.csv</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>Demand rate forecasts</td>
</tr>
<tr>
<td>*material_cwh_masterdata.csv</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>Replenishment leadtimes</td>
</tr>
<tr>
<td>*plant.csv</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>ROPs and SSLs</td>
</tr>
<tr>
<td>*sloc.csv</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>ROPs</td>
</tr>
<tr>
<td>OF_Report_v4.csv</td>
<td>Spotfire</td>
<td>June, 2023</td>
<td>SO types</td>
</tr>
<tr>
<td>Settings.xlsx</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>Regions and continents</td>
</tr>
<tr>
<td>STIPT_PART.txt</td>
<td>SPartAn</td>
<td>Q2, 2023</td>
<td>Scope parts</td>
</tr>
<tr>
<td>transport_params.csv</td>
<td>Celonis</td>
<td>November 1st, 2021</td>
<td>Transportation costs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to June 30th, 2023</td>
<td>and leadtimes</td>
</tr>
</tbody>
</table>

Table 8: Benchmark study input files, whereby ‘mod_mm_’ is abbreviated to ‘*’

7.2 Experimental Setup

Benchmark study experiments are performed on a laptop with 32 GB of RAM and a 3.00 GHz Intel Core i7 vPro. Training and evaluation runs are performed on independently generated data, whereby parameters from the previous section are used. Hyperparameter tuning was performed with 400 iterations of Optuna (Akiba et al., 2019). Trial settings and tuned hyperparameters are given in Appendix D. The trained DRL agent is benchmarked against our NORA implementation, which can be found in Appendix E. Only one DRL agent is trained for all SKUs, as we find that splitting slow and fast movers does not improve performance on the benchmark study dataset.

Given the limited amount of computational power available, we alter Dynaplex’ target labelling procedure while collecting training samples for the benchmark study. We notice that this labelling procedure is affected by the confirmation window of 4 days, and by the first-come-first-serve order in which service orders (SOs) are fulfilled on a local level. Dynaplex commonly finds that delaying
the shipment moment would give the best result. Consequently, it concludes that the best action would be to stop sending shipments for up to 45% of collected samples. We overcome this data imbalance by relabelling samples where the benchmark agent would have continued to act. In such cases, we label the second-best action as the correct action. This relabelling risks leading to suboptimal timing of shipments, but we find it to be necessary given our limited sample size.

As for the ablation study, evaluation runs are performed as a continuous run with a warm-up length of 10,000 days, periods of length 1,000 days and a total of 100 periods, leading to a simulation length of 110,000 days per SKU the model is evaluated on. 95% confidence bounds are calculated as described in Section 6.2. The sample collection timeout is set to 32 hours.

7.3 Results and Analyses

To assess the effectiveness of our method on real-world data, we structure our results and analyses as follows. We present the benchmark results in Section 7.3.1. Section 7.3.2 examines cost components and shipment types to get an intuition on the source of cost differences between NORA and our DRL model. Section 7.3.3 delves into differences in service performance metrics between the two models. Section 7.3.4 seeks to identify when DRL performs well, and when it does not. Finally, Section 7.3.5 investigates several outliers.

7.3.1 Benchmark Results

We evaluate our DRL model on 100 randomly selected SKUs, whereby we do not distinguish between slow and fast movers. The results can be found in Table 9 and Figure 13. SKUs where the NORA implementation performed less than 100 shipments during the simulation period are not included in these 100 randomly selected SKUs, as we consider the results for these SKUs to be too stochastic and therefore too unreliable for our analyses.\(^2\)

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>% DRL sign. better</td>
<td>48%</td>
</tr>
<tr>
<td>% neither sign. better</td>
<td>46%</td>
</tr>
<tr>
<td>% NORA sign. better</td>
<td>6%</td>
</tr>
<tr>
<td>DRL/NORA total cost ratio</td>
<td>94.86%</td>
</tr>
<tr>
<td>DRL/NORA avg. cost ratio</td>
<td>95.22%</td>
</tr>
</tbody>
</table>

Table 9: Benchmark statistics

Statistics can be interpreted as for the second experiment in Section 6.3. Statistical significance is determined using one-sided independent samples t-tests with \(\alpha = 0.05\). The total cost ratio is defined as the DRL/NORA cost ratio of all SKUs together. Lastly, the average cost ratio is defined as the mean of DRL/NORA cost ratios for each SKU separately.

\(^2\)If these SKUs would have been included, the DRL/NORA total cost ratio would have been 95.02% and the DRL/NORA average cost ratio would have been 92.05%.
Table 9 shows that implementation of our DRL model would lead to estimated total cost savings of 5.14%. There does not seem to be much difference between performance on slow and fast movers, given the proximity of the total cost ratio to the average cost ratio. Furthermore, the t-tests indicate a clear preference for DRL over NORA. Figure 13 shows that most cost ratios are in the 85% to 105% range, thereby confirming this preference. We therefore conclude that our DRL model is able to outperform the NORA implementation on real data.

7.3.2 Analysis 1: Cost Components and Shipment Types

In the following analysis, we examine how our DRL model is able to outperform NORA on real data. More specifically, we investigate the type of actions taken by each algorithm and look at the effects on the source of incurred costs. We classify the type of actions taken by each algorithm into one of three groups. Replenishment actions are defined as (1) shipments from the replenishment location to a location without service orders (SOs), or (2) shipments from the replenishment location to a location without priority SOs, but with SSL replenishment demand. Reactive actions are defined as (1) shipments to a location with a priority SO, or (2) shipments to a location with a routine SO and no SSL demand. Finally, proactive actions are defined as all remaining actions.

Overall, our DRL model performs 7.76% more shipments than NORA. Figure 14 plots the total number of shipments for each type, whereby amounts are standardized (stdzd.) by dividing by the total number of NORA shipments. It indicates that our model performs a small number of proactive shipments, although they do not lead to a decrease in reactive shipments. Interestingly, it also shows that the number of replenishments is higher for DRL than for NORA. A likely explanation for these observations is that stock is proactively balanced on a regional or continental level, such that demand can be met reactively from nearby locations to reduce waiting times and save costs. As a consequence, stock sent from the replenishment location would be categorized as a replenishment rather than a reactive shipment, thereby explaining the increase in replenishments.

As overdue costs make up the majority of incurred costs, we examine the consequences of the actions suggested by DRL on the overdue costs for each type of SO. Figure 15 presents the overdue costs for each type of SO, whereby all amounts are standardized (stdzd.) by dividing by NORA’s total amount of overdue costs. The figure shows that DRL is able to reduce overdue costs over all SO types. Unsurprisingly, most costs are related to emergency shipments. However, the plot
also indicates that routine SOs are, under our cost assumptions (c.f. Section 4.3), responsible for a large proportion of costs. This observation is surprising considering the lower volume and costs when compared to priority SOs, but can be explained by longer waiting times due to routine SOs’ inability to source from other locations’ SSL stock.

From our analysis on the shipment types and overdue costs, we draw two conclusions. First, the use of proactive lateral transshipments by DRL is a likely source of cost savings when compared to NORA through regional and continental rebalancing. Second, cost savings are present for all three types of service orders. This insight indicates that the ability to obtain cost savings is robust to changes in cost input parameters.

### 7.3.3 Analysis 2: Contractual Fulfillment

Contract performance is a key consideration, as discussed in Section 2.2. We therefore investigate differences between our DRL model and NORA in terms of three service measures on a per location basis, namely fill percentage, zero-bin percentage and waiting times. We define fill percentage as the mean inventory on hand at a location, divided by the location’s ROP. We define zero-bin percentage as the percentage of time spent in a zero-bin state. Finally, we assess the mean waiting time over all SO types for each location, in addition to zooming in on the mean waiting times for routine SOs, priority SOs and emergency SOs at each location.

Figure 16 presents boxplots for estimated changes in these service performance metrics over all locations in our scope. A positive percentage indicates an increase relative to NORA, whereas a negative percentage indicates a decrease relative to NORA. Values for fill percentages and zero-bin percentages are given in terms of percentage points, whereas the percentages given for the waiting times are relative to the NORA value.

![Figure 16: Boxplot of differences in DRL/NORA service performance metrics](image)

Analyzing Figure 16 leads us to one main insight, namely that application of DRL only has a minor impact on contractual fulfillment. The bounds of the plot, from −3% to 6%, are telling. Variation between routine SOs, priority SOs and emergency SOs is to be expected, as the fulfillment process of routine SOs can be much more stochastic because of their more limited sourcing opportunities and lower prioritization. Nonetheless, the minor decrease in service performance at some of the locations indicates room for improvement at locations with an already higher risk of not meeting service levels, although the observed decrease may also be a consequence of low sample sizes at low demand locations. Overall, we conclude that our method of incorporating service performance is sufficient for fulfilling contracts, although opportunities for improvement exist.
7.3.4 Analysis 3: Predictors of Performance

The benchmark results showed that our DRL model outperforms NORA for most SKUs, but is outperformed by NORA for a minority of them. This third analysis seeks to identify when our method performs well, and when it does not. To this end, we examine the correlation between potential predictors and the DRL/NORA cost ratio for the 100 randomly selected SKUs. We use Kendall’s tau as our measure of correlation, given the non-linear nature of many potential predictors. Results are provided in Table 10. Negative coefficients indicate that higher values for the corresponding variable tend to co-appear (‘concord’) with lower cost ratios, and therefore with better DRL model performance. The opposite holds for positive coefficients.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replenishment leadtime</td>
<td>0.0389</td>
<td>0.5693</td>
</tr>
<tr>
<td>Replenish in Korea</td>
<td>0.3688 **</td>
<td>0.0000</td>
</tr>
<tr>
<td>Sum of demand rates</td>
<td>-0.0249</td>
<td>0.7141</td>
</tr>
<tr>
<td>Sum of ROPs</td>
<td>0.0065</td>
<td>0.924</td>
</tr>
<tr>
<td>Proportion DRT demand</td>
<td>0.0828</td>
<td>0.2468</td>
</tr>
<tr>
<td>Proportion SSL basestock</td>
<td>0.0474</td>
<td>0.4890</td>
</tr>
<tr>
<td>System-wide stockout probability</td>
<td>0.0220</td>
<td>0.7455</td>
</tr>
<tr>
<td>Nr. locations with non-zero demand</td>
<td>-0.1456 *</td>
<td>0.0354</td>
</tr>
<tr>
<td>Nr. locations with non-zero ROP</td>
<td>-0.0379</td>
<td>0.5831</td>
</tr>
</tbody>
</table>

Table 10: Kendall rank correlations for DRL/NORA cost ratios

The results show that receiving replenishments at the Korean central warehouse (CW) rather than at the Netherlands’ central warehouse is significantly negatively correlated to DRL model performance. Figure 17 visualizes the difference in cost ratios for the two possible replenishment locations using a boxplot. Data imbalance is a likely reason for this negative relation, as replenishments are received in Korea for only a minority of SKUs in our data.

