Allocation Decision-Making in Service Supply Chain with Deep Reinforcement Learning

Dmitrochenko, Valentin

Award date:
2020

Department of Mathematics and Computer Science
Information Systems Research Group

Allocation Decision-Making in Service Supply Chain with Deep Reinforcement Learning

Master Thesis

Valentin Dmitrochenko

(1252739)

University supervisors:

dr. Yingqian Zhang — first supervisor
dr. Willem L. van Jaarsveld — second supervisor
dr. Vlado Menkovski — assessor

Company supervisors:

ir. Douniel P.T. Lamghari-Idrissi — TU/e (OPAC)

Eindhoven, 2020

Abstract

Inventory control in customer service supply chain presents a number of complications for large companies, such as ASML, associated with increased operational costs and volatility in service levels towards their customers. These supply chains experience significant complications in efficient long-term planning. Thus, an optimized policy towards managing spare parts inventory stock levels is paramount to improving operational results of this planning. Most of the existing methodologies rely on accurate models of the real-world systems and are vulnerable to uncertainty in demand and lead times within such models. Optimal solutions for such models often become intractable with growth of problem complexity and struggle with inherent stochasticity. Consequently, more generalized and flexible, novel approaches can be applied for the decision making within the inventory control of spare parts.

In this research we model an interconnected network of local warehouses, replenished from a single central warehouse, for spare parts stock of ASML, a large producer of lithography machines. Local stocks are used by the company for after-sales service - mitigation of random demand due to machinery failures. We apply Deep Reinforcement Learning method of Proximal Policy Optimization to the model in order to improve inventory control planning within the model. We optimize the allocation of periodic inventory supply from the central warehouse to local stocks so that the total operational costs are minimized. Consequently, we compare this method with existing baselines (including the currently used system within ASML) and show, that our approach gives an average of 9% reduction in operational costs. The paper describes modeling of the problem and the analysis of the results under other metrics relevant in the domain (e.g. waiting times, stock-outs, etc.) and proposes various scenarios for the application of the DRL methodology in the context of spare parts inventory control.

Contents

List of Figures

List of Tables

Glossary

base stock policy inventory control policy, a special case of well-established (s, S) policy. The latter operates on a simple principle of replenishing the inventory level (stock on hand minus backlogged demand) up until the level S, whenever it drops below the level s. Base stock policy warrants for immediate replenishment in case the stock levels falls below S and is denoted as $(S-1, S)$). This policy is often advocated for in cases with slow moving demand, when the costs of ordering can be neglected in relation to holding and stock-out costs.. vi, 4

fab in microelectronics industry, a semiconductor fabrication plant. vi

fast mover usually characterizes Stock Keeping Unit (SKU)s with high demand rates. This inventory is shipped more often in comparison to the other items, particularly slow movers . vi, vii, 53, 66

installed base technical systems (machinery), installed at the customer sites [42]. vi

slow mover usually characterizes Stock Keeping Unit (SKU)s with low demand rates. This inventory is not shipped as often in comparison to the other items, particularly fast movers. vi, vii, 53, 66

Acronyms

A3C Asynchronous Advantage Actor-Critic. vi

ANN Artificial Neural Network. vi, 2, 7

CSSC Customer Service Supply Chain. v, vi, 3, 4

DRL Deep Reinforcement Learning. vi, 1, 19

DTWP Downtime Waiting for Parts. vi, 4, 51, 86

FIFO First-In First-Out. vi, 15, 46, 87

GAE Generalized Advantage Estimation. vi, 41

IC Inventory Control. vi, 2, 9

MDP Markov Decision Process. vi, 15, 30

NAV Non-Availability. vi, 48, 50

OR Operations Research. vi, 1, 2

PPO Proximal Policy Optimization. vi, 38

RL Reinforcement Learning. vi, 2, 6

SGD Stochastic Gradient Descent. vi, 7, 38

SKU Stock Keeping Unit. vi, vii

SLA Service Level Agreement. vi, 4

TRPO Trust Region Policy Optimization. vi

Chapter 1

Introduction

Up until this day, there is an extensive number of practical problems in the domain of Operations Research, which are directly related to diverse business cases and settings. However, as these problems do present concrete challenges for the modern business, various problem- and domain-specific heuristics have been developed over the years. Furthermore, inventory control for spare part supply chains currently finds itself in a similar situation. Optimal allocation of spare parts in the service supply chain has been considered from multiple angles in the industry and academic circles for years. Consequently, a number of practical approaches or techniques were established for addressing various, often isolated aspects of the problem. Often, such techniques rely on restrictive assumptions and engineered solutions with limited applicability. At the same time, some of these advanced methods produce state-of-the-art results and create substantial value for the business. ASML, a world-leading manufacturer and supplier of complex lithography machinery, has been one of the flagmen for such innovation. Over the past decade, the company implemented and refined a complex system for managing stock planning on every level of its service supply chain. Arguably though, the company's current high-end solutions still struggle with several issues, such as sub-optimal long-term planning, scalability of applied methods, curse of dimensionality, etc.

This paper considers an instance of the optimal stock allocation problem within a supply chain of spare parts in the case of ASML. Optimized stock planning implies millions of euros in cost savings for the company, improved customer satisfaction and reduced uncertainty in its planning. Generally, stock management across the entire service supply chain involves a high number of decisions to be made at various levels of complexity. This research concentrates on the optimization of spare parts stock allocation in a local network of warehouses under service contracts with the customers of the company. Primarily, it focuses on the decision *where, when* and *how much* to allocate inventory from a central stock point.

The optimization methodology, applied in this paper, is Deep Reinforcement Learning (DRL). This a discipline, serving as an umbrella for a variety of algorithmic approaches, aimed at handling large and

complex problems through approximated solutions with certain guarantees of optimality. Recent advancements in the field of Deep Learning was followed by applying Artificial Neural Network (ANN)s in decision making problems with empirical successes within the domain of DRL. This method, however, is novel for the field of inventory control, and was not applied to the spare part allocation problem before. Usage of ANNs within the Reinforcement Learning (RL) frameworks allows for computationally affordable approximation of optimal solutions for largely intractable problems of high complexity, which the case of the problem formulation within this paper.

This chapter describes the background and context of the problem in 1.1, which is followed by the problem description 1.2, research questions 1.3 and the outline of the paper 1.4.

1.1 Problem background

The key challenges in Inventory Control (IC) in general are:

- excessive costs, associated with overabundance of safety stocks placed at different stages of supply chain;
- amplification of the bullwhip effect;
- stochasticity of demand and supply;
- various forced trade-offs between product lead times, transport expenses and available capacity.

Such effects come from the imprecision of interactions between various parties, involved in managing stocks across multi-level supply chain. In its own turn, field of IC in the domain of Operations Research (OR) contains a number of diverse problem settings, which can be isolated and studied independently. Namely, planning of stocks through decisions on stock allocation is one of such settings. The focus of this research is on the specific type of supply chain systems - service supply chain for spare parts. This means that the performance within such system is additionally governed by the contracts, that the company has with its customers. In addition, spare parts for complex machinery relate lumpy demand generation within the system, forming particular patterns and associating with extra risks for holding excessive stocks.

1.1.1 ASML: Customer Service Supply Chain

ASML is one of the largest producers of chip-making machinery worldwide. The company, taking its roots from 1984 with headquarters in Veldhoven, more than 24 thousand employees across locations in 16 countries, designs, develops, manufactures and provides service complex lithography systems for production of semiconductors [1].

ASML supplies high-end complex machines, which require constant maintenance, including service and repairs in case of a malfunction. Its lithographic systems are an integral part of wafer production process for the customers of the company. Therefore, any malfunction of the machinery can cause expensive downtime at the customer sites. Consequently, the company maintains a massive after-sales support structure, which is responsible for guaranteeing minimal disruption for its customers. The aforementioned repairs can be performed only by knowledgeable ASML technicians at the customer sites. In order to perform the repairs, apart from manpower and knowledge, relevant tools and spare parts are required. In particular, the spare parts for ASML machinery may comprise of expensive units of critical importance. Respective department in ASML of Customer Supply Chain Management (CSCM) division is responsible for optimal planning and management spare parts flow within Customer Service Supply Chain (CSSC). Such planning, however, often is complicated by various uncertainties of the real world: randomness of demand, unreliability of supply, force majors, etc.

As per Figure 1.1, we can observe multiple location within CSSC, related to each other through clear hierarchy. Global warehouses (green) serve as the main hubs, from were spare parts are shipped to the continental (orange) and local (blue) warehouses. Continental warehouses, in turn, are the hubs for local stock points in the respective regions. These are the vertical connections within the CSSC. Local warehouses form a network for each region, within which spare parts can be shipped horizontally (e.g. from one local warehouse to another one within the region they are located in). Most naturally, the reality is a little more complex - and shipments technically can be administered across different directions in contrary to the rules of the described hierarchy. However, this structure is the base model of the supply chain in question. Another addition to this structure is the designation of emergency hubs - either continental or global warehouses, which can directly satisfy demand in case of urgency and failure to meet the demand in other, less costly ways. And as far as stock planning goes within the provided structure, ASML has a distinction between global stock planning and field stock planning. The former is related to planning the stock allocation from global warehouses to the continental ones, as well as generally the movement of spare parts on this upper tier. In its turn, the latter refers to planning stock allocation within any region from assigned main warehouse from the upper tier to local warehouses in the region. Here the notion of local *network* is important, as it provides the model basis for this research.

The complex structure described above is a necessity, as the company faces a number of challenges in association with spare parts stock management:

- Increased complexity of the network due to large number of locations worldwide (more then 30 stock points in total) with thousands of different items to be planned for at various stages on the supply chain. Additional difficulties come from the geographic spread of the network and dependence on supply lead times (time between the order for supply and its arrival).

- Nature of the capital goods supply chain means expensive products with low rate unpredictable

demand. This signifies higher holding costs of tied capital and increased transport costs in case of frequent re-balancing of the network. Naturally, in order to combat the uncertainty, high stocks are kept so any random demand can be satisfied directly without the decrease in customer service level. Thus, the holding costs often tend to be higher due to complex forecasting.

- Pressure from ASML customers for increased performance in accordance with Service Level Agreement (SLA)s.

Figure 1.1: ASML Customer Service Supply Chain locations around the world [41]

Another important aspect of ASML spare parts supply chain is the presence of Service Level Agreement (SLA) contracts, which govern the service performance of ASML to its customers. ASML machinery can cost up to €72 000 per hour of downtime [41] for its customers. Downtime generally means all the time, during which machines at customer sites are not operating specifically due to the breakdown of the machinery (as other reasons do not concern ASML under service contracts). Hence, often there are stern restrictions on the total amount (as a relative share) of downtime allowed under the SLA, related to the waiting for the spare part to be delivered to the site of repair. This waiting time is denoted as Downtime Waiting for Parts (DTWP) and can serve as one of the performance metrics in the final evaluation of performance under the contract, as well as in the optimization models for stock planning. These contracts of ASML has other metrics used for a number of cases within the system, but under the scope of this research, the notion of DTWP is sufficient. It is further used in order to assess the performance of the applied methodology.

At this moment, there are two automated systems, involved directly in the optimization of the spare parts in the company. Current stock planning within ASML is based on the models, described in van Aspert [41]. These models, in turn, are built on the idea of base stock policy. The main idea of this policy is determining an optimal safety stock (base stock) level, which is periodically or continuously

replenished. Maintaining this stock level at all times within the supply chain per stock point guarantees optimal performance in terms of balanced costs, related to holding, transport and realization of inventory. Another important part of this solution is the system called NORA, which is directly involved in the governance of spare parts across CSSC. More specifically, NORA also directly governs the replenishment decisions for local stocks, which are the primary focus of this paper. We further consider an adapted version of NORA as the main baseline in this research, detailed in 2.4 - as this is the system directly addressing the replenishment decision.

1.1.2 Problem context and scope

Spare parts management are directly related to machine break-downs and consequent repairs. Failures, in fact, create demand for the local stocks of spare parts. Generally, for a repair to occur, there are three main components, that need to be present:

- Technician, who performs the repair.
- Tools to perform the repair with.
- Spare parts, required to be replaced upon the failure of the machine.

Presence of qualified staff at customer locations is beyond the scope of supply chain optimization. Flow of service tools is also governed by the CSSC, but has a different structure and nature, as most of the articles within this flow are non-consumables with relatively small costs and low essence for immediate downtime prevention. Therefore, the primary source for optimization within CSSC are the spare parts. Thus, this research concentrates only on the flow of spare parts.

ASML service supply chain is complex and involves multiple tiers, as described in the section above. This research does not concentrate on the global planning of stocks on multiple layers, but rather considers field planning within a particular network of local warehouses. Final scope distinction shall be made apparent with explaining exactly what type of decision making is considered. It is formulated as the decision throughout the network on how much stock and when shall be allocated to each of the stock points during the replenishment phase from the central warehouse. Other types of allocations are either out of scope or governed by the dynamics of the model (more details are in Chapter 2).

1.2 Problem domain and description

A simple statement of the problem is the inefficient governance of stock in a network of local warehouses. This network is presented as a collection of local warehouses, which are all replenished from a central warehouse. This replenishment is the allocation decision, which can be optimized, so the long-term operational costs are minimized. Namely, this minimization includes keeping non-excessive stocks (decreased

holding costs) while also minimizing the number of shipments and hence - transport costs. The planning is complicated by stochastic demand and uncertainty in supply lead times. Additional complexity comes from meeting the down time restrictions in accordance with SLAs.

This problem stands at the crossing of various research domains. Below the relevance of each respective area is highlighted. These short summaries provide general justification of each domain, while a more complete overview is given in the next chapter, section 2.1.

1.2.1 Why do we study Inventory Control?

There are several reasons to study possibilities of improving inventory control in the supply chain. First of all, inventory constitutes to large investments and capital tied up in raw materials, work-in-progress, service and finished products, as well as associated with acquisition, transportation, handling and storage costs [2]. Secondly, inventory control has a direct financial and reputational impact on the relationship between supply and demand. It is typical for the supply side to have pre-agreed service level agreements with its customers, upon missing which there might be penalties incurred. Any failure of the supply side to meet the pre-agreed performance leads to customer dissatisfaction due to the avalanche effect, as the demand side cannot also meet their own commitments with their customers. Therefore, IC involves extensive decision making, related to the allocation of stocks in the supply chain. Control over inventory planning allows for reducing overall operational uncertainty of future, as well as increasing sustainability of the business. At the same time, IC problem domains and models often present a challenge in finding the optimal solution. Most of them can be related to complex combinatorial or sequential decision-making problems with large heterogeneous search space.

1.2.2 Why do we apply Reinforcement Learning?

Reinforcement Learning (RL) (introduced in more detail in Section 2.1.2) is widely viewed as a general framework for solving various problems, if they can be formulated as sequential decision-making problems. Practically, this means that if the problem can be formulated as Markov Decision Process [39], RL. This particular problem is believed not only to fit the requirements for RL applications, but warrants for one. Taking decisions at a certain time step tends to propagate the dependencies of this choice in time, meaning that the reward in much later steps would still be related to the previous sequence of actions/decisions taken. Our problem has a clear delayed gratification mechanism, as an allocation decision taken several steps ago can make a significant impact on the operational costs at the current step due to stochastic machine failures. Moreover, the majority of RL methods are simulation-based and work well with generated data. In the case of allocation decision problem, there is no available dataset for applying other learning methods. At the same time, similar models in IC often employ simulations for validation or optimization [2].

It is also important to review application of RL in terms of other available alternatives. Unlike classical Dynamic Programming, RL methods tend to emphasize beneficial experiences and avoid visiting all the states of the system, but only the "useful" ones, which largely decreases the size of the search problem. In contrast with linear and integer programming methods, the iterative nature of RL algorithms allows to avoid too lengthy computations if acceptable results are obtained. Most of RL approaches allows obtaining good approximations of the optimal solutions. RL methods can handle therefore much larger problem sizes, maintaining the quality of the solution and not attempting to receive a closed solution. At the same time, RL methods rely on simulation and generated data, making them more sustainable in terms of data availability, in contrast with other (deep) machine learning methods. RL data is generated often dynamically and does not require manual labeling, which is often a restrictive condition for the supervised setting.

1.2.3 Why do we use Neural Networks?

Customarily, the "Deep" part in DRL means the usage of Artificial Neural Network (ANN)s (more information in Section 2.1.2). The latter are well recognized as a powerful function approximators. RL algorithms employ the power of ANNs to find mappings from complex inputs (state spaces) to complex outputs (actions), representing the sought-for optimized policy. The iterative nature of training of ANNs with Stochastic Gradient Descent (SGD) (primarily) is also well aligned with the RL algorithmic approaches. In the particular problem with spare parts allocation, the curse of dimensionality may severely cripple the performance of other approaches (e.g. Approximate Dynamic Programming, Neuro-Dynamic programming, etc) as we have to reason about the Cartesian product between the sets of spare parts, locations, coupled perhaps with inventory in transit and possibly some other features. Allocation decisions also warrant for a high dimensional representation as they have to be taken in large quantities simultaneously. For example, in a case with a network of 2 warehouses and 2 items, multiple interdependent decisions per warehouse-item pair have to be taken.