![Figure 17: Boxplot of cost ratios for both replenishment locations](image)

Table 10 also shows that the number of locations with non-zero demand rates is significantly positively correlated to DRL model performance. A possible explanation can be found in DRL’s (in)ability to differentiate itself from NORA, which affects the distance it can create to the relatively poor cost ratio of 1.00. Having less locations with demand reduces the amount of choice given to the DRL model, thereby reducing the amount of choices where it could distinguish itself from NORA.
In conclusion, we find two SKU-specific characteristics that are significantly correlated with performance. Receiving replenishments at the Korean CW is negatively related to performance, indicating an opportunity for further improvement. The number of locations with non-zero demand is instead positively related to performance. We hypothesize that this positive relation is a consequence of DRL’s inherent (in)ability to differentiate itself from NORA for certain SKUs, and therefore does not provide an opportunity for further improvement.

### 7.3.5 Analysis 4: Outliers

In this last analysis, we examine the three SKUs with the outlying cost ratios of 60.46%, 64.31% and 137.09% to verify their validity. We respectively refer to them as A, B and C. To understand the reason for which the cost ratios might have outlying values, we start by summarizing shipment type-related statistics in Table 11. The table shows that the DRL model has an atypically high number of proactive shipments for outliers A and B when compared to the total number of shipments and the number of reactive shipments, which can explain their low cost ratios if these proactive shipment decisions are of high quality. Earlier work by e.g., Topan and van der Heijden (2020) already highlighted the potential value of proactive shipments. In contrast, the DRL model does not appear to see opportunities for proactive shipments for outlier C. This lack of proactive shipments can explain the DRL model’s inability to outperform NORA, but does not explain why the DRL model is outperformed by NORA.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Outlier A</th>
<th>Outlier B</th>
<th>Outlier C</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DRL proactives/#DRL reactives</td>
<td>55.21%</td>
<td>168.75%</td>
<td>0.00%</td>
<td>1.06%</td>
</tr>
<tr>
<td>#DRL proactives/#DRL shipments</td>
<td>19.47%</td>
<td>2.76%</td>
<td>0.00%</td>
<td>0.27%</td>
</tr>
<tr>
<td>#DRL reactives/#NORA reactives</td>
<td>99.35%</td>
<td>68.38%</td>
<td>100.50%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Table 11: Shipment type statistics for SKUs with outlying cost ratios

To better understand what might make SKUs A, B and C special, we investigate each in depth. Outlier A is replenished in the Netherlands and has a well above average number of locations with non-zero demand rates. Furthermore, it is a fast mover with relatively low ROP levels, leading to a well above average system-wide stockout probability. The replenishment location and number of demand locations are in line with what would be expected based on the third analysis, thereby explaining why DRL can outperform NORA. The well above average system-wide stockout probability can explain the margin by which it outperforms NORA, as it leads to an increased number of situations where DRL can distinguish itself from NORA. Moreover, we note that outliers of this size are in line with earlier research. Pielage (2018) reported savings opportunities of up to 37% when compared to NORA, Dmitrochenko (2020) found cost reductions of up to 22% and Van Meetelen (2020) claimed savings of up to 41% for specific parameter settings.

Outlier B is a fast mover which is replenished in Korea. It has a below average number of locations with non-zero demand rates and a below average probability of system-wide stockout. None of these aggregate statistics suggest an outlying value, but closer examination of the SSLs and demand rates reveals the likely cause. Most demands take place in two regions which we refer to as (1) and (2). The demand rate in region (1) is 14 times higher than in region (2), but the SSL of region (2) is 50% higher. As a result, NORA will prioritize replenishments in region (2) when system-wide inventory is low, even though replenishing region (1) would be better under our waiting time-based cost parameters. Moreover, we again note that outliers of this size are in line with earlier research.
Outlier C is replenished in Korea and has an average number of locations with non-zero demand rates. Most of its characteristics are close to that of a typical SKU. However, it only has two locations with non-zero ROP. Most stock at the replenishment location falls within SSL bounds, while the other location only has ROP. Furthermore, the average time to shortage at this other location is approximately 40 years when inventory is filled up to ROP. From a location perspective, it may therefore make sense not to fully fill this location. However, routine SOs may only be fulfilled from non-SSL stock. Given that NORA costs and DRL costs are near-equal for priority SOs and emergency SOs, the outlying value can be explained by DRL not always fully replenishing the other location, resulting in a decreased ability of fulfilling routine SOs.

We conclude that there is no reason to doubt the validity of the outliers’ results. Sensible explanations can be found for each of the outliers. Furthermore, outliers of this size are not unreasonable when compared to earlier research.

7.4 Conclusion

The results and analyses show that our DRL model is able to outperform NORA. The benchmark results find cost savings of 5.14% compared to NORA, even though they were obtained using only 32 hours of sample collection on a local computer. Furthermore, they show that DRL performs significantly better than NORA for 48% of SKUs, while being significantly outperformed by NORA for 6% of SKUs. Our first analysis shows that proactive lateral transshipments likely serve the purpose of regional and continental rebalancing, as suggested by the reduced waiting time costs. Moreover, it finds that the cost reductions are consistent over all three types of service orders, thereby indicating robustness to changes in cost parameters.

Our second analysis examines differences in service performance metrics of NORA and the DRL model for all the different locations. More specifically, it compares the average inventory/ROP ratio, the time spent in zero-bin state, the average waiting time, the average waiting time for routine SOs, the average waiting time for priority SOs and the average waiting time for emergency SOs. It finds that differences between NORA and DRL are only minor, leading us to conclude that our method of incorporating contract performance is sufficient for meeting material availability commitments, although there is still room for improvement.

The third and fourth analysis respectively investigate predictors of performance and SKUs with outlying performance. The third analysis finds that DRL performs best for SKUs where replenishments arrive in the Netherlands’ CW, and for SKUs that have non-zero demand rates at relatively many locations. We believe that the first of these predictors provides an opportunity for improvement, as it is likely the effect of a fixable imbalance in training data. The fourth analysis finds that a sensible explanation can be found for each of them, and that they are not surprising when considering the magnitude of outliers found in earlier research on NORA.

Overall, we conclude that our method is able to meet all three components of our objective. Benchmark results confirm that our DRL model is cost-efficient, as it achieves significant cost savings when compared to NORA. The model was trained in 32 hours, thereby confirming its computational efficiency. Finally, our second analysis demonstrates that our model is able to meet material availability commitments.
8 Manual Validation

In this chapter we validate our proposed model by applying it to historical scenarios and examining the outcomes with a supply chain engineer from the Service Automation and Engineering (SAE) team. Section 8.1 explains how data for these historical scenarios is obtained. The experimental set-up is described in Section 8.2. Section 8.3 presents our findings. Lastly, Section 8.4 draws conclusions based on these results.

8.1 Input Files and Scope

The manual validation is performed on data from July 31st, 2023, with the same models as trained for the benchmark study. Data that is required for calculating the model’s input features is hence also collected for this date. All other data is the same as for the benchmark study. Table 12 provides an overview of the used input files. The preprocessing of data shared with the benchmark study is described in Appendix C.1, and preprocessing of data specific to the manual validation is summarized in Appendix C.2.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Source</th>
<th>Day/Period</th>
<th>Values/Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPORT_RBA04.xlsx</td>
<td>SAP</td>
<td>Q2, 2023</td>
<td>GESA-lists</td>
</tr>
<tr>
<td>LOCALPLANTS.txt</td>
<td>SPartAn</td>
<td>Q2, 2023</td>
<td>Scope locations</td>
</tr>
<tr>
<td>*forecast_parts.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>Demand rate forecasts</td>
</tr>
<tr>
<td>*material_cwh_masterdata.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>Replenishment leadtimes</td>
</tr>
<tr>
<td>*nora_sales_orders.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>Outstanding SOs</td>
</tr>
<tr>
<td>*stock_integrated.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>Sales orders</td>
</tr>
<tr>
<td>*zmatconf.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>UIR events</td>
</tr>
<tr>
<td>OF_Report_v4.csv</td>
<td>Spotfire</td>
<td>June, 2023</td>
<td>SO types</td>
</tr>
<tr>
<td>replenishment_data.csv</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>Inventories, ROPs and SSLs</td>
</tr>
<tr>
<td>Settings.xlsx</td>
<td>NORA</td>
<td>June 30th, 2023</td>
<td>Regions and continents</td>
</tr>
<tr>
<td>STIPT_PART.txt</td>
<td>SPartAn</td>
<td>Q2, 2023</td>
<td>Scope parts</td>
</tr>
<tr>
<td>transport_params.csv</td>
<td>Celonis</td>
<td>November 1st, 2021</td>
<td>Transportation costs to June 30th, 2023 and leadtimes</td>
</tr>
<tr>
<td>DAL Worklist NORA.xlsx</td>
<td>NORA</td>
<td>July 31st, 2023</td>
<td>NORA decisions</td>
</tr>
</tbody>
</table>

Table 12: Manual validation input files, whereby ‘mod.mm’ is abbreviated to ‘*’

8.2 Experimental Setup

During the manual validation, we compare the actions proposed by DRL to those proposed by our NORA implementation and to those proposed by the real NORA. To this end, we apply the deep reinforcement learning (DRL) model from the benchmark study and our NORA implementation as given in Appendix E to the input data described in the previous section. The actions proposed by the real NORA are retrieved from historical worklists.