1.3 Research questions

The goal of the project in general is the reduction of operational costs in the allocation decisions in the service supply chain of ASML under service contracts, namely network of local warehouses. In addition, currently the planning and allocation of stocks faces increased nervousness. This nervousness tends to have a negative effect on the operations of the company, as it leads to the decreased ratio between stock, consumed in the network, and applied shipments. Large stocks mean increased holding costs and more capital tied in the inventory, while small stock means a higher risk of failing to meet downtime commitments for ASML customers. Large number of shipments (both proactive and reactive) inflict unnecessary costs and uncertainty in local lead times. In addition, downtimes inflicted by stochastic failures pose a

potential problem to be addressed. Nevertheless, it is realistic to attribute all of these components to the total operational costs within the model of ASML network. Therefore, the practical formulation of the project goal can be presented as finding a cost-efficient solution to the aforementioned trade-off in terms of the stock allocation policy in the network, applying the methodology of Deep Reinforcement Learning.

Thus, the main research question can be formulated as:

How can we decrease operational costs, associated with allocation decisions in the ASML service supply chain network, with application of Deep Reinforcement Learning?

The following research questions (RQs) shall expand the main research question (with a more complete list, including sub-questions, can be found in Appendix A):

1. RQ1: What is the current state-of-art for the spare parts allocation in ASML service supply chain?
2. RQ2: What is the current state-of-art for spare parts management in the field of OR?
3. RQ3: How can the problem be formulated and solved with DRL?
4. RQ4: How can the MDP formulation translated into a simulated environment for DRL?
5. RQ5: Which DRL technique shall be applied to the problem formulation?
6. RQ6: How can the DRL application be analyzed, improved and experimented on?
7. RQ7: How the results of DRL application can be evaluated and analyzed?

1.4 Outline

This paper is composed in the following way. In Chapter 2 first part of the chapter provides a summary of the desk research for the problem, which also provides most of the justification for the chosen methodology, as well as deeper context to the problem formulation. Consequently the chapter provides the formalization of the business problem and its translation to the DRL methodology. Chapter 3 describes experiments test bed and provides feedback on general performance results on the number of defined problem instances within certain research scenarios. Chapter 4 provides more elaborate analytical insights into DRL agent training, scenario and behavior analysis of the agent performance in relation to defined baselines. The paper is summarized with conclusions and further research directions in 5.

Chapter 2

Methodology and Formulation

This chapter provides the formalism and the backbone of the research. It addresses research sub-questions RQ1 and RQ2 in 2.1, with RQ1 fully addressed later in section 2.4. Moreover, section 2.2 provides answers on RQ3 and RQ4, and section 2.3 addresses RQ5.

2.1 Literature review

This part of the paper is dedicated to providing a brief overview of the existing research on the domain and methodology of the research. In addition, the section also provides a more in-depth justification for the choices of the main methodology and problem formalism, used throughout the research - as they are primarily based on the notations and some design choices, encountered in the established literature sources. It also contains general theory and notions, used explicitly in the next chapters.

2.1.1 General Inventory Control theory

The topic of efficient Inventory Control has been intensively studied in the domain of OR for years. In general, inventory control largely depends on the type of inventory and the supply chain setting. The former largely relates to which role the inventory performs and which decisions are associated with it, while the latter is related to the topology of the chain, field of operation and product properties (e.g. perishables, chemicals, etc). Axs in [2] provides a fairly simple classification of inventory types:

- Cycle stock. Result of batch production.
- Safety stock. Used for mitigating demand variations.
- Anticipation inventory. Planned to be used for future peak in consumption.
- Pipeline inventory. Ordered but still not delivered.

- Decoupling stock. Used for separation of decision making at various stock points.

Naturally, such functional classification does not have strict boundaries and assumes interaction between the listed inventory types. Nevertheless, the classification helps to attribute the decisions, applied to stocks at various points in the supply. Now, it is useful to mention that generally IC faces a variety of problems, which are addressed by the inventory management system/model rules and structure. These challenges simultaneously form the basic elements of IC framework (mostly composed out on the basis of overview sources, such as [2], [10], [40]).

Demand model

Any inventory system faces an external demand of some form, which can be modeled in a number of ways, e.g. fitting an existing distribution or time-series forecasting modeling. It is common to establish demand models with various complexity - from constant rate demand (a common case with batch production) to non-stationary demand with trend and seasonality. A common direction of research within this domain concentrates on the bullwhip effect (overview in Wang [44]), dealing with the stock optimization through multiple echelons. De Kok [10] also addresses the fact that often demand is modeled with either Poisson (mainly for spare parts management), Upper Bound Demand or Gaussian, as an approximation of a general distribution. It is often the case that the demand modeling drives the tractability of the problem, as it is directly responsible for the number of states the system can find itself in.

Structure of the modeled supply chain

Under the structural definitions we can put such parameters, which relate mostly to the dimensionality of the problem and its solution space, affecting its tractability. It defines how many stages there are in the considered supply chain and how many inventory subsystems are to be managed within the model. There are several key items to mention in relation to the structure of the problem in IC:

- Number of echelons. Number of echelons define how many decision points can be found within the problem. Multi-echelon models tend to suffer from the curse of dimensionality and often require engineered approaches, helping to artificially reduce the dimensionality of the problem by carefully made design choices.

- Here structure relates to the connections between various entities across all the considered echelons. Initially, mostly serial systems were studied, but since late 90s there has been a rise of prominent ways to handle the general systems (with many-to-many type of connections) under certain assumptions.

Cost model and other performance metrics

Inventory is associated with various costs: holding costs, ordering/setup costs, shortage costs or service

constraints. The specifics of the cost model usually is tailored to the system in question. It is common to use simplified linear cost models as objective functions in various optimization methods. Costs are the primary source of optimization for the majority of models in the field [2], [10].

Ordering policy

The heart of the IC, the policy signifies at what time period and in which quantity a particular item shall be ordered at which location. In its essence ordering policy can be represented as a functional mapping from the reviewed state of the system to the inventory orders determined. Most of the model assumptions are made within this framework element. The key ones are listed as follows:

- Review. Inventory levels can be monitored constantly (continuous review) or at fixed periods of time (periodic review). The latter is a more simple and easy to work with idea due to discrete nature of modeling, however continuous systems are more preferable in terms of adaptability.
- Determining the time and the quantity of the order. These two decisions are the key elements, which shall be handled by a ordering policy, as we want to know when to order and how much. There are two mainly recognized approaches towards these parameters.
  - (R,Q) policy. When the inventory position declines to or below the reorder point R, a batch quantity of size Q is ordered [2].
  - (s,S) policy. The reorder point is denoted by s. When the inventory position declines to or below s, we order up to the maximum level S. The difference compared to an (R, Q) policy is consequently that we no longer order multiples of a given batch quantity. Also can be referred as the *base stock* policy in case of (S-1, S) - the reorder happens with any deviation . These models present templates, by which the ordering can occur. One may define an optimization problem on the basis of these templates.
- Backordering and penalized unmet demand. The choice shall be made in terms how the demand, which cannot be met at the time, shall be handled. First option is to backorder it, that is to record for satisfying it in the future. Another option is to apply the Lost Sales setting, under which the unmet demand is somehow penalized. This can be a financial penalty or loss of customer loyalty up to actual lost sales. While the choice seems to be straightforward, it has large implications on the formulated model, often serving a key structural decision. Axs [2] provides a good basic overview of the models, employing either of the options and gives conclusions on the effect this structural choice has.

The overview is by far not extensive, but sufficient to give a general idea about the main elements of the framework. Now, the development of reorder policies in IC dates to several decades back in time. The most well-versed and widely applied optimization problem formulation in the domain was (and per-

haps still is, in certain sense) the Economic Order Quantity approach [14]. Here, by the means of simple convex optimization the reorder lot size is expressed through the fixed setup costs, holding costs and demand rate per item. The model however introduces a number of restricting assumptions, such as constant and continuous demand, constant ordering and holding costs, fixed lead times, atomic deliveries, etc. Naturally, since then there were a number of extensions to the approach as in [19] or [7], relaxing some of the assumptions under specific conditions, but generally the formulation is restricted within the differentiable function of the order quantity.

Naturally, there are more sophisticated methods of formulating and solving IC optimization problems. However, they normally address a specific problem setting, which cannot be extended as a generalized approach due to problem-specific engineering. Examples of such can be Single- or Multi-echelon inventory systems with specific formulations as Newsvendor problem, Dual Sourcing and Lost sales model, referred by e.g. [12]. A good classification of IC problems can be found also in the review by de Kok et al. [10].Most of the papers in the review also present various optimization techniques on the specified IC problem

Spare parts inventory

This project, however, concentrates primarily on the domain of spare parts inventory management. Service parts assist in keeping of the equipment, systems or products in working conditions in response to both scheduled (planned) or unscheduled (due to failure) maintenance [3].There is a number of peculiarities related to this specific inventory type and its control (as per van Houtum [42]).

- Central supply of spare parts to the local stock points is a standard organization for the spare parts supply. This means having one (sometimes several) central warehouses and a larger number of local spare parts warehouses, which can be allowed to ship out stock (main local warehouses) or only receive stock (regular local warehouses) [42].Hence, the specific polarized topology of the network is emphasized. At the same time, multiple researches by van Houtum and Kranenburg, for instance, encourage strong use of lateral transshipments (movement of inventory between local warehouses) - e.g. in case in ASML with Kranenburg [20] and a problem instance with multiple demand classes [43].

- Downtime/Availability constraints are driven by the capital goods production. The occurrence of unexpected failures forms a large part of the random spare parts demand and presents one of the largest challenges in the sector. Downtime constraints impose additional optimization objective and add another dimension to the complexity of the problem in terms of optimal solution search.

- Creation of strong pooling effect. Meeting and bundling as much demand as possible, where the demand directly depends on the demand of capital goods is the general idea behind applying pooling

in the system. In practice, pooling effect within regional and local networks of stock points can be created with active usage of lateral transshipments, linking the nodes in the network.

- System approach and item approach. System approach means creating a multi-inventory model with complex constraints, which allows joint optimization across multiple items. Item approach, in turn, assumes decoupling each item and its constraints and solving the problem instances separately. Van Houtum, for instance, utilizes and recommends system approach specifically for the spare parts IC. Consequent overview of spare parts management [17] and paper show that a major part of the literature is dedicated to the system approach, and there is already a number of well-developed approaches. Van Aspert [41] considers both approaches and finds the increasing complexity of optimization methods for the system approach, and suggests decoupling of the integral planning into two layers, each with own optimization methodology.
- Service measures. Optimization problem definition largely depends on what actually is optimized. The domain of spare parts management has a number of specific metrics apart from standard cost-based optimization setting, such as (per [42] and [21]):
  - Fill rate. Fraction of demand satisfied directly from stock. Several variations were proposed over the years, each related to the probability of satisfying the arrived demand under some conditions, such as customer fill rate, order fill rate, channel fill rate.
  - Aggregate Mean Number of Backorders. The metric helps to establish the average size of the backlogged demand and assess the magnitude of unsatisfied orders.
  - Aggregate Mean Waiting Time until the arbitrary part demand is fulfilled. Waiting time is another type of metric, which shows not the amount of unsatisfied demand, but rather addresses the downtime, incurred within the system.
  - Average Availability - fraction of time a machine is available.
  - Aggregate Mean Number of Stockouts - number of stockouts for all parts together.
  - Extreme Long Down [21] - a certain guaranteed maximally allowed number of long outages.

  The choice of the metrics here generally may impact the consequent formulations of the reward function in RL approaches.

For the rest, the inventory modeling follows the basic framework elements, mentioned for the general inventory systems as above. As per Boone [3] at the same time, there are several key challenges associated with the spare part management, such as planning for the new product introductions, planning for service requirements of the aging products and parts, maintaining repair cycle process discipline. These challenges are indirectly related to the spare parts stock allocation, but may provide a more common ground for establishing the relevance of the problem and its solution.

Methodologies for solving IC problems within spare parts allocation problem setting

There is a vast variety of methods, which are applied for IC problems. Logically, we concentrate on the spare parts field as per previous section of the literature review and review what are the current approaches to handling the allocation problem. It is worth noting that some IC methods from other fields are also given here in cases, where the problem formulation is similar to the one, described in this research.

Generally, there is a lot of diverse literature on inventory allocation in a general setting, while the share pertaining spare parts domain is substantially less by comparison. However, there is a a good depiction of various spare parts supply chain models in in the book of van Houtum and Kranenburg [42]. Moreover, this book describes an actual case of ASML, as the base research for the book was performed in the company, and the models described in the book are the basis of the current ASML systems, designated to the allocation decision-making. The models from these book are diverse and in a general sense make an explicit use of lateral and emergency transshipments to handle unmet demand. They also provide a repair return flow depiction, not considered in the current problem definition. This work is heavily cited and extended in the work of van Aspert [41], presenting the current system of ASML for designating base stock levels within the respective policy. This work describes the design of the complex system within ASML, used for determining the optimal base stock levels both for global and field stock planning. In the thesis, van Aspert uses an adjusted model from [42] for field planning, while the global planning is performed with the application of an adjusted model from Reijnen [32]. Reijnen describes a less restricted model of a general sense, suitable for the metrics applied for global planning. All the models from aforementioned sources are in some way referenced and used further in this paper. Methodologically, the above mentioned models mostly use a greed heuristic for optimization of the chosen metrics - finding the largest added value from a changed base stock levels vector in terms of the ratio of holding costs to the transport costs upon satisfying the downtime constraints. The evaluation of the base stock levels performance differs per model, but generally makes use of the steady state of the system and queuing theory.At the same time, one can refer to a number of other models, described in [4], [43], which deal specifically with the optimization on two-echelon systems with lateral transshipments; [8] solution for heuristical joint optimization, making use of the structure of the problem; [31], exploring optimization of both stocks and staffing with a full backlog (also used in this research); [21] make use of Extreme Long Downs optimization.

However, all these methods usually are more or loss centered around finding an optimal base stock policy under a set of assumptions and often pre-defined rules and using an engineered approach to solve the problem. Multiple heuristics are explored to obtain closed form solutions, most of which are very problem-specific and can hardly be considered a general approach even across similar problem instances. Finally, nearly all considered problem formulations and models mentioned above do apply a variety of

base stock policy, where allocation actually is based on the base stocks and First-In First-Out (FIFO) approach towards handling the order of allocation. Essentially, the general idea in the current research is often to use a base stock policy to decide on the quantity and allocation timing, while the order, in which all the locations in the network are re-supplied, is determined based on which demand came first. A different approach is currently used in ASML towards the ordering, which is described in section 2.4. However, observing the entire domain of solutions for various problem settings warrants a separate literature study due to most of the approaches are being largely tuned to the described problems, we shall concentrate on the case, which resembles our problem formulation from Section 1.2.

At the same time is also worth to briefly mention inventory models, which exist beyond the domain of spare parts. A Multi-Echelon inventory system with lateral transshipments was addressed by Axsater in [2], where it is generally suggested to use stochastic dynamic programming (solving MDP in a recursive manner (Bellman equations) with memoization within stochastic reward) and a list of older papers addressing the issue in various definitions. Olsson [26] similarly proposes non-linear programming approach for a relatively simple case with unidirectional transshipments under assumptions of non-increasing fill-rate and constant lead-times. As for the specific spare parts network case, it is addressed by [42] and describes an approach of decoupling the system into systems with individual local warehouses. The latter, in contrast with the exact evaluation of the problem, allows to solve larger problem instances, which is comparable with our problem statement. The proposed approach however does have incorporated rule-based decisions, defined in a prescriptive manner. The policy obtained therefore is predisposed to some behavior and is not learned from a specific problem instance. Hence, RL based approach can be aimed at solving the same problem, but learning from the problem instance simulation directly. Overview in [9] provides an extensive overview of the existing modeling approaches. It mentions evolution from exact solutions on smaller and simple (under a lot of assumptions) problem instances to various approximate methods such as Linear Programming (LP), Adaptive Dynamic Programming (ADP) or LP-ADP combined approaches. We review papers applying reinforcement learning methods in the next section.