Actions proposed by DRL and our NORA implementation are recorded in a .xlsx file containing five columns. Each row corresponds to one proposed shipment. The part identifier is given in the Part column. The From column contains the location that the DRL model sources from. The FromType column indicates the type of stock that is sourced from, which can be Excess, ROP, SSL or LastSSL. The To column contains the location that the DRL model sends material
towards. Finally, the ToType column indicates the type of demand that is fulfilled, which can be PrioritySO, ReplenishZeroBinSSL, ReplenishSSL, RoutineSO or ReplenishROP. Records on actions proposed by the real NORA are kept as-is.

8.3 Case Studies and Findings

We perform manual validation through in-depth case studies together with a supply chain engineer from the SAE team. The goal of these case studies is to get qualitative rather than quantitative insight into the strengths, weaknesses and validity of the DRL model. We study actions for 20 SKUs, which we select by looking at the SKUs where the actions proposed by the different algorithms differ the most. As we cannot discuss all 20 case studies in-depth, we categorize them in one of three groups. Section 8.3.1 examines cases with differences in strategy between our DRL model and NORA. Section 8.3.2 looks at cases with differences in allocation prioritization. Finally, Section 8.3.3 discusses cases where differences in decisions are caused by differences in scope.

8.3.1 Case Group 1: Emergent Strategies

We observe several patterns in the actions proposed by DRL, which together appear to constitute a strategy. More specifically, we notice the following patterns:

- The DRL model regularly uses pro-active shipments.
- The DRL model has a tendency to fulfill demand from nearby locations.
- The DRL model sometimes uses an expediting strategy.

To illustrate these patterns, we discuss the case given in Table 13 and visualized in Figure 18. The table’s columns contain the algorithm that proposes the shipment, the sourcing location, the type of stock that is sourced from, the receiving location, the type of demand fulfilled by the shipment and the proposed shipping quantity respectively. Location names are encoded due to confidentiality. The first part of the encoding refers to the location’s region, whereas the second part can be used to identify the location itself. Central warehouses are instead encoded with the ‘CW’ prefix.

The table and figure display each of the observed patterns. Proactive shipments are performed by the DRL model from location A-1 to D-1, from C-4 to C-1 and from C-4 to C-3. The DRL model fulfills priority service orders (SOs) at location D-1 from locations B-1, B-3 and E-1, which are all closer to D-1 than CW-1 is. Finally, the DRL model immediately replaces the stock sent from location B-2 to D-1 by an incoming shipment from CW-1, which shows that nearby stock at B-2 is used to expedite fulfillment of a priority service order at D-1 from stock at CW-1.

Figure 18: Comparison of DRL model actions (left) and NORA model actions (right)
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>From</th>
<th>Supply Type</th>
<th>To</th>
<th>Demand Type</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>B-2</td>
<td>ROP</td>
<td>5</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>ROP</td>
<td>3</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>SSL</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-2</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>A-1</td>
<td>ROP</td>
<td>D-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>B-1</td>
<td>ROP</td>
<td>D-1</td>
<td>Priority SO</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>B-2</td>
<td>ROP</td>
<td>D-1</td>
<td>SSL</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>B-3</td>
<td>ROP</td>
<td>D-1</td>
<td>Priority SO</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>E-1</td>
<td>ROP</td>
<td>D-1</td>
<td>Priority SO</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>G-1</td>
<td>ROP</td>
<td>C-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>G-1</td>
<td>ROP</td>
<td>F-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>A-2</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>Priority SO</td>
<td>3</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>SSL</td>
<td>2</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>CW-2</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>C-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>F-1</td>
<td>Zero-bin SSL</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 13: Case displaying the DRL model’s emergent strategies

We also examine the differences in outcomes. Compared to NORA, DRL sends four more units to B-2, three to D-1 and one to D-2. However, its stock at location C-4 will be two lower and its stock at locations A-1, A-2, B-1, B-3, E-1 and CW-2 will be one lower. We can find logical explanations for each of them. CW-2 has a demand rate of 0 and region A has high inventories compared to its demand rates. Furthermore, B-2 has by far the highest demand rate within region B, and B-1 and B-3 have a higher percentage of their ROP filled than B-2. Finally, region D has by far the highest probability of a regional stockout and region E has one of the lowest. Overall, we conclude that these differences in strategy are reasonable and can be used to argue that application of DRL leads to improved decision-making.

Nonetheless, the actions proposed by DRL also have some weaknesses. Locations B-1 and B-3 are left with relatively little stock. Even though location B-2 could benefit from additional stock, sending four units to this specific location within the region might be too much. Moreover, sourcing from region E would leave it without any stock. An additional potential pitfall could be encountered when trying to implement the strategies in practice. Although they are a logical outcome of our assumptions and input parameters, proactive shipments might cause local resistance at the sourcing location, and use of an expediting strategy might be seen as leading to unnecessary shipments.

8.3.2 Case Group 2: Differences in Allocation

We also observe cases with notable differences in actions due to differences in allocation prioritization, rather than strategy. One of the cases demonstrating these differences is given as an example in Table 14, whose columns and location names can be interpreted as before. The table shows that the DRL model and NORA do not always prioritize the same replenishments. Both algorithms choose to replenish locations C-1, D-1, E-1 and F-1. However, the DRL model chooses to replenish locations A-1 and B-1, whereas NORA chooses to replenish locations E-2 and G-1.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>From</th>
<th>Supply Type</th>
<th>To</th>
<th>Demand Type</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>A-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>B-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>C-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>E-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>DRL</td>
<td>CW-1</td>
<td>ROP</td>
<td>F-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>C-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>D-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>E-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>E-2</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>F-1</td>
<td>ROP</td>
<td>1</td>
</tr>
<tr>
<td>NORA</td>
<td>CW-1</td>
<td>ROP</td>
<td>G-1</td>
<td>ROP</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 14: Case displaying differences in DRL and NORA stock allocation

We find that arguments can be made for the choices of both algorithms. Locations A-1, B-1, E-2 and G-1 all have zero inventory on hand and in-transit. Arguments that can be made for the choices of DRL include the following. Location A-1 has a higher demand rate than E-2 and location B-1 has a higher demand rate than G-1. Furthermore, location E-2 is in a region with multiple units already in stock, in addition to being close to the remaining stock at CW-1. In contrast, the following arguments can be made for the choices of NORA. Location G-1 is the only location within its region for the examined part, but does not have any stock. Moreover, region E has a relatively high risk of regional stockout, although this risk does not account for the nearby stock at CW-1. We conclude that the choices of both algorithms are reasonable, and that qualitative analysis of differences in allocation prioritization does not provide a reason to prefer one above the other.

### 8.3.3 Case Group 3: Incomparable Parts

Actions cannot be compared for all parts. In some cases, the real NORA has access to more information than is available to the DRL model and to the NORA benchmark implementation. Although these cases can be expected given our scope, they highlight the need for scope extensions before implementation. Concretely, we find deviations as a result of the following reasons:

- Sales orders.
- Materials in consignment.
- Materials marked for rework.
- Materials marked for scrapping.
- Service orders for predecessor materials.
- Service orders outside of regular time intervals.
- Differences between min. SSL and SSL.

### 8.4 Conclusion

We manually validate our DRL model in collaboration with a supply chain engineer from the SAE team by studying its proposed actions for 20 SKUs on historical data. Our case studies lead to two main observations. (1) The DRL model appears to follow a strategy that prefers sourcing from
nearby locations, potentially proactively. (2) The DRL model prioritizes demand locations in a different way than NORA, but arguments can usually be made for either. Overall, we conclude that the actions proposed by the DRL model appear to be reasonable, both in terms of overall strategy and local prioritization.

Nonetheless, our manual validation faces two limitations. We only validate our approach for 20 of the most affected SKUs due to the time-intensive nature of case studies, but these 20 are only a fraction of the total number of affected SKUs. Algorithm aversion (Dietvorst et al., 2015) can already occur if our model displays weaknesses in any of the remaining SKUs, even if it outperforms NORA in aggregate. More extensive validation is therefore necessary to ensure stakeholder trust is not damaged. Furthermore, our third group of cases highlight the need for scope extensions before implementation becomes possible.
9 Conclusions and Recommendations

9.1 Summary and Conclusions

The goal of this thesis is to propose a method for improving material allocation and transshipment decisions taken by the NORA algorithm, which automates most operational decisions in ASML’s customer supply chain. To this end, we compare different methods from previous scientific literature and find deep reinforcement learning (DRL) to be the most suitable candidate. Outperforming NORA requires applying a relatively sophisticated method, but this method needs to be able to take decisions in near real-time. DRL is able to meet these often conflicting objectives by training models before letting them take decisions. Although training itself is computationally intensive, trained models can take decisions in near real-time and can be re-used as often as necessary.

Research on DRL for inventory control has, however, thus far been unable to train models for problems of the required scale, as illustrated in Table 15. We improve upon the scientific state-of-the-art in three main ways. First, we propose a variance reduction method that approximates the expected reward for a given period. Second, we present a novel way of formulating the action space. Third, we show that global models can be used to train one model for many SKUs. Together, these techniques allow our DRL model to outperform NORA on a problem with 60 locations and 10,000 SKUs, as we demonstrate in our ablation study.

<table>
<thead>
<tr>
<th>Study</th>
<th>Type</th>
<th>Problem</th>
<th>#SKUs</th>
<th>#locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dmitrochenko (2020)</td>
<td>Master thesis</td>
<td>Fully connected network</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Gijsbrechts et al. (2022)</td>
<td>Journal article</td>
<td>Distribution system</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Van Hezewijk et al. (2022)</td>
<td>Journal article</td>
<td>Capacitated lot sizing</td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>Vanvuchelen et al. (2023a)</td>
<td>Journal article</td>
<td>Fully connected network</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Vanvuchelen et al. (2023b)</td>
<td>Preprint</td>
<td>Joint replenishment</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>This study</td>
<td>Master thesis</td>
<td>Fully connected network</td>
<td>10,000</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 15: Scale comparison with earlier studies using DRL for inventory management

We benchmark our algorithm against NORA on real data for 100 randomly chosen SKUs and find that it can lead to cost savings of 5.14%, whereby we consider both waiting time costs for overdue service orders and transportation costs. Moreover, we find that DRL significantly outperforms NORA for 48% of SKUs, while only being significantly outperformed by NORA for 6% of SKUs. Several analyses are performed to gain insight into the performance of DRL and differences between the two algorithms. The main findings from these analyses are as follows:

- The DRL model makes use of proactive lateral transshipments, but these transshipments do not reduce the number of reactive shipments. Instead, reductions in waiting time costs suggest that they reduce regional and continental stock imbalances.
- Cost savings are achieved over all types of service orders, and can therefore be considered robust against changes in cost parameters for these service orders.
- Location-based differences in service performance metrics between DRL and NORA are only minor, suggesting DRL does not consistently favor some locations over others. DRL hence appears sufficiently equipped to meet material availability commitments.
- DRL performs best when replenishments arrive at the Netherlands’ central warehouse rather than the Korean central warehouse, and when there are relatively many locations with demand.
Finally, we manually validate our DRL model together with a supply chain engineer from the Service Automation and Engineering (SAE) team by running it on historical data for one day. Actions suggested by DRL and NORA are compared and investigated through in-depth case studies for 20 SKUs, leading to two main observations. (1) DRL appears to follow a strategy that emphasizes sourcing from nearby locations, potentially proactively. (2) The order in which demand locations are prioritized can differ between DRL and NORA, but arguments can usually be made for either order. Overall, we conclude that the strategy and demand prioritization proposed by the DRL model appear to be sensible, both in terms of overarching strategy and local prioritization. Nonetheless, scope extensions and additional case studies are necessary before our model can be implemented.

In conclusion, we propose a method that is demonstrably cost-efficient, computationally efficient and able to meet material availability commitments. We have demonstrated its computational tractability and its ability to take decisions on real data. Moreover, the flexibility of deep reinforcement learning enables further scope expansions, while our modular design and use of external libraries support maintainability. We hence conclude that the thesis objective and requirements have been met.

### 9.2 Limitations and Business Recommendations

Considering our encouraging results, we recommend implementation of DRL to ASML. This implementation would not just be a one-time step forward in terms of cost savings, but it would also set the stage for a wide range of potential future improvements. Its flexibility can facilitate extensions such as local sourcing, dynamically adapting to contract statuses and optimization of service tool sharing. Moreover, it can be used to eliminate both implicit and explicit assumptions present in NORA, such as certainty of service order (SO) demand, static demand rates and Poisson distributed demand at smaller locations.

Our DRL model does, however, have two limitations that need to be addressed before it can be implemented. First, the scope needs to be extended to include sales orders and demand for upgrades, installs and relocations (UIR). Furthermore, it should include cases such as consignment, rework, scrapping, predecessor materials and SOs outside of regular time intervals. Second, several assumptions are made due to unavailability of data, such as assumptions on overdue fulfillment costs and assumptions on the time between SO creation and ultimate need date. These assumptions need to be investigated to ensure they do not negatively affect results.

Considering the time and effort required to address these limitations, we recommend a gradual move towards DRL implementation. Such an approach would allow ASML to reap short-term benefits, while simultaneously maximizing long-term returns. Concretely, we recommend the following short-term measures to address limitations and build the required competency:

- **Digital Twin:** Absence of a quantitative feedback loop makes it difficult to evaluate NORA’s decision rules. Building a digital twin for the operational planning could create such a feedback loop, thereby facilitating future improvements. Furthermore, it could serve as a simulation model to train a DRL model on in the long-term.

- **Process Understanding:** We believe a better understanding of the demand fulfillment process could bring new opportunities for improvement to light. Possible avenues for research include estimating costs for overdue SOs and obtaining insights into when routine SOs get upgraded to priority or emergency SOs, and when priority SOs get upgraded to emergency SOs. Such research would also help in making the digital twin as realistic as possible, while simultaneously reducing the amount of required DRL model assumptions.
- **Machine Learning Competency**: Data science and machine learning techniques are becoming increasingly important. Developing knowledge on such techniques as a side competency would enable the SAE team to perform basic analyses of their own processes, while facilitating communication with the Service Analytics and Robotics team when more advanced applications are desired. Moreover, it would create a knowledge base for DRL implementation.

If these short-term measures are followed, implementation of a DRL model is only a relatively small step. However, it may be interesting to incorporate further improvements and extensions to the DRL model during or after implementation. We propose the following next steps:

- **Contract Performance**: Contracts are currently incorporated through re-order points (ROPs), safety stock levels (SSLs) and in terms of costs assigned to overdue service orders. The benchmark study found that this approach is likely to be sufficient to meet contracts, but still leaves room for improvement. ROPs and SSLs restrict the action space, possibly leading to suboptimal actions. Moreover, this approach does not allow taking the current contract status into account. Investigating alternative methods to incorporate contractual commitments may therefore be worthwhile.

- **Optional Scope Extensions**: Our scope excluded several optional yet consequential factors, including service tools and GESA’s decision-making process. Each of these could potentially lead to significant additional savings and is hence worth investigating.

- **Maximizing Performance**: The estimated cost savings should not be seen as the maximum cost savings. Increasing available computational power and improved data engineering could lead to significant improvements. Our benchmark study model was trained in 32 hours on a local computer, which is low by industry standards. Furthermore, the discrepancy between SKUs for which replenishments arrive in Korea rather than the Netherlands suggests that performance can be improved by addressing data imbalance, for example by training one model for SKUs that arrive in the Netherlands and one for those that arrive in Korea.

### 9.3 Future Research

The summary and conclusions establish that DRL can be industrialized for ASML’s service network. Our business recommendations address modelling weaknesses, such as our limited scope and our need for assumptions. This last section looks further ahead, and discusses potential avenues for future research on contract performance, model explainability and model maintainability.

The previous section discussed the potential value of using a different method to incorporate contractual commitments. One possible option would be to use an approach akin to Lagrangian relaxation, whereby researchers seek to identify location-based penalty costs that lead to contract fulfillment under minimal costs. Lagrangian relaxation has already been established as an optimization technique for spare parts problems, see for example Basten and van Houtum (2014) and Van Houtum and Kranenburg (2015). Moreover, this approach could enable dynamic adaptation to contract status if the penalty costs are estimated as a function of contract status. DRL can then prioritize locations at risk of not meeting contracts, while deprioritizing locations with safer contract statuses.

Interpretability and explainability are preconditions for building stakeholder trust and for creating a feedback loop for future improvements. As DRL is a black-box method, measures are necessary to ensure interpretability and explainability. Much literature is already available on general methods to interpret DRL and explain its actions, as summarized in literature reviews by Puiutta and
Veith (2020) and Heuillet et al. (2021). The same does, however, not hold for applications of DRL to inventory management problems. We hence recommend research into interpretability and explainability methods specific to DRL applied to inventory management. Possible directions include research on methods to monitor concept drift, identify exceptions and identify the need for manual interventions.

Each of these methods can be grouped into the wider category of governance. Operationalizing DRL models does not only require an initial implementation, but also continuous monitoring and maintenance. Our study only focuses on building a model for a static process following a fixed set of rules, and does not provide a framework for monitoring and maintenance. We are not aware of any other studies on DRL for inventory control that addresses topics such as governance structures, handling concept drift, handling exceptions and facilitating manual interventions either, even though multiple studies have been published on application of DRL to real-world inventory problems (e.g., Madeka et al., 2022; Liu et al., 2023). We therefore encourage case studies to identify frequently faced challenges and establish best practices for the monitoring and maintenance of DRL models for inventory management.
Bibliography


Vanvuchelen, N., De Moor, B. J., and Boute, R. N. (2023b). The use of continuous action representations to scale deep reinforcement learning for inventory control. *Available at SSRN 4253600*.


Appendix A  Abbreviations and Terminology

**Basestock level:** Specifies the order-up-to level at a location. When the sum of local and pipeline inventory for a location is below this level at a decision moment, the decision is taken to order a quantity that brings it back up to this level.