2.1.2 Reinforcement Learning

Generally, Reinforcement Learning is a simulation-based method, which requires an acting entity (*agent*) to learn certain behavioral patterns from a given problem setting (*environment*) through a reward mechanism. This environment finds itself in a number of states as a consequence of decisions (*actions*), taken by the agent. Therefore, agent aims to learn a policy π, allowing it to effectively select actions in states, which lead to the trajectories with the highest cumulative reward. Therefore, a typical problem definition in terms of RL is a Markov Decision Process (MDP) as in Sutton and Barto [39] with a finite set of states S, action set A, reward function $R(s, s')$ and transition probability function $P(s, s')$. The MDP is the base for the (often simulated) environment the agent learns from. Traditionally, finding itself in a state s, it makes an action a according to its current policy π, which leads to a new state s' and results into an

intermediate reward r.

One the main ideas here is the presence of the Markovian property:$p(s_{t+1}, r_t|s_t, a_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots) = p(s_{t+1}, r_t|s_t, a_t)$. The latter means that the next state only depends on the current state and current action, so we do not have to consider all the previous states in the trajectory. Trajectory is simply a sequence of states and actions of type : $< s_1, a_1, s_2, a_2 \ldots >$, which ends with a terminal state of the system.. With the holding of Markovian property we can then derive an iterative method of updating learned policy. Policy $\pi(s, a) = p(a_t = a|s_t = s)$ essentially maps states to actions, so when the state is a start of an arbitrary trajectory, the action is selected that would gain the largest return G (accumulated reward over the trajectory).

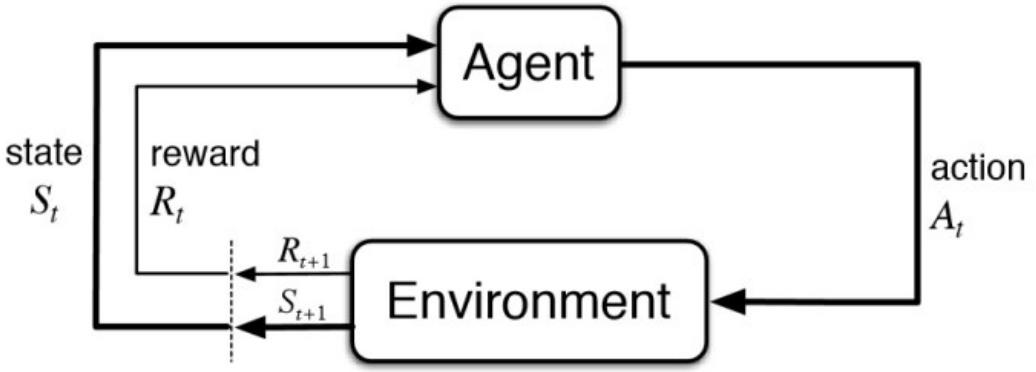


Figure 2.1: Agent-Environment interactions in an MDP [39]

Naturally, agents play out multiple sessions in the simulated environment, called *episodes*. Each episode ends with a terminal state or upon reaching a time limit T. During this episodes information is collected, such as the trajectories and agent learning (which may happen during the run or at the end of the episode, as well as after every fixed number of episodes). RL in some typologies is placed as a field of its own or as a sub-field of Approximate Dynamic Programming. Nevertheless, it is an umbrella-term for various methods, MDP framework with the idea of reducing search space across state and action spaces by emphasizing rewarding experiences. Therefore, RL methods can (to an extent) overcome curse of dimensionality , which affecting Dynamic Programming methods, as RL agents are not required to visit all the states of the system to achieve a solution. At the same time, inherent stochasticity of the majority of RL methods cannot always provide optimality guarantees to the obtained solutions.

MDP variations and RL variations

Proper definition of the MDP is of paramount importance to the success of applying RL methodologies to the problem. It also determines the nature of RL approaches applied for solving a particular formulation of the Markov processes. While the above definition of the MDP is a traditional one, there are other

variations of the formulation, also used in RL. One of them is Semi Markov Decision Process, introduced specifically in RL setting for continuous tasks in [5]. In short, this setting relaxes the directly-follows relation between two states and reviews the continuous time domain. The latter means that there might be alternations to the state of the system between the two states, observed by the agent. Such formulation allows to incorporate simulations with continuous time and still retain the same mathematical models, which are used for solving the regular MDP.

Another extension to the regular definition of the MDP is Partially Observable Markov Decision Process (POMDP), firstly widely introduced for AI tasks in [18]. In this setting the dynamics of the system is also defined by an MDP, but the underlying ("true") state of the system is not known. The agent within RL has to work with the set of observations O instead of working with states directly. It also operates with belief of what state the environment currently is in. The latter requires some additional mathematical complexity to be introduced. Its solutions also are dependable on what was observed, rather on the system states, which introduces another level of uncertainty. Nevertheless, there are proofs that eventually these setting still can yield optimal solutions.

The notion of POMDP is essential in terms of Multi Agent Reinforcement Learning. The latter as opposed to the regular RL setting, employs several agents, each interacting with each other and optimizing a global objective. Since there are several agents, each agent, apart from a global objective, has a local optimization task. Naturally, MARL setting can be used to separate complex tasks into several smaller ones. But in this case each agent does not observe the entire system state and works with POMDP - as it observes only a part of the true system state, which corresponds to the local optimization that the agent is performing.

RL foundations

Optimization in RL generally is determined by the feedback from the reward function. Since most of the problems in this setting have delayed gratification, we often consider a sequence of rewards, corresponding to the sequence of actions taken in the environment. That means that we are not interested in maximizing the reward at each step, but rather the summed reward over many steps in order to avoid greedy optimization, leading to sub-optimal results. Then the *discounted return* as a sum of accumulated rewards is considered $\mathbb{G}_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, where γ is a discount rate (determines on how much emphasis we put on the recent and past rewards). Then we optimize using the return notion, applying functions that can provide information about how well the agent is doing either in a particular state alone, or provided with a specific state-action pair. At the heart of the RL paradigm lie two functions [39]. Value functions is defined as:

$$V^{\pi}(s) \;=\; \mathbb{E}_{\pi}[R_t | s_t = s] \;=\; \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big| s_t = s\Big] \tag{2.1}$$

, and it is used to assess in general how much reward would be obtained starting from the given state s, allowing to assess the "usefulness" of the state. And action-value function is defined as:

$$Q^{\pi}(s,a) \;=\; \mathbb{E}_{\pi}[R_t|s_t=s,a_t=a] \;=\; \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\Big|s_t=s,a_t=a\Big] \tag{2.2}$$

, and it allows to obtain the value of the action, conditioned on the state we take it in. The functions related through

$$V^{\pi}(s) \;=\; \sum_{a}\pi(s,a)Q^{\pi}(s,a) \tag{2.3}$$

It is an important quality of the value functions, that they satisfy the following recursive definition.

$$\begin{aligned} V^{\pi}(s) \;&=\; E_{\pi}[R_t|s_t=s] && (2.4)\\ &=\; E_{\pi}\Big[r_{t+1}+\gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\Big|s_t=s\Big] && (2.5)\\ &=\; \sum_{a}\pi(s,a)\sum_{s'}P^{a}_{ss'}\Big(R^{a}_{ss'}+\gamma E_{\pi}\Big[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\Big|s_{t+1}=s'\Big]\Big) && (2.6)\\ &=\; \sum_{a}\pi(s,a)\sum_{s'}P^{a}_{ss'}\Big(R^{a}_{ss'}+\gamma V^{\pi}(s')\Big) && (2.7) \end{aligned}$$

It is imperative that an optimal policy can be derived from the definitions above - optimal policy is π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)\ \forall s$. Consequently, the optimal value function is given as $V^*(s) = \max_{\pi} V^{\pi}(s)\ \forall s$ and optimal state-action function is given as $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)\ \forall s \in S$.

Any RL algorithm uses one or both of these functions. Initial algorithms in the field directly diverged on the basis of which functions were used, forming value iteration and policy iteration.

In general, typology of RL algorithms is quite diverse. We make a number of distinctions on the basis of meta-features of the learning process. Here, it is worth to mention an important distinction between two types of RL algorithms : On-policy (e.g. SARSA, policy gradient methods) and Off-policy methods (e.g. Q-learning). The distinction is related to which policy is used to collect data for further policy optimization. On-policy methods use the same policy for data collection - behavior policy b - as the policy for taking actions π, so these two are the same. Off-policy methods make a distinction between these two policies. Further differences are mentioned in the next section 2.1.2

There are many algorithm typologies within RL in the method of optimization, so we would emphasize three most popular groups of algorithms, that stand at the roots of the most prominent methodologies today, that showed good empirical results:

- Monte-Carlo (MC) methods. This family of algorithms relies on sampling. In its basic form, an episode is played out by the agent, generating a trajectory with actions taken with accordance to the current policy. The update of the policy happens at the end of the episode and depends on the collected rewards along the trajectory. MC methods are distinguished by the guarantees, imposed on the sampled data and estimators used for reaching the optimal policy.

- Q-learning. The objective of the method is to learn Q-values for each state-action pair in the system. Originally, it is a tabular method, where the table is of size $|S| \times |A|$. It is an off-policy method, since actions are taken in accordance with the maximum Q-value at the state s - greedy selection of action. In the long-term however the greedy selection in accordance with Q-function tend to demonstrate stable results. A natural problem of these algorithms is related to exploration, which is usually addressed by random action selection with a certain probability at any step during the learning process.

- Policy gradient method. The method is based on the policy gradient theorem, which proves that the update of the policy can be independent from the gradient of the state distribution. Thus, we can parametrize the policy (most popular approach, as per first implementation in REINFORCE algorithm from [39] - with a neural network).

An actual algorithm overview is closely related to the DRL methodologies, presented in the next section, while this selection is more generally applicable to all RL methodologies.

Deep Reinforcement Learning

As briefly mentioned before, Deep Reinforcement Learning (DRL) makes use of the Artificial Neural Networks, powerful function approximators. A neural network can be used to approximate (learn) a value function, or a policy function. That is, neural nets can learn to map states to values, or state-action pairs to Q values. Even simpler so, neural nets can directly represent $\pi_\theta(a|s)$, a mapping from $\mathbb{S}$ to $\mathbb{A}$, parametrized with θ parameters (weights) of the neural network. Rather than use a lookup table to store, index and update all possible states and their values, which impossible with very large problems, we can train a neural network on samples from the state or action space to learn to predict how valuable those are relative to our target in reinforcement learning.

Since the domain of RL algorithms is quite diverse and contains different approaches, it might be challenging to describe them all, but night be beneficial to perform a superficial assessment which of them can be potentially used for the IC problem at hand. The introduction of DRL methods roughly from 2013 - value based learning with Deep Q-Network (DQN) [25] and policy based learning TRPO [33] or Asynchronous Actor-Critic Agents (A3C) [24] (Actor-Critic framework on the border between the various types of approaches) - allowed for handling the high dimensional data, and the consequent works on the methods stabilized the learning process sufficiently. Since then Q learning approaches were intensively

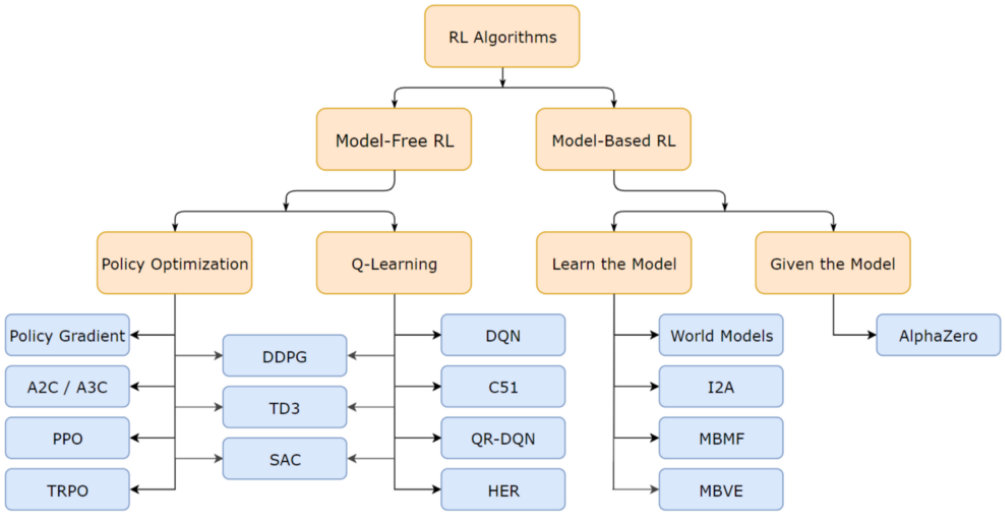


Figure 2.2: Overview of DRL algorithms

developed and studied, resulting in the emergence of a number of well-defined frameworks, such as for instance Rainbow [16] combining all the recent advancements in value based methods. These methods usually were bench-marked in the domain of Atari games or board games (e.g. chess, Go, etc.) with well-defined MDP definitions and usually limited action spaces. Policy based methods have just recently received a wide acclaim, as algorithms such as DDPG [22], managed to show good results on tasks with continuous spaces. The latter traditionally are considered difficult for the value methods due to the need to make a greedy selection from traversal of the entire action space.

Structure of DRL algorithms and approaches can be found in Figure 2.2[1]. First, the distinction is made between the model free and model based methods. The first category does not required any notion of the underlying model of the environment (e.g. no knowledge of transitional probabilities, clearly defining the dynamics of the system). These methods originally were considered to be the main advantage of RL methods in contrast to classical Dynamic Programming, which requires the exact model of the environment. It is often the case that the model of the system is not given or available - it is where the model-free methods prevail. However, the second category can fully utilize the knowledge of the model and result in a more stable and sample efficient learning once the model is available/learned. Moreover, the long-term planning tasks often are only properly learnable with a model, that can be queried dynamically and adaptively during agent training. Further division of model-free methods was addressed and explained in the previous section. Division in model-based methods is rather self-explanatory: either the dynamics model is known in advance or we have to first learn it (often approximated also with the use of neural network by learning a $\mathbb{S} \rightarrow \mathbb{S}$ mapping).

DRL algorithms also can be distinguished as online and offline. Online approaches have agent constantly

[1] https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

interact with the environment and sample data as training progresses. Offline methods first are aimed at collecting a sufficient sampled training dataset (with some policy b) and then inferring an optimized policy π. At the same time, while offline methods are by definition also off-policy, online methods can be both on-policy (e.g. PPO) or off-policy (e.g. DDQN).

RL for IC

The general idea of applying (D)RL methods in IC and OR is building a form of a simulation environment for the agent to interact with. Naturally, simulation is only one way to generate required training data, but is the most popular one. At the same time, some academic research, especially related to classical RL and dating back to 2000s, make use of fixed data sets for learning.

DRL allows to avoid most of the assumptions, made by existing heuristics methods. For instance, we can incorporate stochasticity of various types in the simulated environment and let the agent adapt to it. However, the major assumptions in the setting of RL are to be made in terms defining the MDP for the problem, namely the understanding of the state space S and action space A, as well tuning the reward mechanism. Most of the methods below optimize the reward function in terms of costs, which incorporate most of the optimization metrics, used in the domain of IC.

Despite a peaked interest towards the applied DRL, the current research on the topic is quite limited. There are are two distinct peaks in published papers on the topic. First relevant literature related to the combination of reinforcement learning and OR problem statements dates back to 2000s up to 2010. At that point it was already recognized that RL can be used in the domain, but the techniques still struggled with the curse of dimensionality, large state and action spaces and stable learning, as well as simply correctly defining the problems in MDP setting. A good example of such approach is [11], where the problem is formulated as an SMDP and a custom SMART algorithm is applied to a relatively simple problem instance: a serial supply chain of three stages, a state space as a vector with 3 inventory positions per stage. Consequent works of the same team developed extended the problem formulations, but still retained a relatively small state space, which limits practical applications of this algorithmic approach. The dimensionality issues prevented researchers from extensively exploiting large search spaces of a single agent approaches in the domain. Thus, next researches generally tried to address the problem of collapsing the state and action spaces through a set of heuristics or transformed representations.

There were several attempts to formulate classical supply chain problems in a multi-agent setting such as [38], [46]. The two papers chronologically extend each other and form first attempts to formulate the MDP in terms of several agents with global cost optimization. While no specific results are offered in the papers, the definition of individual reward function per agent and the overall objective function, based on the holding and backordering with costs increasing over time, can be of interest as an example of cost function formulation, specific to the domain. However, the authors propose a poorly scalable vanilla Q-value based algorithm. Another, even earlier attempt with multi-agent setting was tried by Stockheim

et al in [37], but applied in a setting of decentralized optimization of job scheduling and acceptance in a sequential supply chain setting. In the paper each agent (representing a party at any tier of the supply chain, multiple parties are available at each of the tiers) deals with a compressed state space with four relevant domain-specific features, required for the acceptance and scheduling decisions. An interesting detail from the proposed approach is the usage of a deterministic controller, which performs the role of an orchestrator for the agents, structuring agent interactions with the environment and shared knowledge.

Gijsbrechts [12] propose the application of mentioned A3C algorithm for solving the presented problem settings of dual sourcing, lost sales and multi-echelon models for stock control in supply chain of generic nature. The paper used a more extensive state definition than before and applied minimal adaptations towards the problem formulations, emphasizing DRL as a general problem solving approach. It is interesting in our case as it uses a type of decision close to the allocation decision in our problem setting. The dynamics of the used model, however, differs substantially in terms of time domain, capacity effect on demand generation and transport options. Still, the reward design and methodology applications are of interest from the perspective of practical application. Another take on DRL in terms of a supply chain problem is the governance of inventories in terms of classical setting of Beer game, presented in [28]. This a setting with a sequential supply chain, where the flow of inventory goes downstream from a manufacturer to a retailer with some possible intermediaries. The idea of the authors relies on formulating a POMDP problem for a single agent, responsible for a particular stage in the supply chain (e.g. manufacturer) and training it with a DQN algorithm. The trick is that the agents ideally while acting in own interests shall try and reduce the overall costs in the supply chain. Since the agents are purposefully not given the entire state information (due to the basic beer game setting), a general feedback framework is introduced which constantly supplies each agent the information about the overall cost function. Such definition produces a multi-agent setting for learning.The ideas are expanded in a more recent of the same authors in [27].

2.1.3 Summary and Contributions

Existing literature seems to have a gap when it comes to applying a (deep) reinforcement learning methodology within a spare parts inventory control. IC on its own is a subject to optimization in the paper [12] with a well-tested, pure DRL method, but even this problem setting has (1) a different type of supply chain, (2) a different structure of the network and (3) a different (although similar) type of decision. The current work aims at bridging the gap between the latest attempts to address the allocation decisions, handled jointly in [42], and modern capabilities proved by [12] and [28]. More specifically, we formulate the unique definition of the allocation problem and apply the selected DRL method in order to generate process insights of technical nature, evaluation of DRL method performance and analyze the behavior of DRL agent in relation to the research baselines.

2.2 Problem formalization and definition

Here the problem shall be represented as a model, governed by specific rules. In order to apply DRL methodology, it has to be implemented as a simulated environment. Then we provide a translation of this model into an MDP formulation, as the RL methods work with the MDP framework. Simulation implementation details alongside with the validation model for the simulation are described in Appendix B.1. This section of the report is aimed at answering research questions RQ3 and RQ4.

2.2.1 Model description and key definitions

This section a business problem into a formal setting with relevant notation introduced. Consequently, this research concentrates on several instances of the problem, which can be more restrictive then the generalized representation , which is addressed in 3.3. The model below is to a large extent based on the model of van Houtum and Kranenburg, Chapter 5 [42].

Here we expand on the brief description from section 1.2 A local network of warehouses is considered, being a part of global spare part supply chain. One can observe a schematic representation of the model in Figure 2.3, which we refer further in this section. First we present a brief overview of the model from this Figure.

We have several local warehouses (WH1 and WH2 in Figure 2.3), each associated with a group of machines at customer sites. Random demand comes to these local warehouses for certain SKUs. It is satidfied directly from the local stock, but in case it cannot be satisfied directly, we have to use a lateral transshipment from another local warehouse in the network. In case the whole network does not contain the required SKU, we use a costly emergency shipment. In the "happy" flow of this model, all local warehouses are periodically replenished from the central warehouse for all the SKU positions. Central warehouse receives a centralized supply and then we have to allocate this supply across local warehouses, so in the long-term we experience minimized costs in the system. Rest of the notation in the figure is explained below.

Let us define a set of locations L, which are connected with a set of edges E. In the broad definition it is a complete graph. However, a specifically one would define three types of locations, which determine the topology of the network. $\forall e \in E$ we can define a function $e(start, end)$ function defines one connection in the network. The following rules govern the relations between various parts of the network.

- There are no self loops in the network. $\forall\, l \in L : e(l, l) = \emptyset$
- Central warehouse CW: $\exists\, CW \in L \forall\, l \in L\ : CW \neq l\ \wedge\ e(CW, l) \in E\ \wedge e(l, CW) = \emptyset$, thus there are no connections from any other location to the central warehouse.
- Set of Main Local Warehouses LMW: $\forall\, l_1 \in LMW\ \forall\, l_2 \in L \setminus CW\ :\ e(l_1, l_2) \in E$, thus there are connections to all other nodes on the network.

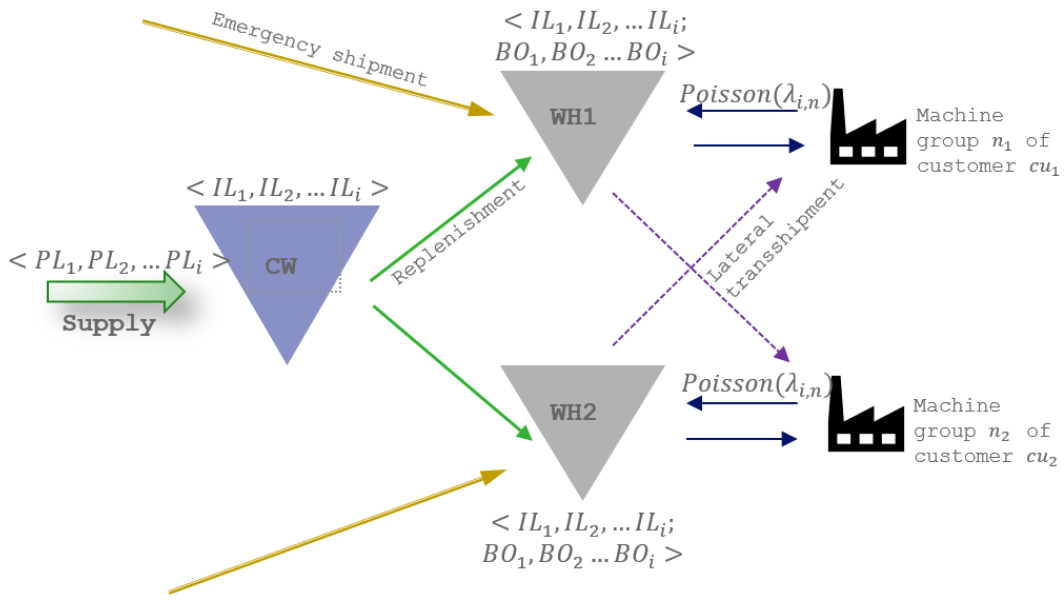


Figure 2.3: General model of a local network

- Set of Local Regular Warehouses: LRW: $\forall\, l_1 \in LRW\ \forall\, l_2 \in L \setminus LRW\ :\ e(l_2, l_1) \in E$, thus there are connections from all the other sets to the regular warehouses.
- The sets do not intersect and form the set L: $CW \cap LMW \cap LRW = \emptyset$, while $CW \cup LMW \cup LRW = L$. Also, for simplicity we define $J = LMW \cup LRW$ set of all local warehouses in the network.

Naturally, there is a set of machine plan groups N, distributed over the locations J. Each machine group n belongs to a particular customer cu from customer set C and requires one or more SKUs i from the item set I.

Relations between sets and network parameters

These sets form the state and structure of the system, as well as influence its dynamics in accordance with the rule, upon which they interact. Locations serve as the nodes on the graph, connected by the edges. As per Figure 2.4, we can observe the relations between the sets. A local warehouse can have multiple machine groups, serviced by the local stock in case of a failure. Machine group can be primarily assigned only to one local warehouse, as well as one machine can be assigned to only one machine. At the same time, customers may have several machine groups, scattered across various locations in the network. Locations are related to each other only by the means of existing edges. The structure of the network and relations between the sets are parametrized upon the definition of the model instance and remain stationary throughout the operations simulation.

Key elements of the model are defined as:

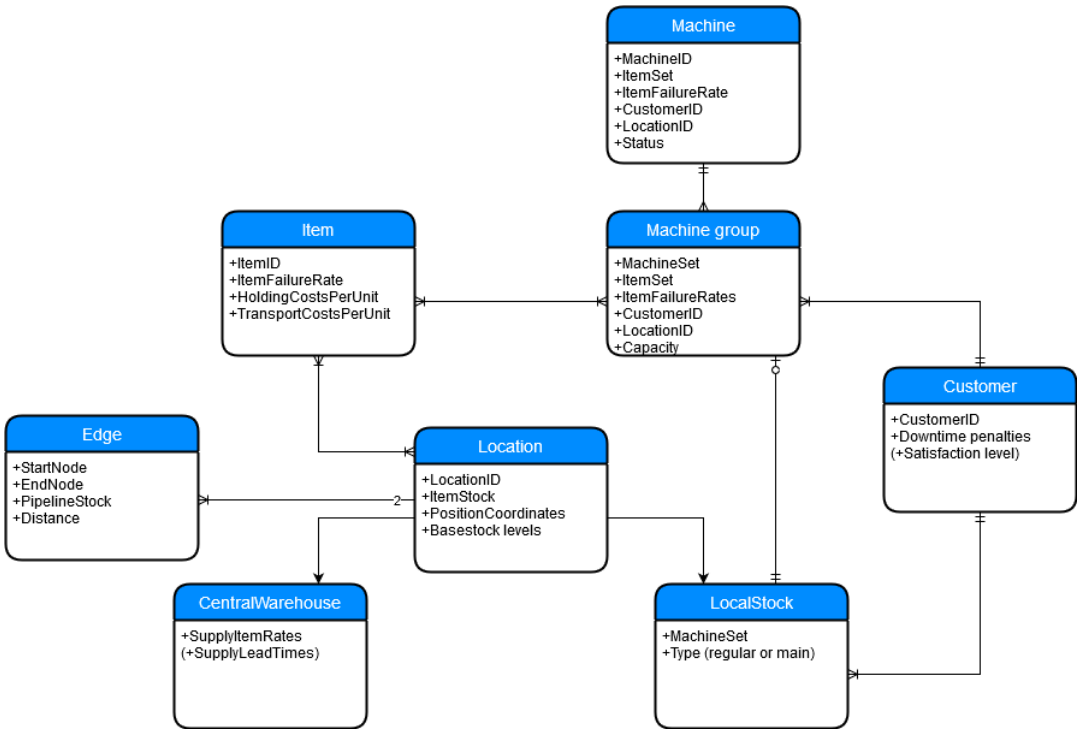


Figure 2.4: Relationships between the sets

- SKU $i \in I$. Each SKU has designated costs associated with it:
  - c_i^{em} are costs per 1 emergency shipment - transporting 1 SKU i from outside the network directly to a local warehouse. These are unified $\forall j \in J$, and are the highest costs associated with transport.
  - c_i^{lat} are costs per 1 lateral transshipment - transporting 1 SKU i from one local warehouse to another. These costs are fixed in order to preserve fixed relation to emergency costs.
  - c_i^h are costs per 1 item of SKU i in stock. These are relatively small in comparison to transport costs, as they are estimated per potentially high stock levels.

  There are no costs associated with supply transport or allocation decision.

  The system has a vector of pipe-line stock $PL_i :< PL_1, PL_2, \ \dots\ , PL_i > \ \forall i \in I$ - all SKUs in transit, which are en route towards the central warehouse from external suppliers.

- Local warehouses $j \in J$ and central warehouse CW, each as $l \in L$. Each of them has a vector of inventory levels $IL_l :< IL_{1,l}, IL_{2,l}, \ \dots\ , IL_{i,l} > \ \forall i \in I$. We also define separately vector of back-logged demand in backorders for local warehouses $BO_j :< BO_{1,j}, BO_{2,j}, \ \dots\ , BO_{i,j} >$. At any time t the relationship between the two is that $min(IL_{i,j}^t, BO_{i,j}^t) = 0$.

  Moreover, each local warehouse is assigned a base stock level within vector $b_j :< b_{1,j}, b_{2,j}, \ \dots\ , b_{i,j} >$

$\forall i \in I$, which is stationary for a problem instance.

- Machine plan group $n \in N$. Each plan group is assigned to exactly one warehouse, which we further denote as j^n. Each group generates demand per SKU, thus we have a vector of demand rates $\lambda_n :< \lambda_{1,n}, \lambda_{2,n}, \ldots, \lambda_{i,n} > \ \forall i \in I$. Additionally, each machine group has base capacity $|n|$ of how many machines are there in the group. Upon any stochastic failure, it is decreased, and it is increased upon each successful repair. Therefore, current capacity $|n|^t$ at time t follows the rule: $0 \leq |n|_t \leq |n|$.

  Finally, each plan group has a *lateral order list* denoted v_n. It is an ordered list of local warehouses, from which the plan group can be re-supplied with item i in case $IL_{i,J^n} = 0$, as j^n is the default option for satisfying demand *locally*. In our network all local warehouses are main and can participate in lateral transshipments. This lateral order list contains $v_n :< 1, 2, \ldots, j > \forall j \in J \setminus j^n$ and is compiled upon problem instance definition (more details in B.1).

- Customer $cu \in C$. Each customer is assigned a number of machine groups. The each customer has a target down time $DTWP_{cu}^{obj}$, defined as sum for all SKUs, associated with machine groups of this customer. Naturally, in accordance with SLAs with customers, there is a constraint on the down time in the system:

$$DTWP_{cu}^{obj} \leq \sum_{n \in N^{cu}} \sum_{i \in I} DTWP_{cu}^{i,n} \tag{2.8}$$

  , where N^{cu} are all machine groups of customer $cu \in C$ and $DTWP_{cu}^{i,n}$ is the accumulated down time per machine group per SKU of customer $cu \in C$.

  Each customer has penalty costs, associated with not being able to satisfy the constraint above, which we define as c_{cu}^{dt}. Such costs are applied per 1% of missing $DTWP_{cu}^{obj}$.

Flow of spare parts and supply

The main dynamic driving force in this model is the demand for SKUs, which is stochastic. Consequently as demand depletes local stocks, they need to be replenished. This replenishment is performed from the central warehouse, and with each replenishment we perform an allocation decision per SKU of (1) how much to allocate and (2) where to allocate. This allocation decision therefore is on *quantity* and *destination* of the replenishment. This is our main decision coming from the employed *policy* on allocation, addressed by the methodologies, described further in sections 2.3 with DRL method and 2.4 with algorithmic baselines.

The central warehouse, upon replenishment ,in turn, has to be replenished. It receives supply of SKUs extraneously. In order to perform supply, we need to know (1) how much to order to the central warehouse and (2) when to order. The model makes this decision internally through a base stock policy. This policy is derived from an existing heuristic for field (local) stock planning from van Aspert [41], and is explained

in more detail in section B.1.4 as it is also used for the validation of the simulation. This heuristic uses steady state system evaluation and applies a greedy selection in line with the method used in Chapter 5 of van Houtum and Kranenburg [42]. Optimized base stock levels $b_{i,j} \forall i \in I, j \in J$ are obtained eventually from the heuristic and are used as an input in the current model. Using these levels per local warehouse, supply order quantities can be determined as:

$$\forall i \in I \; \forall n \in N : \; supply = \sum_{j \in J} (b_{i,j^n} - IL_{i,j^n} + BO_{i,j^n}) - IL_{i,CW} - PL_i \tag{2.9}$$

Such formula allows to form a summed up order for the total supply for each SKU, incoming into the central warehouse after a specified lead time, defined for this SKU. From the time perspective, periodic review is applied with a relatively short review period τ_{sup}. Further on, within the research only models with a period $\tau_{sup} = 1$ are considered, which is close to continuous review setting given the relatively slow moving nature of the demand in the spare parts for complex machinery.

Demand generation

Stochasticity of the demand for spare parts is the key dynamic element in the model. Random failures, constituting the demand per machine and/or machine groups, define the dynamics of the system. Each failure means an event of SKU $i \in I$ being requested by either a specific machine $m^l \in M | j \in J$ or a machine plan group $n^l \in N | j \in J$ at a specific location. As one may see, the demand generation can be realized in two ways, keeping the choice unrelated to the structure of the network, as the location would remain an unchanged part of the demand - in the end, it will still be a demand event for a particular local stock. Practical implementation allows for both methods to be used.

The modeling of the machine demand either explicitly or implicitly is a design decision. In a general scope of the problem it can be realized in either way, but in the model definition within this paper we use a demand, generated per machine plan group. The main difference is the demand distribution for the items. In case of a single machine, it has been showed that the spare parts follow the Weibull distribution:

$$f_X(x; \lambda, k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{k}\right)^{k-1} \exp\left(-(x/\lambda)^k\right) & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.10}$$

At the same time, a group of machines , following Weibull, was proven to generate Poisson process with a constant rate $\lambda_{i,n}$, where $i \in I, n \in N$. At the same time it shall be noted, the simulation tool accommodates groups of machines and allow for setting individual demands per machine-item pair as Weibull in case each group would have a size of 1. If it is larger, it is easier to apply average item failure rates per item and follow Poisson process, which is modeled on the basis of Poisson distribution in accordance with formula:

$$P(k\ events\ in\ interval) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{2.11}$$

, where λ is the arrival rate of events in the defined period. Thus, demand is modeled as an independent Poisson process per SKU-machine group pair $Poisson(\lambda_{i,n})$. One may find that such notation was mentioned previously per demand rate vector λ and is used in Figure 2.3.