**Beer game:** Classical decentralized supply chain management problem, also known as the beer distribution game. It is a cooperative problem that features a serial supply chain, whereby decisions at each location in the sequence are taken by an independent agent with only limited information on the state of the overall system. The goal of each agent is to minimize the system-wide cost, which is given by the sum of leftover inventory costs at each location plus the stockout costs at the last location in the chain.

**CPA:** Continental Parts Availability. Type of contract where demand is met from more distant local warehouses, and service commitments can (but don’t necessarily need to) be specified in terms of material availability and waiting time. See Section 1.1.

**Critical level:** Specifies an access threshold for non-DRT demand. When a location’s inventory level is above the critical level, all demand can be met from that location. Otherwise, only DRT demand can be met from that location. Also known as holdback level.

**CRNs:** Common Random Numbers. Method for variance reduction in (deep) reinforcement learning. See Section 4.2.

**CSCM:** Customer Supply Chain Management. Department responsible for ensuring field material availability on an operational level. See Section 1.2.

**CSD:** Customer Service Degree. Service performance metric equivalent to aggregate fill rate. See Section 1.1.

**CW:** Central Warehouse. Serves local warehouses. See Section 1.2.

**DCL:** Deep Controlled Learning. Type of deep reinforcement learning algorithm developed specifically for operations management problems. See Section 4.2.

**Distribution system:** Classical centralized supply chain management problem with a diverging supply chain structure. Contains few distribution centers upstream and many demand-incurring locations downstream. The goal is typically to minimize a combination of holding and stockout costs by optimizing replenishment decisions, given full control of and information on the network.

**DRL:** Deep Reinforcement Learning. See Section 3.2.

**DRT:** Delivery Response Time. Concept aimed at reducing extreme long downs (XLDs). See Section 1.1.

**DTWP:** DownTime Waiting for Parts. Service performance metric related to aggregate mean waiting time. See Section 1.1.

**Dual sourcing:** Classical supply chain management problem. Involves a single location with two supply options. One supply option has a lower leadtime but higher purchasing cost, the other has a higher leadtime but lower purchasing cost. The goal is typically to minimize the sum of purchasing, stockout and holding costs.

**DUV:** Deep UltraViolet. One of the main business lines of ASML. See Section 1.1.

**DWM:** Down Waiting for Material. Service performance metric related to aggregate mean waiting time. See Section 1.1.
**EUV**: Extreme UltraViolet. One of the main business lines of ASML. See Section 1.1.

**Fab**: Semiconductor manufacturing plant. See Section 1.2.

**FCFS**: First Come First Serve. Implies that locations that request replenishment first will be served first. One of the assumptions of the SPartAN algorithm. See Section 1.2.

**GESA**: Global Emergency Sourcing Automation. Algorithm that automates activities related to emergency sourcing. Is managed by IT in collaboration with the SAE team. See Section 1.3 for context on GESA and Section 5.2 for our implementation.

**GPA**: Global Parts Availability. Type of contract where demand is met from global warehouses, and service commitments are specified in terms of material availability. See Section 1.1.

**HERO**: Holistic Event Reservation Overview. Platform providing overview of scheduled events, including the materials needed for executing them. Is managed by the SAE team. See Section 1.3.

**Holdback level**: Specifies an access threshold for non-DRT demand. When a location’s inventory level is above the holdback level, all demand can be met from that location. Otherwise, only DRT demand can be met from that location. Also known as critical level.

**IID**: Independent and Identically Distributed. Refers to the assumption that a set of stochastic processes do not affect each other (i.e., they are independent), and their processes can be fully characterized using exactly the same parameters (i.e., they are identically distributed). See Section 1.2.3.

**Installed base**: Set of all installed machines. See Section 1.2.

**Inventory position**: Sum of a location’s on-hand and pipeline inventories.

**Lateral transshipment**: Shipment between two LWs. See Section 1.2.

**Leadtime**: Time between the moment a decision is made to order or ship a material to a certain location, and the moment the material arrives at this location.

**Lost sales**: Classical supply chain management problem. Involves a single location where all unmet demand is lost. The goal is typically to minimize the sum of stockout and holding costs.

**LPA**: Local Parts Availability. Type of contract where demand is met from a nearby local warehouses, and service commitments can (but don’t necessarily need to) be specified in terms of material availability, waiting time and delivery response time. See Section 1.1.

**LW**: Local Warehouse. Regional warehouse that serves one or more customer locations. See Section 1.2.

**MDP**: Markov Decision Process. Special type of process that evolves both stochastically and as an effect of taken decisions. Often used to model and formalize the environment of a (deep) reinforcement learning agent. See Section 4.3.1.

**NAV**: Non-AVailability. Refers to when demand is incurred but no material is on stock. See Section 1.1.

**NAV risk**: Risk of Non-AVailabilities. Refers to an approximation of the difference between the expected number of NAVs given the current inventory and the expected number of NAVs if the inventory would be filled up to ROP. See Section 1.3.

**NORA**: Network Oriented Replenishment Automation. Algorithm that automates a large part of the operational processes and decisions within the CSCM department. Is managed by the SAE team. See Section 1.3 for context on NORA and Section 5.4 for our implementation.
**Part**: Material that goes into a machine during an event. See Section 1.2.

**Pipeline inventory**: Inventory in transit to a location.

**ROP**: Re-Order Point. Output of the SPartAn algorithm during tactical planning, possibly enriched by planners. See Section 1.2.

**SAE**: Service Automation and Engineering. Team responsible for managing HERO, GESA and NORA. Part of the Planning & Delivery department. See Section 1.3.

**SKU**: Stock Keeping Unit. Type of part or tool.

**SLA**: Service Level Agreement. Defines the service performance level ASML commits to delivering. See Section 1.1.

**SO**: Service Order. See Section 1.3.

**SSL**: Safety Stock Level. Output of the SPartAn algorithm during tactical planning, possibly enriched by planners. See Section 1.2.

**Tool**: Material that does not go into a machine during an event. See Section 1.2.

**UIR**: Upgrades, Installs and Relocations. Also referred to as scheduled events. See Section 1.3.

**UND**: Ultimate Need Date. Latest date at which service order should be fulfilled. See Section 1.3.

**XLD**: eXtreme Long Down. Occurs when the waiting time for parts is exceptionally long. The exact amount of time is region-dependent. See Section 1.1.

**Zero-bin**: Situation where a location has zero inventory on hand.
### Appendix B Approximate Evaluation

#### B.1 Procedure by Kranenburg and Van Houtum

This appendix describes the approximate evaluation procedure by Kranenburg and Van Houtum (2009) in more detail. The procedure’s input parameters are as follows. Let $J$ denote the set of all local warehouses (LWs) and $K$ the set of all main LWs. Assume Poisson distributed demand and let $M_{i,j}$ and $\hat{M}_{i,j}$ respectively denote the local and total demand rate for SKU $i \in I$ at LW $j \in J$. Further assume that demand that cannot be met by an LW is fulfilled through an emergency shipment at a central warehouse. Let $k_j = k$ if $k$ is the main LW for regular LW $j$ and $K(\tilde{k}, k)$ denote the set of all LWs from what LW $\tilde{k}$ would try to source before trying LW $k$. Lastly, let $L(s, \rho)$ denote the proportion of lost sales under basestock level $s$ and utilization rate $\rho$:

$$L(s, \rho) := \frac{\rho^s/s!}{\sum_{x=0}^{\infty} \rho^x/x!}$$

The intermediate and output variables are as follows. Let $\beta_{i,j}(S_i)$ be the fraction of demand for SKU $i$ at LW $j$ met locally given basestock vector $S_i$, let $\alpha_{i,j,k}$ be the proportion of demand for SKU $i$ at LW $j$ fulfilled by main LW $k$ and let $\theta_{i,j}(S_i)$ be the fraction of demand for SKU $i$ at LW $j$ fulfilled through an emergency shipment. Using the defined parameters and variables, the approximation algorithm by Kranenburg and Van Houtum (2009) can be written down as in Algorithm 4.