The model operates under the assumption of the full backlog, as the spare parts failures signify breakdowns of the machinery. These normally have to be repaired in the end in any case, therefore the lost sales model is not applicable in the considered case (or at least, is not considered under this problem formulation).

Handling demand and transport

Whenever demand occurs per a machine group, it can be handled in accordance with the following procedure:

- *Local stock.* As each demand of SKU $i \in I$ is bound to a particular machine group $n \in N$, and therefore a particular local warehouse j^n, first the local stock at this location IL_{i,j^n} is checked. If it is sufficient to satisfy the demand, the local stock level is reduced accordingly and the demand is considered to be met.

- *Lateral transshipment.* If at the previous stock the local stock was not sufficient to satisfy the demand, it is possible to supply the demand from a different warehouse in the network. Here, another design decision is encountered, as the order in which the other locations in the network shall be traversed has to be defined. It is a static order, which is fixed upon the definition of the problem, but the metric used for the ordering can be different [42] - it can be manually preset or a distance measure can be used. In the latter case we take the next closest local warehouse in the network, and continue in the fashion until we either encountered a sufficient stock level or have traversed the whole list of local warehouses. Specifically our model definition is to use Euclidean distance to define the ordering of the warehouses [2]. Once there is a sufficient stock level to satisfy the local demand, the lateral transshipment is performed from the other local warehouse to the machine group. For simplicity, the distance between the two warehouses is taken into account, while the distance and time for resupply at the customer site are neglected. Thus, the lateral transshipment helps to still satisfy the local demand within the network at the cost of moving the spare parts from another local storage and incurring waiting time, related to the distance between the two local warehouses as well. For the time before the lateral transshipment arrives, the demand is backlogged.

- *Emergency shipment.* In case the entire network does not have sufficient stock to satisfy the local demand, it has to be satisfied externally. Due to comparatively long supply lead times, an emergency

[2] The simulation tool also allows for setting the ordering of warehouses for lateral transshipments manually

shipment procedure is initiated, as it would still allow for reduced and predictable downtime for the customer, associated with the failure. The shipment takes time (typically longer then any lateral shipment) and costs money (typically, much higher then any lateral transshipment).An emergency shipment is the last retreat for the model to meet the local demand, which is backlogged until satisfied.

These steps help to handle any occurrence of random demand in the system. These steps noticeably directly define costs of the model in time and money, and have the following relations:

- Costs: $c_{cu}^{dt} > c_i^{em} \geq c_i^{lat} \geq c_i^{sup} \; \forall i \in I, cu \in C$ with $c_i^{sup} = 0$
- Lead times for local stock re-supply: $l_i^{sup} \geq l_i^{em} \geq l_i^{lat} \; \forall i \in I$

Key assumptions and summary

There are several assumptions to be mentioned in relation to network parameters, also summarizing the points mentioned above:

1. Demand is modeled as independent Poisson processes per SKU-plan group pair with constant demand rates.
2. All demand is backlogged to be handled eventually - as any repair must eventually be addressed in the customer service supply chain.
3. Replenishment from central warehouse bears no cost in money or time. Such choice is logical since these costs are constant and shall not to be optimized for.
4. All newly supplied SKUs are new-buys and there is no return flow for the repair of spare parts.
5. Emergency lead time is constant and unified across all the SKUs and locations.
6. Lateral transshipment lead times are constant across all the SKUs and locations. They depend only on the distances between the local warehouses.
7. Network structure and cost model are fixed and are not changed in simulation.
8. We concentrate on networks with only main local warehouses, hence the assumption of *full pooling* is applied. As networks mostly of relatively small sizes are considered, such assumption is realistic and shall level the playing field across all the methods, applied to solving the model.

A local network of warehouses serves a number of customer locations. Each location has an assigned local warehouse. When demand for a spare part occurs randomly at the customer location, it can be satisfied from this assigned local warehouse at no costs. Otherwise, in case this warehouse has no stock for the demanded item, it can be sought for in the network and shipped from a different warehouse via

a lateral transshipment. And in case the latter is also not possible, we use an emergency shipment at highest costs. Ideally though, we shall always allocate the periodic supply to the central warehouse in such a way, that the total operational costs over the duration of the contract are minimized. Hence, we need to apply a smart policy for such decisions.

The model, described above, is translated into a simulated environment, so it can be used in the DRL methodology. The implementation details and specific design choices are, therefore, listed Appendix B.1.

2.2.2 Defining Markov Decision Process

Now, when the model is described, it needs to be adequately translated into a Markov Decision Process (MDP) formulation, so it can be used for an RL technique. The notion and key points are listed in section 2.1.2. MDP is a well-suited framework for the definition of the problem at hand, as it is aimed at structural decision making within a stochastic environment. More importantly, the model described in the previous section can potentially include multiple decision points. MDP formulation allows to clearly specify which decisions are to be taken into account and optimized for. In this sense, the general model for CSSC field stock planning from the previous section can be translated into multiple MDP settings, each dealing with a particular decision typology.

Markov Decision Process shall be defined with the following terms:

- State space $\mathcal{S}$, which shall contain all the information required for performing a decision within the process.
- Action space $\mathcal{A}$, which shall cover all possible decisions available within the process.
- Transition probability function $\mathbb{P}(s, s')$, which is a mapping between two consecutive states, defining the dynamics of the system with the action $a \in \mathcal{A}$ leading from one state s to another state s'.
- Reward function $\mathcal{R}_a(s, s')$, returning the feedback on how good was the taken action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ leading to the new state $s' \in \mathcal{S}$.

Further on we provide definition for all of these points except for $P_a(s, s')$, as the transitions are not a part to be addressed explicitly in this research. They are handled by the dynamics of the model, defined in the section above.

State space $\mathcal{S}$

State space $\mathcal{S}$ in the given model shall be composed of two dimensions, where the first one is the size of I, as we can define a number of parameters per SKU independently. The other dimension has all relevant information pertaining the considered SKU, either explicit or implicit. Simply put, it should contain all

the information, related to the main mechanisms of simulation mechanics: demand, stocks, supply and shipments. Elements to take into account are:

1. Stock levels $IL_{i,j}$ all SKUs $i \in I$ and local warehouses $j \in J$ of size J. The most basic and important information about the state of the network are the current stock levels per local warehouse. We need this information to count the holding costs of the inventory at given moment in time. Further on, stock levels change is the indication of how the performance of the agent propagates in time and address all the four main elements of the simulation.

2. Stock in central warehouse $IL_{i,CW}$ and pipe-lined stock PL_i for all SKUs $i \in I$ of sizes $|CW| + |PL|$ for this SKU, amounting to 2. This information is also crucial as these two levels directly participate in the allocation decision and supply ordering together with the stock levels across the locations in the network.

3. Current time period t of size 1. Since one of the ideas behind the current MDP definition is to obtain reward over a period of time within the simulated environment, it has to include the notion of time. The current time period is included in the state definition up until the final horizon T.

4. Backorders BO of size $|J|$. Current backorders for each location can be included into the definition of state space as they correlate with the demand and may allow the network to adjust to the existing demand patterns with enough training.

5. Cumulative backorders BO^{cum} of size $|J|$. Accumulating the backorders through the episode allows to give the notion of the accumulated downtime together with mapping the demand patterns. That might be beneficial for delayed gratification handling and constitutes to the learning of the terminal reward.

6. Cumulative downtimes $DTWP_{i,n,cu}$ $\forall i \in I, j \in J, cu \in C$ of size $|M|$. A more straightforward and specialized approach is to directly include downtime per machine group as the episode progresses through the state space. Such number would directly correlate with the terminal reward. We include specifically $DTWP_{i,n} = \sum_{cu \in C} DTWP_{i,n,cu}$

7. Cumulative number of shipments m^{lat} and m^{em} of size 2. These are two numbers, corresponding to the running amount of lateral and emergency shipments respectively. Such information can be added explicitly to give the agent the feedback on the transport aspect of the simulation

8. Structural elements of the simulation, such as costs $c_i^{lat}, c_i^{em}, c_i^h$ and base stock levels b. These are static elements, which are important for making the decision on allocation, as they participate in reward and supply calculations.

All of these elements contribute in various ways to the calculation of rewards in reward function R and define transitions $\mathbb{P}$. While all these elements are relevant for the description of the system state, we have

to define a state space $\mathcal{S}$, usable by the DRL agent. That means that a valid input information for an ANN has to be conjured.

Various combinations may be tested and used within the research. Primary concern with the state definition is to include the exact amount of information required to make the allocation decision. Naturally, we want to include all the information, which is required for transitions and reward calculations within the model. There is, however, a very practical side to it as well. If the DRL agent has a lack of information in the state space definition, it would not be able to learn optimal policy and may converge to simple sub-optimal solutions. And in case the state space definition, supplied to the DRL agent has information overabundance, the ANN can have problems in training, caused by inconsistent gradient updates. In this regard, the most stable and reliable definition of the state space was found to be the combination of elements 1,2,3,6 from the list above per SKU. This definition is used throughout the rest of the research universally for all problem instances, as it strikes a good balance between information richness and necessity. Consequently, such results in the size of the state $\forall s \in \mathcal{S}$ - $|I| \times (|J| + |N| + 3)$:

$$s = \begin{bmatrix} < IL_{1,1} & \dots & IL_{1,|J|} > & < DTWP_{1,1} & \dots & DTWP_{1,|N|} > & IL_{1,CW} & PL_1 & t \\ < IL_{2,1} & \dots & IL_{2,|J|} > & < DTWP_{2,1} & \dots & DTWP_{2,|N|} > & IL_{2,CW} & PL_2 & t \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ < IL_{|I|,1} & \dots & IL_{|I|,|J|} > & < DTWP_{|I|,1} & \dots & DTWP_{cu}^{|I|,|N|} > & IL_{|I|,CW} & PL_{|I|} & t \end{bmatrix}$$

The last element t is constant per row, and such design is maintained for simplicity of implementation.

Action space $\mathcal{A}$

Action space $\mathcal{A}$ of the problem shall cover all possible decisions available within the process. Action in this problem is defined as a matrix of size $|I| \times |J|$, as we have to allocate certain quantity of each SKU $i \in I$ to each location $j \in J$.

$$\forall a \in \mathcal{A},\ a = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,|J|} \\ a_{2,1} & a_{2,2} & \dots & a_{2,|J|} \\ \vdots & \vdots & \ddots & \vdots \\ a_{|I|,1} & a_{|I|,2} & \dots & a_{|I|,|J|} \end{bmatrix}$$

Further we also use notation $a_t \in \mathcal{A}$ to signify an action at time step t, and $a_{i,j}^t$ in order to isolate a concrete allocation of SKU i to local warehouse j. There are several issues to be addressed in the action definition in terms of the MDP formulation.

First, it shall be decided when the actions take place in the process. Our definition of the model in the previous section allows for some flexibility on the spectrum, however it is logical to assign action period as a discrete, constant rate - e.g. each day within the simulation or each full time period $\tau_a = 1$. Another

consideration for such design decision is that such allocation decision period is currently applied within ASML system.

Second, the lower and upper bounds are to be designated. The lower bound of the action space for each SKU-location pair is 0, as at each action step it shall be possible not to allocate anything from the central warehouse. The upper bound definition is a bit more tricky. It is important to take into account how much freedom shall be given to the solution methodology (e.g. DRL agent in our case) in terms of the size of the search space. High upper bound on one hand means more flexible search of the optimal policy. On the other hand it might create a lot of unnecessary states, complicating search space without any additional value. One of the restrictions here is whether an action at any moment of time is available. Notion of *action availability* in terms of the defined problem means that there is enough stock in the central stock to be allocated in accordance with the proposed action. In other words available actions satisfy the condition:

$$\exists a \in \mathcal{A} \, \forall i \in I : \sum_{j \in J} a_{i,j} \leq IL_{i,CW} \tag{2.12}$$

Such action shall be in the set of available actions $a \in \mathbb{A}^{av} \subset \mathcal{A}$. This means that it is plausible to assume a maximum available quantity in the central warehouse. Such estimations can be achieved in various ways: we can evaluate average stock in CW in the steady state, determine a base stock level for the CW or simply determine the maximum supply order, which might come into the warehouse. In the latter case, in accordance with the supply formula and under assumption of empty backorders, it is possible to simply sum up the base stock levels of the local warehouses. Such solution also gives a good estimate of the steady state inventory levels in CW, as these base stock levels are determined in a similar fashion. Thus, the upper bound on the action per SKU-location pair is the sum of the base stock levels across all the locations for these SKU. We present action space to be constrained as:

$$\forall a \in \mathcal{A} : 0 \leq a_{i,j} \leq \sum_{j \in J} b_{i,j} \;\; \forall i \in I, j \in J \tag{2.13}$$

Third, the nature of the action space have to be addressed. Naturally, within the MDP the action space can be only discrete, however, in terms of actual implementation there might be variants on how to realize in practice - refer Appendix B.1.

Reward function $R_a(s, s')$

Reward function returns the feedback on how good was action $a \in \mathcal{A}$ taken in state $s \in \mathcal{S}$ leading to the new state $s' \in \mathcal{S}$. Reward function is directly related to the optimization objective of the problem. In case of our model, we formulate the optimization problem as minimization of total operational costs, incurred by the system C (as per 1.3).

Total operational costs can be formed out of several cost factors. However, these cost factors do express various objectives, related to performance within the mode, which is showcased below. As far as the

design of the reward function goes, the key design decisions are (1) how much reward to give from the observed state and (2) whether special (non-state related) additions and/or alternations to the reward shall be given. The first decision mainly deals with the structure of the reward, while the second decision is addressed later in the section. Reward within this problem consists of three principal cost factors:

- Holding costs c_i^h per each SKU $i \in I$. These costs correspond to the costs of the capital, tied in stocked inventory. They drive the stock levels down, as higher levels would correspond to higher costs.

- Transport costs c_i^{lat} and c_i^{em} per each SKU $i \in I$. This factor corresponds to the costs of shipping spare parts across the network. These costs drive the stock levels up, as high stock does not cause additional re-shipments on account of more demand being satisfied through local stock.

- Downtime costs c_{cu}^{dt} per customer $cu \in C$. These cost factor correspond to the penalties or bonuses, obtained from maintaining a target contract downtime. Simply, there is a target downtime share in the contract between ASML and its customer, e.g. 98% of time all the machines of the customer shall be up. In case the uptime of machines is below this target downtime at the end of the contract (or a designated reporting period), downtime penalties are applied. This is the only type of cost factor which can be negative, signifying a bonus for meeting and possibly exceeding the downtime restriction. Downtime costs are important for the generalized optimization, as they allow for the agent to also optimize for the downtimes, being an important quality metric for ASML customers.

Combining the cost factors above, we can obtain different reward function designs. We define three reward function designs, applied to the model setting.

First, we consider operational costs purely as a combination of holding costs HC and transport costs TC. Such choice is partially inspired by the existing spare part domain research [42], [41],[43] and (D)RL formulations [13], [29]. These two factors form the *interim reward* R_{int}, which can be calculated each (action) step throughout the episode in a traditional fashion for a DRL setting - or, rather, at any time t.

$$\begin{aligned}
&\max \quad \mathcal{R} \\
&\text{with} \quad \mathcal{R} = R_{int} = -(k_{hc} HC_t + k_{tc} TC_t) && \forall t \leq T, \\
&\qquad HC_t = \sum_{i \in I} \sum_{j \in L} IL_{i,j}^t c_i^h, && \forall i \in I, j \in J, \\
&\qquad TC_t = \sum_{i \in I} ((m_t^{lat} - m_{t-1}^{lat}) c_i^{lat} + (m_t^{em} - m_{t-1}^{em}) c_i^{em}), && \forall i \in I, j \in J.
\end{aligned} \tag{2.14}$$

, where k_{hc} and k_{tc} are respectively coefficients for holding and transport costs, which can influence on how much weight each of the elements would have; m_t^{lat} and m_t^{em} are the cumulative numbers of lateral and emergency shipments respectively. These are 0 in the beginning of each episode, and consequently we take the difference between the previous record and the current number to account to all the shipments,

occurred between the two moments t and $t-1$. This design is aimed at controlling the allocation of stock in the network by preserving the balance between quantities of safety stock and number of extra shipments. Holding costs are to drive the stocks down, while transport costs (directly related to stock-out situations) would drive the stock levels up. As far as the downtime of customer machines goes, it is indirectly optimized through the application of optimal base stock levels, obtained from the validation model (described in B.1.4). In accordance with that model, based on van Aspert work [41], base stock levels for local warehouses are optimized with respect to constraints on DTWP (downtime) and make sure we have enough SKUs supplied into the network. Further efficiency of operations depend on the reward definition above.

At the same time, we can directly optimize for downtime in our model with a different reward design. The downtime costs generally correspond to the observations, collected during the entire episode, and tell whether we were able to meet the target within the duration of the contract T. These reward is referred as terminal reward R_{ter} and naturally is calculated at the end of the episode, since then we can judge about the total downtime accumulated. Then the reward design is hybrid, denoted as R_{hyb}, as we use R_{int} for all the timesteps but the terminal, where is supplemented by the downtime costs with R_{ter}:

$$
\begin{aligned}
\max \quad & \mathcal{R} \\
\text{with} \quad & \mathcal{R} = R_{hyb} \\
& R^t_{hyb} = R^t_{int} && \forall t < T \\
& R^t_{hyb} = R^t_{ter} = -(k_{dt} DT_t + k_{hc} HC_t + k_{tc} TC_t) && \forall t = T \\
& DT_T = \sum_{cu \in C} \left(\frac{DTWP^{obj}_{cu} - \sum_{n \in N^{cu}} \sum_{i \in I} DTWP^{tot}_{i,n,cu}}{T} \right) c^{dt}_{cu}) && \forall i \in I, n \in N, cu \in C
\end{aligned}
\tag{2.15}
$$

, where DT is downtime costs; $DTWP^{tot}_{i,n,cu}$ denotes total downtime accumulated during the episode for SKU $i \in I$ per machine group $n \in N$ of customer $cu \in C$; k_{dt} is weight coefficient of downtime costs DT. Such reward design allows to optimize for the downtime within the system directly. As one may notice per formula above, we can receive positive downtime costs in case the target downtimes were not reached. This is intentional, as in this case the agent shall be rewarded for being able to over-perform under the SLA.

However, such hybrid reward function design creates an unbalanced in reward signals the agent receives. On one hand, there needs to be a sizeable difference in the magnitudes between R_{int} and R_{ter} for the terminal reward to contribute to the learning. On the other hand, large reward in the end of the episode may prevent agent from proper learning, undoing the learning received during the episode. Moreover, such hybrid reward leads to high variance in rewards and complicates learning in general and value function V^{π} (refer 2.1.2) prediction specifically. In order to avoid such pitfalls, the reward signal shall be stabilized, which can be achieved by mixing R_{int} and R_{ter} together to form a new interim reward,

denoted as R_{mix} :

$$
\begin{aligned}
\max \quad & \mathcal{R} \\
\text{with} \quad & \mathcal{R} = R_{mix} = -(k_{hc}HC_t + k_{tc}TC_t + k_{dt}DT_t) && \forall t \leq T, \\
& HC_t = \sum_{i \in I} \sum_{j \in L} IL_{i,j}^t c_i^h, && \forall i \in I, j \in J, \\
& DT_T = \sum_{cu \in C} \left(\frac{DTWP_{cu}^{obj} - \sum_{n \in N^{cu}} \sum_{i \in I} DTWP_{i,n,cu}^t}{T}\right) c_{cu}^{dt}) && \forall i \in I, n \in N, cu \in C
\end{aligned}
\tag{2.16}
$$

, where $DTWP_{i,n,cu}^t$ is the cumulative downtime for SKU i per group n per customer cu. As this definition is close to the original interim reward, we have to redefine only the downtime costs DT_t per time t. We also use the correction t/T on how far we are in the episode, so that the later downtime rewards are more important and shall more clearly contribute to the interim reward calculation. Mixed reward design preserves the idea of optimizing the downtimes directly in addition to holding and transport costs, while reduces the variance in rewards and brings down the gap between reward signals.

Thus, designs are used in the research, leading to different results, described in Chapter 3.

- Only interim reward R_{int}
- Both interim and terminal reward with hybrid reward R_{hyb}
- Mixed (interim) reward R_{mix}

These form a set of reward functions $\mathfrak{R} : \{R_{int}, R_{hyb}, R_{mix}\}$. Now, all of them deal with the actual costs, incurred by the system. The final reward design, however, should deal with one more with on more restriction - action availability from 2.2.2, expressed as $\forall a \in \mathcal{A} : 0 \leq a_{i,j} \leq \sum_{j \in J} b_{i,j} \ \forall i \in I, j \in J$. In order to handle availability, it is possible to apply two methods:

- Masking unavailable actions. In such a way, a mask is created, which at each timestep t only allows the policy to choose from available actions $\mathbb{A}^{av}$. Such operation is associated with high computation intensity and quickly may become unviable or even intractable with the growth of action space.
- Penalization of unavailable actions. Such method only computes if $a_t \in \mathcal{A}$ belongs to $\mathbb{A}^{av}$. If it does, no penalty shall be applied or even a bonus can be provided to the agent, while if the action is unavailable, the agent shall receive respective feedback.

Due to increased computational complexity, the option with penalization is realized. The amount of penalty has to be defined then. Instead of using a fixed high penalty, we apply a flexible coefficient, called *miss factor* and denoted as mf. Say, the agent supplies action a_t. Miss factor then corresponds to how much a closest available action (or rather set of actions) was "missed" - floored average across all SKUs.

Then the miss factor is introduced to the reward by multiplying it with the holding costs at this timestep, as this costs are related to the availability of actions through inventory levels vector IL. And if the action was available, miss factor would become ≤ 1 and shall no be applied. Instead we apply a fixed bonus of reducing the holding costs by the factor of 2 (arbitrary empirical choice, which can be adjusted with different problem instances). Thus, we have final reward design as :

$$\begin{aligned}
\max \quad & \mathcal{R} \\
\text{with} \quad & \mathcal{R} = R && \forall R \in \mathfrak{R} : \{R_{int}, R_{hyb}, R_{mix}\}, \\
& mf_t = \left\lfloor \frac{IL^t_{i,CW} - \sum_{j \in J} a^t_{i,j}}{|I|} \right\rfloor && \forall i \in I, \\
& HC_t = mf_t \cdot HC_t && HC_t \in R, mf_t > 1, \\
& HC_t = HC_t/2 && HC_t \in R, mf_t \leq 1
\end{aligned} \tag{2.17}$$

Therefore, penalization is applied within each of the reward design functions and is related to how much wrong the agent was. Consequently, miss factor is also used to observe the progress of training as low miss factor means that the agent is able to at least perform valid actions.

MDP summary and complexity

A short summary on the MDP definition is given here, as per the list defined in the beginning of this section.

- State space $\mathcal{S}$. We define main state space elements and formulate the input state dimensions, used by the consequent DRL methodology. Growth in our state space definition comes from the sets $|I|, |J|, |N|$ primarily (and other factors mentioned above), as well as maximum quantity of stock, that can be accumulated in the network during an episode. To give an example, a network with 4 local warehouses, 4 machine groups and 2 SKUs generates $|S| = 2 \times (4+4+3) \times IL_{max}(S, \lambda) = 22 \times IL_{max}(S, \lambda)$, where $IL_{max}(S, \lambda)$ is maximum attainable stock level, depending on the vectors of base stock levels and demand rates across SKUs and plan groups. Naturally, this can be a very high number, as theoretically thousands of SKUs can be stored in the network in worst case scenario. In practice however, such bound be set safely at $IL_{max}(S, \lambda) = T \times \sum_{j \in L} b_{i,j} \times \sum_{i \in I} \sum_{n \in N} \lambda_{i,n} T$ with $\tau_{sup} = 1$ as it should not be reached by the agent - or can be left unbounded. In case every element of b is 2 and $T = 360$, and summed yearly demand rate of 700, we receive $|S| = 22 \times 360 \times 8 \times 700 = 44352000$, growing very fast.

- Action space $\mathcal{A}$. As shown before, its growth primarily depends on sets $|I|, |J|$ and base stock levels for the upper bound. As per example above, a network with 4 local warehouses, 4 machine groups and 2 SKUs generates $|A| = 2 \times 4 \times \sum_{j \in L} b_{i,j}$. In case every element of b is 2, we receive

$|A| = 64$, growing fast. The upper bound posed here is also a matter of design choice and is subject to deliberation.

- Transition probability function $\mathbb{P}(s, s')$ is not defined explicitly, but is underlying the simulation model for the model defined in the previous section.
- Reward function $\mathcal{R}_a(s, s')$. We apply three possible designs to the problem at hand, however, consequently we show results only per R_{int} and R_{mix}. Reason is that during empirical trials R_{ter} showed inferior performance and was enhanced into R_{mix}. We still mention it as a development step toward R_{mix} as an attempt to include direct downtime optimization.

With such large action and spate spaces and relatively complex reward design, application of DRL may help to approximate a nearly optimal policy. Such approach is considered in the next section.

2.3 Deep Reinforcement Learning methodology

This section give a brief overview of the DRL methodology applied within this research. The main justification was given in previous section 2.1.2 Literature review, and here the actual details of implementation and application are given.

2.3.1 Choice of DRL methodology

Amongst the diverse family of DRL algorithms, it might be uneasy to specify the one approach that would be fitting to the problem at hand. However, it is possible to narrow down the search. As per section 2.1.2, we know the typology of RL algorithms from 2.2, where the first choice is to be made between a model-free and a model-based method. In the considered case, there is a model of the system in a form the simulation, however, the stochastic nature of the simulation prevents using this model directly for the transition probabilities model, traditionally used by the model-based methods. Moreover, learning such a model as a neural network is a separate extensive task within a supervised setting, coupled with the potentially large state space. Finally, a tree-based search across action sequences , applied by the current most prominent model-based methods (e.g AlphaZero [36]), is again unviable in the setting with the highly dimensional action space. Thus the consideration is made in the favor of a model-free method. Consequently, now the choice is between a policy gradient and a Q-value based method. The latter has problems with large action spaces due to argmax operation, employed by these methods for the greedy action selection during training. Such makes these methods very computationally intensive, which is not suitable within this project. Policy-gradient methods however directly optimize the policy with Stochastic Gradient Descent (SGD), mainly via the gradient updates of the weights and biases in the neural network. Thus, the choice is made in the favor of a policy-based method. Within this family of algorithms Proximal Policy Optimization (PPO) [35] has had some supported success with the stochastic environments [15]

and is generally recommended for complex domains with large action spaces (e.g. continuous actions in robotics and physics [6], [23]). There are several advantages of PPO methodology in comparison with other model-free methods:

- Relatively high sample efficiency and speed. PPO allows for collection of training data and then re-using the samples in the training process. In comparison with ACER [45], which requires large replay buffer and reduced speed due to computation, PPO manages to get acceptable results at higher speed of training.
- Parameter tuning. With DRL practicalities of training often come down to the search in the space of model hyper-parameters, which can generate good results. By comparison with its predecessor TRPO [33], PPO has a simplified loss calculation and has fewer parameters that require tuning.
- Expected training stability. In its essence PPO has an idea of taking gradual update steps towards an optimal policy, preventing gradient overflow in the network. Our problem has several factors, contributing to its reward and is stochastic in nature, which is anticipated to produce rewards of high variance - leading to unstable updates in training. While there are various methods to address high reward variance, PPO naturally is designed to handle its repercussions.

2.3.2 Proximal Policy Optimization

Within this research a standard implementation of PPO is applied from the paper of Shulman [35] with the Actor-Critic framework and clipped objective function. An illustrative figure from the paper is provided here as well in Figure 2.5

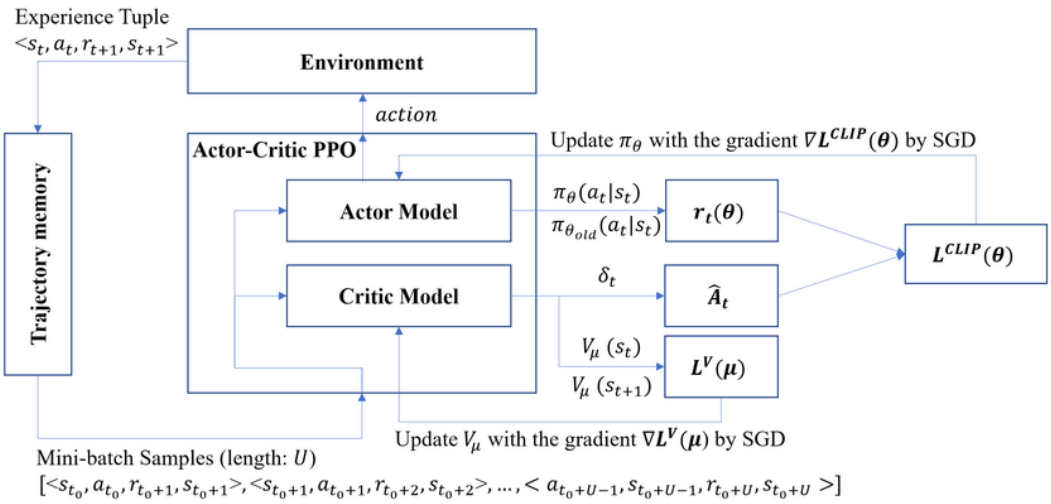


Figure 2.5: PPO architecture with Actor-Critic framework

As mentioned above, PPO is a policy gradient method, which means that it uses a neural network to

parametrize the policy π_θ and optimize it through gradient descent/ascent. Such optimization is allowed by the Policy Gradient Theorem (in Sutton and Barto [39]). Most of the policy gradient algorithms employ various methods to improve learning stability and convergence and address numerous challenges in the domain, e.g. high variance of rewards, high-dimensional actions, convergence to local optima, exploration vs. exploitation [39], etc.

PPO originates from the ideas, explored by TRPO - Trust Region Policy Optimization, its logical predecessor. This algorithm, introduced by Shulman in 2015 is based on the idea of taking small updates so the new policy does not change too much, thus preventing huge updates and gradient explosions within the neural network. It enforced a KL divergence constraint on the size of policy update at each iteration. We control the difference between the behavior policy β and new policy π by controlling the KL value, signifying the difference between the two action distributions, while maximize the parametrized objective $max_\theta \, \mathbb{J}(\theta)$ subject to:

$$\mathbb{E}_{s_t \sim d_\theta(s_t)}[KL(\pi_{\theta_{old}}(\cdot|s_t)|\pi_\theta(\cdot|s_t))] \leq \delta_{KL} \tag{2.18}$$

, where δ_{KL} defines the size of the trust region [30] and is set manually.

TRPO guarantees monotonic improvement over policy iteration, but is difficult to tune and train as it requires calculation of the Hessian matrix. Instead, in 2017 PPO was introduced , which makes use of a clipped surrogate objective loss while retaining similar performance. PPO still makes use of the ratio between old and new policies $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$. We need to put a constraint on the distance between the two policies, but without usage of KL divergence constraint. Instead we use a clipped objective function to limit r.This is a likelihood ratio, interpreted as the measure of similarity between the two policies, restricting which with clipping replaces a more computationally complex KL-divergence constraint.

$$J^{CLIP}(\theta) = \mathbb{E}[min(r(\theta)\hat{A}_{\theta_{old}}(s,a), clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a))] \tag{2.19}$$

The *clip* function puts the value of the ratio between $1-\epsilon$ and $1+\epsilon$. upon using the minimum, we discourage the optimization from increasing policy update to extremes. In addition to the clipped loss, which is responsible for actual policy optimization, there are two more elements to the total loss function - loss of the value function and entropy factor.

$$J^{total}(\theta) = \mathbb{E}[J^{CLIP}(\theta) - c_1(V_\theta(s) - V_{target})^2 + c_2\mathbb{H}(s, \pi_\theta(.))] \tag{2.20}$$

In the formulas above one may notice that PPO makes use of the Actor-Critic framework (as per the original paper [35]). The framework is applied via a multi-head neural network architecture with two models at then end - Actor and Critic, each retaining a number of neurons to produce relevant results. Actor model is responsible for learning the action distribution - our sought-for policy $\pi(s,a)$. Critic model, at the same time, is designated to learning/approximating the value function $V^\pi(s)$. The Critic gives a feedback on how well the model can estimate future reward from current state s - hence the value function loss $(V_\theta(s) - V_{target})^2$ is aimed at minimizing the distance between the current prediction and

the target. Entropy factor is added in order to provide the feedback on the shape of the current action distribution. High entropy loss would signify that actions can be drawn almost uniformly, which would require stronger updates.

$\hat{A}_\theta$ parameter in the formula is associated with the estimated advantage, employed generally by most of the DRL algorithms in one form or another. As per the original implementation, within this research we also make use Generalized Advantage Estimation (GAE), conceptually introduced in [34]. That allows for more stable and guided training. ϵ factor is included as one the parameters of the model.

Without reiterating the original paper, there are several notes that shall be mentioned in relation to the composition of the used neural network architecture.