---

**Algorithm 4 Approximate evaluation procedure by Kranenburg and Van Houtum (2009)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\forall j \in J \setminus K : \beta_{i,j}(S_i) := 1 - L(S_{i,j}, M_{i,j}t)$</td>
</tr>
<tr>
<td>2</td>
<td>$\forall k \in K : \hat{M}<em>{i,k} := M</em>{i,k} + \sum_{j \in J</td>
</tr>
<tr>
<td>3</td>
<td>$\forall k \in K : \theta_{i,k}(S_i) := L\left(\sum_{k \in K} S_{i,k}, \sum_{k \in K} \hat{M}_{i,k}t\right)$</td>
</tr>
<tr>
<td>4</td>
<td>$\forall k \in K : \beta_{i,k}(S_i) := 1 - \hat{M}_{i,k}t$</td>
</tr>
<tr>
<td>5</td>
<td>$\forall k \in K : A_{i,k}(S_i) := 1 - (\beta_{i,k}(S_i) + \theta_{i,k}(S_i))$</td>
</tr>
<tr>
<td>6</td>
<td>for all $k \in K$ do</td>
</tr>
<tr>
<td>7</td>
<td>$\hat{M}<em>{i,k,k} := \frac{A</em>{i,k}(S_i) \hat{M}<em>{i,k}}{1 - \prod</em>{l \in K(k,k)} (1 - \beta_{i,l}(S_i))}$, $\exists l \in K \setminus {k} : S_{i,l} &gt; 0$ otherwise</td>
</tr>
<tr>
<td>8</td>
<td>$\hat{M}<em>{i,k} := \hat{M}</em>{i,k} + \sum_{k \in K, k \neq k} \hat{M}_{i,k,k}$</td>
</tr>
<tr>
<td>9</td>
<td>$\beta_{i,k}(S_i) := 1 - L(S_{i,k}, \hat{M}_{i,k}t)$</td>
</tr>
<tr>
<td>10</td>
<td>$A_{i,k}(S_i) := 1 - (\beta_{i,k}(S_i) + \theta_{i,k}(S_i))$</td>
</tr>
<tr>
<td>11</td>
<td>if $\exists k \in K :</td>
</tr>
<tr>
<td>12</td>
<td>Go back to step (7)</td>
</tr>
<tr>
<td>13</td>
<td>$\forall k \in K, \tilde{k} \in K, k \neq \tilde{k} : \alpha_{i,k,k}(S_i) := \beta_{i,k}(S_i) \frac{\hat{M}<em>{i,k,k}}{\hat{M}</em>{i,k}}$</td>
</tr>
<tr>
<td>14</td>
<td>$\forall j \in J \setminus K, k \in K : \alpha_{i,j,k}(S_i) := \begin{cases} (1 - \beta_{i,j}(S_i)) \beta_{i,k}(S_i), &amp; k = k_j \ (1 - \beta_{i,j}(S_i)) \alpha_{i,k,j,k}(S_i), &amp; k \in K, k \neq k_j \ 0, &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>15</td>
<td>$\forall j \in J \setminus K : \theta_{i,j}(S_i) := \begin{cases} (1 - \beta_{i,j}(S_i)) \theta_{i,k_j}(S_i), &amp; K = \emptyset \ (1 - \beta_{i,j}(S_i)) \theta_{i,k_j}(S_i), &amp; \text{otherwise} \end{cases}$</td>
</tr>
</tbody>
</table>
The approximate evaluation procedure consists of three components. Steps (2) and (3) calculate the fill rate and overflow demand at the local LWs. Steps (4) to (14) calculate fill rate, demand met using lateral transshipments and emergency shipments for the main LWs. Finally, steps (15) to (16) calculate the demand met using lateral transshipments and emergency shipments for the local LWs in the second part.

The procedure is approximate rather than exact due to steps (3) and (9). These steps use a one-moment approximation for the demand at the main LWs, after which they assume it is distributed according to a Poisson distribution with the fitted mean. In reality however, this demand distribution is not exact as the demand rate at a main LW depends on the stock levels at other LWs. It is therefore not constant and consequently not Poisson. Nonetheless, the authors demonstrate that their method works well. They use the evaluation procedure as the basis for a greedy algorithm and compare its performance to a lower-bound found using Dantzig-Wolfe decomposition, where they find gaps of at most 3.7%. Interestingly, this performance is likely to be insensitive to the return time distribution, as found by earlier studies on two-echelon networks (Alfredsson and Verrijdt, 1999; Vliegen and Van Houtum, 2009).
B.2 Calculation of Critical Level Fillrates

In this appendix we derive the fill rates encountered by premium (DRT) demand and non-premium (non-DRT) demand. We consider a single-location system with critical level \( k \), which incurs non-premium demand at a rate \( \lambda_1 \) and premium demand at a rate \( \lambda_2 \). Both demand processes are assumed to be Poisson processes. We examine the system over a single period, which we define as the continuous time interval \( [t, t+1) \), and assume that no replenishments take place in this period.

Calculating the non-premium fillrate \( \beta_1 \) is relatively straightforward. Let \( D \sim \text{Poisson}(\lambda_1 + \lambda_2) \) denote the total demand, premium and non-premium, and let \( I \) denote the inventory on hand at the start of the given period. Trivially, \( \beta_1(I) = 0 \) when \( I \leq k \). For \( I > k \), we have:

\[
\beta_1(I) = \frac{\mathbb{E}[\min\{D, I - k\}]}{\mathbb{E}[D]}
= \frac{\sum_{x=0}^{I-k-1} x \mathbb{P}\{D = x\} + \sum_{x=I-k}^{\infty} (I-k) \mathbb{P}\{D = x\}}{\mathbb{E}[D]}
= \frac{\sum_{x=0}^{I-k-1} x (\lambda_1 + \lambda_2)^x e^{-\lambda_1 - \lambda_2}}{\lambda_1 + \lambda_2} + \sum_{x=I-k}^{\infty} (I-k) (\lambda_1 + \lambda_2)^x e^{-(\lambda_1 + \lambda_2)}
= \frac{\sum_{x=0}^{I-k-1} x (\lambda_1 + \lambda_2)^x e^{-\lambda_1 - \lambda_2}}{\lambda_1 + \lambda_2} + (I-k) \left( 1 - \sum_{x=0}^{I-k-1} \frac{(\lambda_1 + \lambda_2)^x e^{-(\lambda_1 + \lambda_2)}}{x!} \right)
= \frac{I-k}{\lambda_1 + \lambda_2} - e^{-(\lambda_1 + \lambda_2)} \sum_{x=0}^{I-k-1} \frac{(I-k-x)(\lambda_1 + \lambda_2)^{x-1}}{x!}
\]

The premium demand fillrate \( \beta_2 \) is harder to calculate. We model the system as a delay time model (cf. Arts and Van Houtum, 2021). Let \( X \) be the time until the inventory on hand reaches critical level \( k \) as the delay. Then \( X \sim \text{Erlang}(\lambda_1 + \lambda_2, I-k) \) when \( I-k > 0 \) and \( X = 0 \) otherwise. Let \( Y \) denote our delay, the time between the moment the inventory on hand reaches \( k \) and the moment it becomes 0. Then \( Y \sim \text{Erlang}(\lambda_2, \min\{I, k\}) \). If \( I \leq k \), \( X = 0 \) and the demand follows a Poisson process with a constant rate. \( \beta_2(I) \) can then be calculated in a manner similar to \( \beta_1(I) \):

\[
\beta_2(I) = \frac{I}{\lambda_2} - e^{-(\lambda_1 + \lambda_2)} \sum_{x=0}^{I-1} \frac{(I-x)(\lambda_2)^{x-1}}{x!}
\]

Let us now derive an expression for \( \beta_2 \) when \( I > k \). As per the PASTA property, the fill rate is equal to the proportion of time spent in a state where \( I > 0 \). Consequently, the fill rate is equal to 1 minus the proportion of time spent in a zero-inventory state. If we use \( t \) to denote the time to stockout, the time spent in this state is \( [1-t]^+ \), leading to the following derivation for \( I > k \):

\[
\beta_2(I) = 1 - \int_0^\infty \left[ 1 - t \right]^+ f_{X+Y}(t) dt
= 1 - \int_0^1 (1-t) f_{X+Y}(t) dt
= 1 - \int_0^1 (1-t) \int_0^t f_Y(t-x) f_X(x) dx dt
= 1 - \int_0^1 (1-t) \int_0^t \frac{(\lambda_2)^k (t-x)^{k-1} e^{-(\lambda_2)(t-x)}}{(k-1)!} \left( \frac{(\lambda_1 + \lambda_2)^{k-1} e^{-(\lambda_1+\lambda_2)x}}{(I-k-1)!} \right) dx dt
\]

68
We now subsequently change the order of integration, substitute by $u = t - x$ and expand the expression:

\[
1 - \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \int_{t=x}^{1} \frac{(1 - t)^{k} t^{k-1} e^{-\lambda_2(t-x)}}{(k-1)!} \ dt \ dx \\
= 1 - \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \int_{u=0}^{1-x} \frac{(1 - u - x)^{k} u^{k-1} e^{-\lambda_2 u}}{(k-1)!} \ du \ dx \\
= 1 - \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \int_{u=0}^{1-x} \frac{(\lambda_2)^{k} u^{k-1} e^{-\lambda_2 u}}{(k-1)!} \ du \ dx \\
+ \int_{x=0}^{1} k(\lambda_1 + \lambda_2)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \int_{u=0}^{1-x} \frac{e^{-\lambda_2 u}}{k} \ du \ dx \\
+ \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \int_{u=0}^{1-x} \frac{e^{-\lambda_2 u}}{(k-1)!} \ du \ dx
\]

The functions entailed by the integrals on the right-hand side are now Erlang probability distribution functions. We can hence replace the integrals with the Erlang cumulative density functions, and expand the resulting expression:

\[
1 - \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \left( 1 - \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n e^{-\lambda_2(1-x)}}{n!} \right) \ dx \\
+ \int_{x=0}^{1} k(\lambda_1 + \lambda_2)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \left( 1 - \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n e^{-\lambda_2(1-x)}}{n!} \right) \ dx \\
+ \int_{x=0}^{1} \left( \frac{\lambda_1 + \lambda_2}{(I - k - 1)!} \right)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x} \left( 1 - \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n e^{-\lambda_2(1-x)}}{n!} \right) \ dx
\]

\[
= 1 - \int_{x=0}^{1} \frac{(\lambda_1 + \lambda_2)^{I-k} x^{I-k-1} e^{-(\lambda_1 + \lambda_2)x}}{(I - k - 1)!} \ dx \\
+ \frac{(\lambda_1 + \lambda_2)^{I-k} \lambda_2}{(I - k - 1)!} \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} \ dx \\
+ \frac{k(\lambda_1 + \lambda_2)^{I-k} \lambda_2}{(I - k - 1)!} \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k} \frac{(\lambda_2)^{n-1} (1-x)^n}{n!} \ dx \\
+ \frac{I - k}{\lambda_1 + \lambda_2} \int_{x=0}^{1} \frac{(\lambda_1 + \lambda_2)^{I-k+1} x^{I-k} e^{-(\lambda_1 + \lambda_2)x}}{(I - k)!} \ dx \\
- \frac{(\lambda_1 + \lambda_2)^{I-k} \lambda_2}{(I - k - 1)!} \int_{x=0}^{1} x^{I-k} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} \ dx
\]