First of all, it is a feed-forward network with multiple heads, diverging into an Actor and Critic models. The input of the network has the dimensionality of the used state space definition from MDP $\mathcal{S}$ (see 2.2.2), which is then flattened into a 1-dimensional array. This then is plugged into the dense 2-layered network of 256 neurons each with ReLu activation functions. After the second dense layer, there is a split into multiple heads, each of which represents an action $a_{i,j} \in \mathcal{A}$ - in the case of our model it is an SKU-plan group pair. Each action then has either a Gaussian distribution trained (mean and standard deviation parameters) or a discrete softmax layer for the probability of choosing a particular value per the SKU-plan group pair. This distinction depends on either continuous actions (real numbers) are used or discrete, respectively. Such are the representations of the policy π. This part of the network is related to the Actor model. To no surprise, the Critic model shares the dense layers with Actor, but has a different head with regression prediction of the value for each observed state. The dense layers within the network are shared between Actor and Critic, as such allows for the reduction in computational power required for training and increases the speed of training with. Thus, the Actor model outputs a distribution over actions for the current state s, while Critic provides the value function prediction $V^\pi(s)$. Naturally, the size of the Actor softmax layer depends on the restriction condition in equation 2.6 and depends on the set sizes for local warehouses and SKUs: $|J| \times |I| \times \sum_{j \in J} b_{i,j},\ \forall i \in I, j \in J$. Moreover, part of the actions within this restricted space are still unavailable as per availability condition in equation 2.5. The latter is addressed further in section 2.3.2.

Choice of the amount and size of dense layers, as well as activation functions comes from a linear hyper-parameter search, initially based on applied PPO and other DRL architectures from e.g. OpenAI and RLib implementations. [3]

Training methodology

The construction of the training procedure for PPO is comprised of several elements. In Figure 2.6 one can observe the schematic depiction of the training process. It is an iterative process as per PPO algorithm procedure. A fixed number of episodes is run every training iteration in order to gather data for training

[3] https://openai.com/blog/openai-baselines-ppo/; https://docs.ray.io/en/latest/rllib-algorithms.htmlppo

$\mathcal{D}$ - until the dataset is full. In our case the episode length is fixed at T, thus $\lfloor |\mathcal{D}|/T \rfloor$ steps are required to fill the dataset exactly. Then adaptive gradient descent training on for K epochs on mini-batches of size $M \leq |\mathcal{D}|$ from $\mathcal{D}$ of the neural net is applied in accordance with the PPO objective optimization - loss function J^{total} from equation 2.13. One can find the pseudo-code procedure on the training process in Algorithm 2. The procedure at iteration k is stopped whenever it is considered that the algorithm has converged to a particular policy π_k (specific conditions were inferred empirically and are addressed in section 3.4) - but basically the main condition is the minimized fluctuation of the average total reward per episode.

In terms of meta-elements, the procedure is presented as follows:

1. Definition of the model and MDP:

   (a) Define parameters of the problem instance (e.g. sizes of main sets, cost model, lead times, etc.) for the simulated environment

   (b) Use the validation model and obtain optimized base stock levels vector b to complete the definition of the problem instance

   (c) Define the MDP (state and action space sizes and nature, reward function design)

2. Perform PPO algorithm steps (as per Algorithm 2)

3. Once training is considered finished, trained policy π is obtained. Now, it can be used for the *inference*. The latter defines any runs of the simulated environment with the fixed PPO policy (or any fixed method, including the baselines of the research, see 2.4).

From the technical standpoint, the procedure described above is implemented in Python, where RLib framework with Gym wrapper with custom simulated environment are applied for training and consequent inference.

Training a PPO agent in practice does not produce a fixed result on its own. It requires extensive *tuning* - a search in a hyper-parameter space once the key elements of the framework are established. The latter includes the definition of the reward function, state and action spaces - these are related to the MDP definition and are part of problem instance definition for the experiment. These definitions are in turn used for the definition of the architectural choices within the ANN - size of the input and output layers, as well as the structure of the Actor output (either continuous or discrete). Consequently it is possible to determine a set of PPO parameters, which are expected to guide the training appropriately. Most relevant are listed here:

- Value function coefficient c_1. This coefficient allows to control the effect of the value function loss at the full loss function as per

Algorithm 1: PPO training procedure

Result: Optimized $\pi_\theta(a|s)$

initial policy parameters $\theta_0 \leftarrow$ random weights;

initial value function parameters $\mu \leftarrow$ random weights;

clipping parameter ϵ;

training iteration $m \leftarrow 0$;

while *not done* **do**

  $s_0 \leftarrow$ sample initial state from simulated environment;

  $s \leftarrow s_0$ set current state to the initial one;

  while *training dataset* $\mathcal{D}_{\Updownarrow}$ *is not full* **do**

    initialize new episode;

    for *each episode step till* T **do**

      $a \sim \pi_\theta(a|s)$;

      Apply a to the environment and receive new state s';

      $s \leftarrow s'$;

      $r \leftarrow R(a, s)$;

      record (s, a, r, s') into memory $\mathcal{D}_m$;

    end

  end

  $\theta_{old} \leftarrow \theta$;

  for *each update step* **do**

    Sample minibatch of n samples (s_i, a_i, r_i, s'_i) of i index from $\mathbb{D}$;

    Update value function:

    for *each* (s_i, a_i, r_i, s'_i) **do**

      $V_{pred} \leftarrow$ compute target value using temporal difference $TD(\lambda)$ with $0 \leq \lambda \leq 1$;

    end

    $\mu \leftarrow \mu + \alpha_{\gtrsim}(\frac{1}{n}\sum_i \nabla_\mu V_\mu(s_i)(V_{pred} - V(s_i)))$;

    Update policy:

    for *each* (s_i, a_i, r_i, s'_i) **do**

      $\hat{A}_i \leftarrow$ compute advantage with V_μ and GAE;

      $r_i(\theta) \leftarrow \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta\,old}(a_i|s_i)}$;

    end

    $\theta \leftarrow \theta + \alpha_\pi \frac{1}{n}\sum_i \nabla_\theta min(r_i(\theta)\hat{A}_i, clip(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i)$;

  end

end

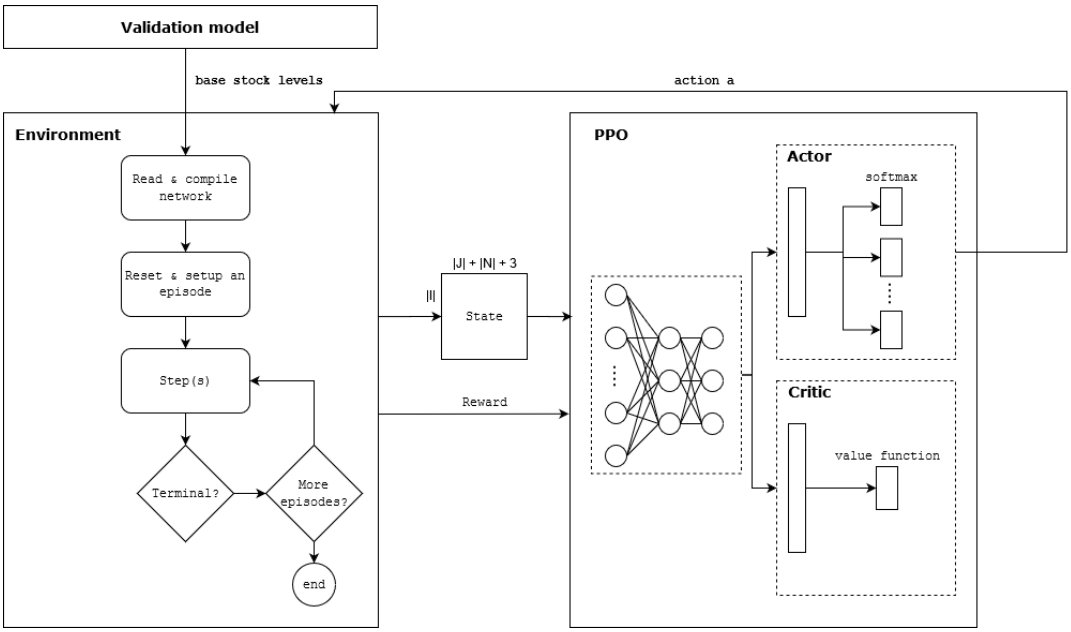


Figure 2.6: Training architecture

- Value function clip parameter and gradient clip parameter. These parameters respectively clip the values for critic model and the gradient update, preventing the model from large updates and keeping the output normalized around smaller values.

- Number of SGD iterations K, train batch size $\mathcal{D}$ and SGD minibatch size M. These parameters are interrelated. With train batch size it is decided how many samples are drawn from the environment interactions, while minibatch size determines smaller batches on which an SGD iteration is performed. The number of such iterations is also determined. High train batch size would lead to a more diverse dataset of higher variance for training, while the size of the minibatch and the number of iterations would determine how much overfitted on the drawn minibatch the neural network will become.One iteration for instance, would signify no overfitting and data re-usage, while high number of iterations may lead to heavy overfit on each data sample causing the agent to constantly re-learn.

- Clip parameter and learning rate. PPO clipping parameter is related to smaller updates as the total loss is controlled. At the same time, the learning rate allows us to regulate the magnitude of the update steps - both parameters solve the speed vs. quality balance.

- Action space nature. It can be either continuous or discrete. The actual action space $\mathcal{A}$ of the problem is discrete inherently, but this definition does not hold in the training process. In training we can use discrete actions, which corresponds to using a softmax layer for action selection at the end of the Actor model. An alternative is the application of the continuous action space. Such means

the training of the parameters for a Gaussian distribution, from which real-numbered actions can be sampled. In order to later applied within the model, this actions are to be rounded.

In its essence the training process for the considered methodology can be defined in two major ways with respect to choosing the hyper-parameters for the model.

- End-to-end training. This approach assumes the hyper-parameters listed above to be fixed and unchanged until training is stopped. Therefore the initial set of hyper-parameters remains fixed through all the training episodes.
- Refining/checkpointing training. This is an iterative approach, where an agent is trained up until it is observed to hit a local minimum. After that, the training is stopped, hyper-parameters are altered in order to get the agent out of the local minimum, and the training is resumed. Such approach requires manual interference with the training. At the same time, it may allow for improved results since the training is guided adaptively. Normally, it can be applied as a linear search across parameter combinations: as the agent becomes more knowledgeable, the learning rate can be reduced, while the value coefficient is increased: then we take smaller updates with a larger feedback from an already well-trained critic model.

Validation of actions

PPO acts upon the defined action space. It draws actions from the defined space, which may results into invalid, unavailable actions - discussed in action space definition 2.2.2. While training, unavailable actions are penalized, as described before in reward design 2.2.2. At the same time, valid variants, based on the actions of the agent, are to be applied to the model. The procedure of doing so is iterative and allocates each SKU in the sequence of locations until the central stock is out. In order to preserve proportions, dictated by the agent, we make a greedy selection here with $max(a_{i,j})$ for current $i \in I$ at each step. Thus we traverse the entire a and make allocations until $IL_{i,CW} = 0 \; \forall i \in I$.

In such manner, we ensure that actually applied actions are always valid, even if the original action was penalized for being unavailable. Similarly, if continuous action space is used, all the real values for actions are rounded to the nearest integer to prevent the supply of incorrect actions.

Other options were also considered, such as using a fixed priority list of warehouses, based on there network position, and random order - instead of the max selection. Neither of these options however sustained empirical tests. Moreover, a fixed list creates biased actions, and random prioritization creates additional, unaddressed stochasticity, preventing the agent from stable learning.

Algorithm 2: Application of valid actions

Result: $a \in \mathbb{A}$

Input:

$a^{\pi} \in \mathcal{A}$;

$a_{i,j} \leftarrow \lfloor a_{i,j} \rfloor \; \forall i \in I, j \in J$;

for $i \in I$ **do**

  while $IL_{i,CW} > 0$ **do**

    allocate $max(a_{i,j}) \forall j \in J$;

    decrease $IL_{i,CW}$ and $a_{i,j^{max}}$;

  end

end

2.4 Research baselines

Allocation decision making is performed in accordance with a particular policy, which can be represented in various ways. While with application of DRL it is represented with an ANN, it is also common to apply various algorithmic approaches to perform the replenishment actions. These methods therefore can be used as baselines for measuring the performance of the DRL agent. There are two main baselines, used within this research: First-In First-Out (FIFO) and NORA. This section gives a brief overlook of the mechanics of the both methods (policies).

FIFO

This is an industry applied approach, which is also widely used in many theoretical models. The models from van Houtum, van Aspert and Reijnen, used as reference models for the model in the paper, is not an exception. The operational basis of this approach is rather simple, as its main principle is to make allocation on the basis of the which demand has come first. In order for the method to work, we have to retain a list with all the incoming demand for the system, as well as keeping the information on the SKU and origin of the demand. And its principle of performance otherwise is tightly coupled with the classical base stock policy.

Algorithmically, it can be described as following:

- Check the ordered list of the demand. If there is no item there yet, choose a random warehouse for assignment. This option is necessary for the starting process.
- Pop the first item and perform allocation from the central warehouse if possible. If it is not possible, then this entry is skipped to the next SKU.
- We proceed until all the stock at the central warehouse is distributed.

FIFO generally tends to push as much stock to the local warehouses as possible, basing its ordering on the previous demand. However, its method is a greedy selection, which is rather unstable in cases, like (1) hectic demand with high variance; (2) several high throughput SKUs in the system; (3) situation with joint optimization when machine groups have capacities - as there we have several SKUs influencing the same machine group, making greedy selection suboptimal. Nevertheless, FIFO can be considered as a relevant baseline within this research, as this method is often assumed for the models in the domain. Assuming FIFO allows for analytical solutions to take place and perform steady state analysis on the system.

NORA

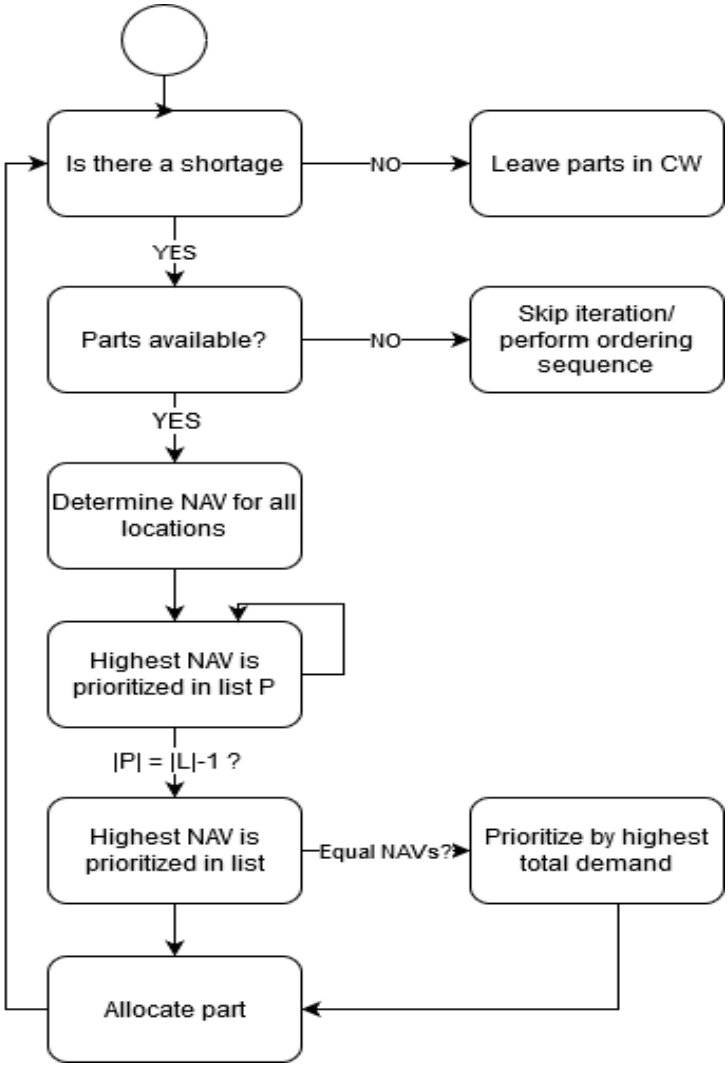


Figure 2.7: NORA algorithmic procedure

As mentioned before, NORA is the current system within ASML that governs the replenishment of spare parts in the entire supply chain and specifically within local networks. In its general form, NORA

essentially operates on the basis of a decision tree. While the system itself is complex and handles multiple flows and operations within stock planning, we consider primarily the part of NORA corresponding to the replenishment decisions. Within this part, NORA algorithm is based on a decision tree, which considers several key decision points with binary choices. These choices are based on the structural decisions (where we are making the replenishment) and state of the network actual ordering metric. This research used an adapted version of NORA algorithm, scoped to handling only the lower tiers of the CSSC - local networks. Within this scope, most of the arbitrary logic of decision trees for NORA was removed as there are no special cases within the local network. At the same time, the concept of at heart of NORA - Non-Availability (NAV) risk - remained in tact, as well as the general iterative procedure, which is presented in Figure 2.7

It is primarily base around the concept of NAV risk - probability of being out of stock during the lead time for replenishment. This risk is quantified for each location and SKU in an iterative manner until the allocation is fulfilled - and serves as ordering metric for locations in the network. The higher the risk is of a stock-out for the location when allocating a particular SKU - the more chances are it ends up to be the first in the order of replenishment. This metric is continuously re-calculated and applied in the procedure. In order to perform this calculations, NORA has to have a good estimate of the optimal base stock level for all the locations in the network, as well as have a good forecast of the demand rates. If either is missing, NORA's performance, especially within the adapted version, degrades quickly. At the same time, good estimates of these parameters allow NORA to make reasonable choices in the anticipation of future due to full information on the demand in the system, making it a strong baseline policy.