69
We finish our derivation by again substituting for the Erlang cumulative distribution function on lines (1), (3) and (5), and rewriting the resulting expression:

\[
= 1 + \left( \frac{k}{\lambda_2} - 1 \right) \left( 1 - \sum_{n=0}^{I-k-1} \frac{(\lambda_1 + \lambda_2)^n e^{-(\lambda_1 + \lambda_2)}}{n!} \right) \\
+ \frac{I - k}{\lambda_1 + \lambda_2} \left( 1 - \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-(\lambda_1 + \lambda_2)}}{(I-k)!} - \sum_{n=0}^{I-k} \frac{(\lambda_1 + \lambda_2)^n e^{-(\lambda_1 + \lambda_2)}}{n!} \right) \\
+ \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} dx \\
- \frac{k}{\lambda_2} \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} dx \\
- \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} dx \\
= 1 + \left( \frac{I - k}{\lambda_1 + \lambda_2} + \frac{k}{\lambda_2} - 1 \right) \left( 1 - \sum_{n=0}^{I-k-1} \frac{(\lambda_1 + \lambda_2)^n e^{-(\lambda_1 + \lambda_2)}}{n!} \right) \\
- \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_1}}{(I-k)!} \frac{1}{\lambda_1 + \lambda_2} \\
+ \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \left( 1 - \frac{k}{\lambda_2} \right) \int_{x=0}^{1} x^{I-k-1} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} dx \\
- \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \frac{(\lambda_2)^{k-1}}{(k-1)!} \int_{x=0}^{1} x^{I-k-1} (1-x)^k e^{-\lambda_1 x} dx \\
- \frac{(\lambda_1 + \lambda_2)^{I-k} e^{-\lambda_2}}{(I-k-1)!} \int_{x=0}^{1} x^{I-k} e^{-\lambda_1 x} \sum_{n=0}^{k-1} \frac{(\lambda_2)^n (1-x)^n}{n!} dx
\]

In our C++ implementation, the remaining integrals are solved numerically using the composite Simpson’s 1/3 rule with \( m = 20 \) intervals, which empirically leads to accuracy up to the 5th decimal digit. As the summations within the integrals are independent of \( I \), they can be precomputed over different values of \( x \) in \( \theta(mk) \) time. The effective precomputation runtime complexity is hence given by \( \theta(m + I) \) per value of \( I \).
Appendix C  Data Preprocessing

C.1  Benchmark Study

The aim of this appendix is to describe the preprocessing of the data collected for the benchmark study. Section C.1.1 explains how ROPs and SSLs are retrieved. Section C.1.2 serves the same purpose for demand rate forecasts. Preprocessing of the GESA sourcing sequences is described in Section C.1.3 Section C.1.4 details how transportation costs and leadtimes are estimated. Section C.1.5 examines the preprocessing for replenishment locations and leadtimes. Finally, Section C.1.6 elaborates on how the proportion of routine, priority and emergency SOs is calculated.

C.1.1  ROPs and SSLs

We retrieve data on the ROPs from the mod_mm_plant.csv and mod_mm_sloc.csv files, and retrieve data on the SSLs only from the mod_mm_plant.csv file. Both files contain ROPs for unique combinations of material ID and location, but only mod_mm_plant.csv also contains the SSLs. We start preprocessing by removing data points where LOCALPLANTS.txt does not contain the data point’s location, where STIPT_PART.txt does not contain the data point’s material code. For the SSLs, this step concludes the preprocessing. For the ROPs, we concatenate the datasets from the two .csv files and sum ROPs where the location/material combination is the same.

C.1.2  Demand Rate Forecasts

Demand rate forecasts are retrieved from the mod_mm_forecast_parts.csv file. This file contains demand rate forecasts for unique combinations of part ID, location and forecast category. Our first preprocessing step is to remove data points where the location cannot be found in the LOCALPLANTS.txt file, or where the material cannot be found in the STIPT_PART.txt file. Next, we delete all data points where the forecast category is ‘Forecast Sales’, ‘UIR’ or ‘Diagnostics demand’. ‘Forecast sales’ is related to sales orders and ‘UIR’ to UIR demand, which are both out of scope. ‘Diagnostics demand’ does not lead to actual part usage and can therefore be ignored.

The remaining data consists of seven forecast categories, namely ‘Additional demand’, ‘Factory Forecast CPA’, ‘Factory Forecast GPA’, ‘Factory Forecast LPA’, ‘Forecast CPA’, ‘Forecast GPA’ and ‘Forecast LPA’. Forecasted non-DRT demand rates for a given location/part combination are obtained by summing rates from ‘Factory Forecast CPA’, ‘Factory Forecast GPA’, ‘Forecast CPA’ and ‘Forecast GPA’. Forecasted DRT demand rates are obtained by summing rates from ‘Factory Forecast LPA’ and ‘Forecast LPA’. ‘Additional demand’ is counted as DRT for LWs and as non-DRT for CWs. Both demand rates are divided by 30.5 to obtain mean arrivals per day. Finally, we remove all negative demand rate forecasts, which can be considered erroneous.

C.1.3  GESA Sourcing Sequences

The GESA emergency sourcing sequences are retrieved from the EXPORT_RBA04.xlsx file, which is a direct export from SAP. We start preprocessing this file by eliminating all data points where the ‘To’ or ‘From’ location is not in the LOCALPLANTS.txt file. Next, we assume that the emergency sourcing order is the same for LPA, CPA and GPA contracts. We hence remove data for CPA and GPA contracts (‘*UC3’ and ‘*UG3’), as this data can be considered redundant under our assumption. Finally, we remove data with ‘ACTIV’ values that are not ‘YP_3’, ‘YS_3’, or ‘YF_3’. The remaining LPA sourcing sequences (‘*UL3’) are used as input for training our DRL model, whereby sourcing locations are sorted in ascending order on the ‘COSTA’ column.
C.1.4 Transportation Costs and Leadtimes

Data on transportation costs and leadtimes is retrieved from the `transport_params.csv` file, which contains trimmed historical cost and leadtime averages for unique combinations of sending location, receiving location and shipment type. Trimmed means are obtained by leaving out the 16% lowest and 16% highest observations for each unique combination. To ensure reliability of data, we remove all combinations with less than 10 observations. We further remove all combinations where the mean cost or leadtime is lower than 0 and round all leadtimes higher than 14 down to 14, as we consider our historical data to be unreliable in such cases. The remaining data points cover only 37.56% of unique combinations. We impute missing values by aggregating on region or continent. Concretely, we apply the following procedure:

1. If the number of observations for the sending location to the receiving location’s region is at least 10, use the weighted average of these leadtimes and costs.
2. Else if the number of observations for the sending location’s region to the receiving location is at least 10, use the weighted average of these leadtimes and costs.
3. Else if the number of observations for the sending location’s region to the receiving location’s region is at least 10, use the weighted average of these leadtimes and costs.
4. Else if the number of observations for the sending location to the receiving location’s continent is at least 10, use the weighted average of these leadtimes and costs.
5. Else if the number of observations for the sending location’s continent to the receiving location is at least 10, use the weighted average of these leadtimes and costs.
6. Else if the number of observations for the sending location’s region to the receiving location’s continent is at least 10, use the weighted average of these leadtimes and costs.
7. Else if the number of observations for the sending location’s continent to the receiving location’s region is at least 10, use the weighted average of these leadtimes and costs.
8. Else if the number of observations for the sending location’s continent to the receiving location’s continent is at least 10, use the weighted average of these leadtimes and costs.

The collected and estimated leadtime averages are expressed in hours. However, our simulation environment models shipment arrivals as happening just before decision moments, which are modelled as daily occurrences. Leadtimes therefore need to be expressed as integer days. As a last step, we hence divide the leadtimes averages by 24 and round them up to the nearest integer.

C.1.5 Replenishment Hubs and Leadtimes

Data on the replenishment leadtimes is collected from the `mod_mm_material_cwh_masterdata.csv` file. This file contains both the replenishment leadtimes for consumed materials to a central warehouse, and the central warehouse where the replenishment takes place. Our first preprocessing step is to remove data points where the parts are not included in the `STIPT_PART.txt` file. Next, remove data points where the value for the ‘CentralWarehouse’ column does not match the value for the ‘Plant’ column. We assume that leadtimes of 5 days or less are erroneous and replace them with the standard replenishment leadtime of 100 days, and do the same for the 0.31% of SKUs with a missing value. We further assume that SKUs with a missing value for ‘Plant’ are replenished at the central warehouse in Veldhoven. Finally, we add 14 days to all replenishment leadtimes to account for the time it takes to process inbound stock at the central warehouses.
C.1.6 Service Order Types

We collect data on the proportions of service order types from the OF_Report_v4.csv file, which is an export from the Order Fulfillment v4 Spotfire dashboard. This file contains data on among others the type of SO, its creation date and its UND status. To read in the file without errors, we first manually delete irrelevant columns. Next, filter all SOs whose ‘Cancelled Orders’ column value equals ‘Cancelled’ and all SOs whose ‘Priority_names’ column value does not equal ‘Routine’, ‘Priority’ or ‘Emergency’. Finally, we count the number of occurrences for ‘Routine’, ‘Priority’ and ‘Emergency’. The proportion of demand for each of the SO types is then given by this count divided by the total number of SOs in the dataset.
C.2 Manual Validation

This appendix describes the preprocessing of the data collected for the manual validation, and consists of three sections. Section C.2.1 elaborates on the parts in scope for the manual validation. Section C.2.2 details how we retrieve the inventory on hand, inventory position, ROP and SSL. Lastly, Section C.2.3 explains how we retrieve the outstanding service orders. We also use data on the demand rates, transportation costs and transportation leadtimes for the manual validation, which we preprocess in the same manner as in Section C.1.2.

C.2.1 Filtering SKUs

The parts in scope for the manual validation differ from those of the benchmark study. We remove parts for which we have missing data on inventories, inventory positions, demand rates, ROPs, SSLs, replenishment leadtimes or replenishment location. Furthermore, we remove parts for which there are active UIR events in the mod_mm_zmatconf.csv file and parts whose material code does not start with ‘SERV’.

C.2.2 Inventory Levels

We retrieve the inventory on hand, inventory position, ROP and SSL from the replenishment_data.csv and mod_mm_stock_integrated.csv files. The former is an export from NORA, and the latter is one of the NORA input files. We preprocess the data as follows. From the mod_mm_stock_integrated.csv file, we identify the amount of sales order stock by looking at the ‘Quantity’ and ‘StockCategoryDescription’ columns. This quantity is subtracted from the ‘Stock’ column of the replenishment_data.csv file. Finally, we also remove all parts for which the demand rates or replenishment location are not known.

Variables are calculated as follows. The inventory position is calculated by summing the ‘Stock’, ‘dnIncoming’ and ‘totalInTransit’ columns. The inventory on hand is given by the ‘Stock’ column. The ROP is equal to the sum of the ‘ROP’ and ‘ROP SLOC’ columns, minus the ‘totalInit’ column. Finally, the SSL is defined to be the minimum of the newly calculated ROP and the ‘SSL’ column.

C.2.3 Outstanding Service Orders

Data on outstanding service orders is retrieved from the mod_mm_nora_sales_orders.csv file. We start preprocessing by removing all rows where the ‘SalesDocumentType’ column has a different value than ‘ZKB’, by removing all parts where the ‘Material’ column value does not start with ‘SERV’ and by removing rows where the ‘OrderPriority’ has a different value than ‘01’ or ‘02’. Next, we transform the ‘RequiredDate’ to a numerical value and ‘OrderPriority’ to a boolean. Service orders past their UND have their values set to 0, whereas all other service orders have their values set to the time until their UND in days. Lastly, we remove all service orders for parts for which either no inventory levels are known, no demand rates are known or for which the replenishment location is unknown.
Appendix D Hyperparameter Settings

D.1 Ablation Study

This appendix contains an overview of the hyperparameters used in the ablation study. Table 16 presents the hyperparameters used in the first experiment for scenarios 1 to 4. Table 17 presents the hyperparameters used in the second experiment for scenario 5a (slow movers) and 5b (fast movers). The GELU activation function refers to the Gaussian Error Linear Unit function introduced by Hendrycks and Gimpel (2016), and the Adam optimizer refers to the adaptive moment estimation algorithm introduced by Kingma and Ba (2014).

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden layers</td>
<td>128,64,32</td>
<td>256,128,64</td>
<td>512,256,128</td>
<td>1024,512,256</td>
</tr>
<tr>
<td>Dropout layers</td>
<td>0.05,0.15,0.01</td>
<td>0.05,0.15,0.01</td>
<td>0.05,0.15,0.01</td>
<td>0.05,0.15,0.01</td>
</tr>
<tr>
<td>Activ. function</td>
<td>GELU</td>
<td>GELU</td>
<td>GELU</td>
<td>GELU</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>Adam</td>
<td>Adam</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>Mini-batch size</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Horizon length</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Alpha</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Min. nr. rollouts</td>
<td>20</td>
<td>40</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>Max. nr. rollouts</td>
<td>100</td>
<td>200</td>
<td>400</td>
<td>800</td>
</tr>
<tr>
<td>K-val optimizer</td>
<td>1.96</td>
<td>1.96</td>
<td>1.96</td>
<td>1.96</td>
</tr>
<tr>
<td>Frac. rand. actions</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Max. nr. samples</td>
<td>10m</td>
<td>10m</td>
<td>10m</td>
<td>10m</td>
</tr>
<tr>
<td>Nr. generations</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Timeout</td>
<td>1s</td>
<td>10s</td>
<td>10s</td>
<td>14,400s</td>
</tr>
</tbody>
</table>

Table 16: Hyperparameter settings for the first four experiments of the ablation study

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Scenario 5a</th>
<th>Scenario 5b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden layers</td>
<td>512,512,256,256</td>
<td>512,512,256,256</td>
</tr>
<tr>
<td>Dropout layers</td>
<td>0.01,0.01,0.01,0.01</td>
<td>0.01,0.01,0.01,0.01</td>
</tr>
<tr>
<td>Activ. function</td>
<td>GELU</td>
<td>GELU</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0002</td>
<td>0.0002</td>
</tr>
<tr>
<td>Mini-batch size</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Horizon length</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Alpha</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Min. nr. rollouts</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>Max. nr. rollouts</td>
<td>400</td>
<td>800</td>
</tr>
<tr>
<td>K-val optimizer</td>
<td>1.96</td>
<td>1.96</td>
</tr>
<tr>
<td>Frac. rand. actions</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Max. nr. samples</td>
<td>10m</td>
<td>10m</td>
</tr>
<tr>
<td>Nr. generations</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Timeout</td>
<td>14,400s</td>
<td>28,800s</td>
</tr>
</tbody>
</table>

Table 17: Hyperparameter settings for the last two experiments of the ablation study
D.2 Benchmark Study

This appendix provides an overview of the benchmark study’s hyperparameters and optimization thereof. Table 18 presents the Optuna (Akiba et al., 2019) hyperparameter optimization settings. Table 19 shows the resulting hyperparameter settings, which were used for the benchmark study. The GELU activation function again refers to the Gaussian Error Linear Unit function presented by Hendrycks and Gimpel (2016), and the AdamW optimizer refers to an adjusted version of Adam presented by Loshchilov and Hutter (2017).

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Min. value</th>
<th>Max. value</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nr. of layers</td>
<td>3</td>
<td>5</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>Layer size</td>
<td>128</td>
<td>1024</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>Dropout prob.</td>
<td>0.0002</td>
<td>0.32</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0001</td>
<td>0.16</td>
<td>Logarithmic</td>
</tr>
</tbody>
</table>

Table 18: Optuna trial settings for hyperparameter optimization of the benchmark study

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden layers</td>
<td>170,631,152</td>
</tr>
<tr>
<td>Dropout layers</td>
<td>0.02644,0.21833,0.27415</td>
</tr>
<tr>
<td>Activ. function</td>
<td>GELU</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.00013039</td>
</tr>
<tr>
<td>Mini-batch size</td>
<td>256</td>
</tr>
<tr>
<td>Horizon length</td>
<td>100</td>
</tr>
<tr>
<td>Alpha</td>
<td>1.00</td>
</tr>
<tr>
<td>Min. nr. rollouts</td>
<td>15</td>
</tr>
<tr>
<td>Max. nr. rollouts</td>
<td>75</td>
</tr>
<tr>
<td>K-val optimizer</td>
<td>1.96</td>
</tr>
<tr>
<td>Frac. rand. actions</td>
<td>0.05</td>
</tr>
<tr>
<td>Max. nr. samples</td>
<td>10m</td>
</tr>
<tr>
<td>Nr. generations</td>
<td>1</td>
</tr>
<tr>
<td>Timeout</td>
<td>115,200s</td>
</tr>
</tbody>
</table>

Table 19: Hyperparameter settings for the benchmark study
Appendix E  Benchmark Algorithm

Our benchmark algorithm closely follows the real NORA decision logic, and is similar to the base agent. The benchmark agent’s decision rules for selecting receiving locations are visualized in Figure 19. The prioritization is the same as for the base agent. However, it is less likely to stop sending shipments due to unavailability of stock, as it allows using more restricted types of stock (e.g., SSL) when the receiving location would have the appropriate type of demand. All other decision rules are the same as for the base agent, and can be found in Section 5.4. Notably, fulfilling based on NAV\(^3\) when selecting receiving locations follows the procedure given in Figure 8, selecting sending locations happens according to the decision rules in Figure 9 and fulfilling based on NAV\(^3\) is done by following the decision tree in Figure 10.

![Decision Tree for Receiving Location Selection](image)

Figure 19: Overview of receiving location selection for the benchmark agent

The benchmark differs from the NORA decision logic in three main ways. First, it differs in terms of scope. Cases such as excess stock, UIR events, sales orders and parameter changes are not included in our scope and hence not included in our benchmark. Second, it only distinguishes between CWs and LWs in terms of replenishment location and non-replenishment location. Considering them differently would give the DRL agent an unfair advantage compared to the benchmark agent, as cost estimates for unfulfilled service orders are not dependent on location. Third, our NORA benchmark does not perform an optimisation loop after action selection to reduce out-of-region and out-of-continent shipments. This optimisation loop is known to have little impact on outcomes due to other restraints in the benchmark agent, such as prioritizing sourcing from within the region, but implementing it would considerably slow down evaluation runs. It is therefore excluded such that tighter confidence bounds can be obtained on simulation run results.

\(^3\)For the ablation study, we leave out the right-hand component of the NAV risk formula to ensure a fair comparison with the DRL model. If this component had not been left out, the benchmark would have been disadvantaged by its prioritization of locations with tighter contracts while the penalty for unmet demand is independent of this prioritization. It hence becomes: $E[NAV] = (\lambda^3 \lambda^3 + m) / (\sum_{j=0}^{\frac{3}{2}} \lambda^j)$