Chapter 3

Numerical Experiments and Evaluation

This chapter presents the design and key considerations for the numeric experiments. It aims primarily to address research question RQ6. Consequently, we assess the performance results of these experiments. In order to interpret any results, we have first to describe the performance metrics in 3.1, used for evaluation and analysis as per RQ7 (including the discussion in Chapter 4). Afterwards, we are able to establish the experimental setup in 3.3, while the results of the experimentation are concisely presented in 3.4.

The main research goal of this paper is to explore application of DRL and whether it can be successfully used for addressing the allocation problem within ASML network (and similar spare parts supply chain models). Therefore a single experiment in this context is comprised of:

- Clearly defined problem instance: setting various parameters in order to model a realistic setting for a network, including its structure, size, cost model, lead times, demand rates, etc.
- MDP setting: define the framework structure, in which the agent is to be trained.
- Training setting: hyper-parameters for the DRL agent algorithm.

Experiments include two stages: training and inference. Research baselines, FIFO and NORA, are not trained, as this methods rely on algorithmic decision making in accordance with fixed rules. PPO agent, in contrary, requires extensive training, during which the ANN learns unique rules for allocation decision making. Inference, however, already assumes a presence of a trained/defined policy on PPO. Therefore, a number of simulated runs is performed for FIFO, NORA and PPO under same conditions and model parametrization. Then these are considered to be the results of an experiment on particular problem instance.

Problem instances are to be determined before the experimentation phase can take place, as they de-

termine the structure of the final analysis and comparability of performance between various policies. At the same time, MDP and training settings are subject to adjustments in the course of experimentation. Final results are reported for the settings, which generate most optimal performance results for PPO. However, first we define the evaluation metrics to be used for the performance assessment and analysis.

3.1 Performance evaluation metrics

In most of related literature, the optimization metric is the absolute, against which the success of the DRL agent performance is measured. We can address is as the primary metric. Such position is well-justified, as reward (in our case) directly corresponds to the objective of the optimization problem. While that might be true for an isolated instance of a problem, in a more general sense there are other metrics, which highlight various aspects of agent performance. Moreover, these secondary metrics can be primary optimization objectives in a setting with a different reward design. Here, various additional metrics are explored and assessed for the policies within this research.

Costs

Costs metric was rather extensively covered in 2.2.2, as this is the main metric used in the reward design. Coincidentally, this is the main metric mentioned in the main research question in 1.3. In terms of evaluating the performance, cost model can be used flexibly, as performance of the obtained policy can be studied under different cost models. The latter is possible as the cost metric simultaneously addresses various aspects of policy performance: distribution of stock levels, stock-outs, shipments and service levels for ASML customers.

Here it is important to emphasize how the total operational costs are to be used in the evaluation of the performance. Costs reduction is the objective of the minimization problem, formulated in 2.2.1. It is addressed by maximizing negative reward as per section 2.2.2. Costs are decoupled into holding, transport and downtime factors. While holding and transport costs were utilized previously for evaluation of model in the domain of spare parts IC, downtime costs per SLA are not covered explicitly in the form they are optimized for within this research. Normally, downtime is controlled via hard constraints within the optimization method (r.g. as per validation model from B.1.4). DRL, however, due to the iterative nature of its learning process, may be impacted by hard constraints on experienced downtime and prevent the agent from learning an optimal policy. Therefore, an adaptive factor, directly dependent on the accumulated downtime, is introduced within the cost model.

Nervousness and NAV

As defined in 1.3, nervousness in spare parts planning can be interpreted as the ratio between the consumed spare parts and number of applied shipments. In reality such factor implies the fact that often a

spare part can be re-shipped several times across different locations in the network, and only then consumed. Consequent experimentation showed that such metric is not representative for the performance of the policies, applied in this research, due to the absence of the global tier within the model. Such makes the re-shipments implausible, as we rather use emergency shipments for the supply new spare parts, and lateral shipments occur only once per a spare part. Moreover, there are also no pro-active lateral transshipments, as they require another decision to be made, which is out of scope in this project. Thus, this metric is not eventually used in the evaluation of the agent performance.

A similar logic applies for using the NAV risk (more detail in 2.4) as a benchmark performance metric. We do not use NAV to compare the performance of the different methods due to the the absence of the full picture and one-tier model definition. NAV, moreover, the assessment of NAV risk is used to directly optimize for the stock-outs in the system. PPO optimization in this research is performed for the total operational costs. Instead of NAV risk, we directly quantify the amounts of occurred stock-outs in the model for the analysis of PPO behavior, presented further in Chapter 4.

Downtime (Waiting for Parts)

Downtime within the system is one of the key metrics, distinguishing the spare parts supply chain model from the other inventory models in OR. Downtime of the machines at customer sites is directly related to the level of customer satisfaction and SLA performance. As mentioned before, we use the notion of Downtime Waiting for Parts (DTWP) ,incurred in the system. As for the DRL method, DTWP participates in the optimization within the problem both directly and indirectly. The indirect effect is applied across all the policies and experiments, as each model and simulation run employ base stock levels, obtained from the validation model. These levels, in turn, were optimized with the DTWP constraint (more details in B.1.4). Direct optimization for the caused downtime comes from the reward designs R_{hyb} and R_{mix}. Therefore, downtime, together with being the primary SLA performance metric, shall be considered within the analysis of the results.

Namely, we are interested in the average total downtime per episode, as well as downtimes per a single machine group. It corresponds to the efficiency of the allocation policy in terms of the contracts. Moreover, the average downtime duration is of interest for analysis as such measurement relates to the average duration of a stock-out.

Demand satisfaction

There are various ways to account for the demand statistics. It is possible to account for stock-outs (as per section above), which occur in the case, when demand is not satisfied from the local warehouse. These can be counted along the progression of the simulation episode, and naturally are calculated as the sum of lateral and emergency shipments over the episode. As per [32] and [41], we segment total satisfied demand $\forall i \in I \; \forall j \in J$ into:

- $\alpha_{i,j}$ is the fraction of demand satisfied from the local stock in j for all $n \in N_j \subset N$
- $\beta_{i,j}$ is the fraction of demand, satisfied from lateral transshipments from the other warehouses in the network.
- $\theta_{i,j}$ is the fraction demand, satisfied from emergency shipments.

Logically, $\alpha + \beta + \theta = 1$ shall always hold for all $i \in I$. These metrics are directly used for the validation of the simulation model (more detail in B.1.4). Such separation allows comparing performance of the allocation decision policy with respect to different local warehouses. Moreover, it highlights the general effectiveness of the system in handling stochastic demand. By observing how the demand was satisfied in an episode (on average), it is possible to establish general behavioral trends for the DRL agent and compared methodologies.

3.2 Analysis scenarios

Having considered several evaluation metrics, it is important to clearly establish the main directions for the analysis of the allocation decision problem. These directions are the basis for formulating problem instances, used in the numerical experiments. Moreover, these scenarios define, in which context we compare the performance of the policies, namely - DRL agent and baselines' results.

3.2.1 Scenario 1: Problem complexity

Within this scenario, we consider a number of instances with a linear increase in complexity. Namely, the dimensionalities of state $\mathcal{S}$ and action $\mathcal{A}$ spaces are increased. Moreover, additional stochasticity can be introduced to the problem in order to make the search space of the problem less sparse. The latter means introducing more states with information, useful for the learning of the agent. In such a case, it is possible to expect that the baseline methods would not have any computational complications, but can experience decreased performance due to more complex optimization, especially with the of increase in the number of stochasticity sources. Such expectation is also based on the fact, that neither of the baseline methods applies a joint optimization procedure for all the SKUs in the model. PPO policy is expected to have problems with training and may experience premature convergence to local optima as the problem becomes more complex. At the same time, the choice of PPO (as the applied DRL methodology) was partially motivated by the expected ability of the method to cope with the rapid growth of the state and action spaces.

This scenario mostly contributes to the exploration of the PPO applicability in a real-life business setting and potential scalability of the problem definition, required for the real world application.

3.2.2 Scenario 2: Optimization objective focus

Within the definition of the MDP, provided in section 2.2.2, there are multiple reward function designs, which correspond to different (augmented) optimization objectives. Therefore, it is relevant to observe which formulation can lead to different policies and impact the success of the DRL application. While baseline methods are not expected to experience a change in the optimization objective, such a change shall impact the performance of our PPO agent. Primarily, it is expected that when PPO optimizes directly for the downtime in the system, its operational costs can have even larger improvements in relation to the performance of the baselines. Such expectation is motivated by the fact that neither FIFO, nor NORA optimize directly for the DTWP. As per the validation model for the simulation, these methods rather rely on optimal base stock policy, obtained with the consideration of DTWP constraints.

3.2.3 Scenario 3: Diverse demand patterns

In relation to the test beds, presented in [42] and [41], we formulate several instances, where SKU with a low aggregate demand rate is compared to the SKU with a high throughput. Expensive spare parts for complex machinery are often treated as slow movers, while there is also a case for more common, replaceable parts - fast movers. Also, we define a ”normal” demand pattern, where the demand rate is referred to as even. Such qualitative distinctions are created for the comparability and contrasting of the situations under diverse steady states of the system. Additionally, the idea here is to study the case, when the system has both types of demand patterns and observe what are the differences in handling these two groups of SKUs policy-wise.

Studying various demand settings does not simply mean increased stochasticity, but allows studying the impacts of stochasticity on the process of PPO learning. There are several directions, in which comparisons can be made between various demand rates, since these are defined per an SKU-plan group pair:

- First and most straightforward way is to determine a single demand rate for an SKU. In that sense, a slow moving SKU would have the same low demand rate across all plan groups within the model. The same would apply for high and even demand rates per SKU.

- We can designate varying patterns of demand within one SKU for different plan groups. Then, some local warehouses in the network would have low consumption for this particular SKU, while the others would have a heightened demand. Thus, we would compare slow vs. fast mover demands within one item.

- Consequently, we can also compare the slow vs. fast patterns between two different SKUs.

Such cases are to be analyzed in relation to the implications for demand satisfaction in the model.

3.3 Experimental design and setup

After defining main directions of analysis and evaluation in the scenarios above, we shall restrain the set of parameters within the problem instances. Since the allocation problem at hand has a large number of hyper-parameters, which can be tuned across multiple options, it was decided to clearly outline a subset of those to be explored, with other parameters are to be fixed across all the problem instances. Such design secures comparability of the results across the defined instances. These parameters were decided to be the most impactful and relevant for the problem exploration.

- Number of local warehouses $|J|$. In accordance with the model, there is only one central warehouse in any instance of the problem. However, it is possible to expand the size of the network and simply increase the number of local warehouses. This number directly influences the dimensionality of the problem, as the parameters related to locations (stock levels, cumulative backorders and, indirectly, number of machine groups) are included in the analyzed state space. It also increases the search across the action space. Basically, the growth of the locals heightens the complexity of the problem, but at the same time allows for increased flexibility for the DRL agent.

- Number of SKUs $|I|$. This parameter also directly influences the dimensionality of the state and action spaces, and it the same time it is directly related to the complexity of the problem. However, unlike the size of the network, the amount of SKUs is not imperative to the solvability of an instance. Potentially, we can consider each SKU within the model separately, although this research concentrates on the joint optimization across the SKU set I.

- Demand rate patterns $\lambda_{i,n} \forall i \in I, n \in N$. This is a complex parameter, as there is a certain room for maneuver in defining the demand rates per problem instance. In accordance with extensive experimentation and academic literature (e.g. [10]), demand rate often directly contribute to the tractability of the problem. Here, demand rates are based on the test beds from [42] and [41]. The main purpose of changing this parameter (as per Scenario 3) is to study qualitatively a set of situations to be handled by PPO. Therefore, we study slow (low λ) and fast (high λ) demand rates in relation to SKUs and plan groups, as well as consider even (normal) λ for the instances of higher complexity.

- Supply lead times distribution. Additional source of stochasticity can come from the random supply lead times. This uncertainty can additionally complicate the training process, as it would tend to affect the learning of the true value function in the Critic model.

First two parameters relate to the analysis within Scenario 1, as well as the variability in supply lead times. Demand rate patterns corresponds to study within Scenario 3, while Scenario 2 is addressed by the changes in the MDP setting.

Instances	# locals	# SKUs	Demand rate patterns	Variable lead times
1	2	1	SKU1: low	no
2	2	2	SKU1: low; SKU2: high	no
3	4	1	SKU1: low	no
4	4	2	SKU1: low vs. high; SKU2: even	no
5	4	2	SKU1: low vs. high; SKU2: even	yes
6	4	4	SKU1: low ; SKU2: low vs. high; SKU3: even; SKU4: high	yes

Table 3.1: Problem instances for experiments

As far as the MDP definition goes, the state space is constrained as per 2.2.2. Same section contains the following options considered in the MDP setting;

- Action spaces nature: can be either discrete or continuous. Such distinction brings not conceptual, but technical changes as per section 2.
- Reward function design: $\{R_{int}, R_{mix}\}$. As mentioned in 2.2.2, R_{hyb} is not used in the final experiments, as R_{mix} supplements its ideas.

3.3.1 Experiment parameters

Finally, on the basis of the adjustable parameter subset and analytical scenarios, 6 instances of the spare parts allocation problem are formulated.

Permutations of the parameters from the previous section are presented in the table below. The rest of the model parameters are fixed and provided further in the text.

Some of the parameter permutations require additional explanation. It is important to highlight the demand patterns relations. These instances are based on the test instances from Chapters 5 and 6 from van Houtum and Kranenburg [42], as well as van Aspert [41]. In accordance with their definitions for demand rates, the ratio between "low" and "high" demand rates is 1:7, where low is taken as $\lambda_{low} = 50$ parts a year; $\lambda_{high} = 350$ and $\lambda_{even} = 150$ parts a year. All the instances are to be trained with T as 360 days. Longer terms were also used for explorative reasons (such as $T = 400$ or $T = 720$), but final results are not presented in the report.

In general, realistic definition of the cost model is established as an approximated ratio: $1 : 8 : 24$ for $c^{lat} : c^{em} : c^{dt}$ for all SKUs and customers. Such rule maintains a realistic depiction of prioritization

Name	Type	x coordinate	y coordinate	Lateral order
CW	central warehouse	0.0	0.0	-
WH01	local warehouse	3.0	2.0	WH02,WH03,WH04
WH02	local warehouse	2.0	2.0	WH03,WH04,WH01
WH03	local warehouse	2.0	1.0	WH04,WH01,WH02
WH04	local warehouse	1.0	4.0	WH02,WH01,WH03

Table 3.2: Network structure and lateral order

between various cost parameters. ch are as a rule 1/100 of c^{lat} per unit, as these are estimated in large quantities at each time step t.

Lead times are distributed as follows. For the lateral lead times we use the structure of the network. Within the network we defined a grid, on which warehouses are placed as per Table 3.2. One can also find the respective lateral order between the warehouses. The only change is that for instances with 2 locals, logically each of them would be in the list for laterals, without WH3 and WH4. Then each lead time is based on the Euclidean distance between the two locations and is measured in hours by multiplying the distance by 2.

Longest lateral lead time l^{lat} relates to l^{em} roughly as 1 to 4. l^{em} is fixed at 2 days for each SKU, while supply lead times are 7 times longer, depending on the SKU in question. Variability of the lead times was modeled simplistically with $uniform(5 \cdot l^{em}, 7 \cdot l^{em})$. These parameters can be a subject for future research.

Having defined the changed and fixed parameters for the model, it is also important to clearly establish MDP settings for the experiments. Each defined problem instance is an experiment. Therefore, the final set of experiments is of size $6 \times 2 \times 2 = 24$ in the form of $instance \times reward \times action$. Each instance had a separate agent trained for it with respective search across various hyper-parameters. When it was considered that the training has converged and the result is accepted, we perform the inference on the problem instance with the usage of FIFO, NORA and PPO agent across a selected sample size. It is determined through the minimized standard error $SEM = \frac{s}{\sqrt{n}}$, where s is the standard deviation of the sample and n is the sample size. It was defined that a sample size of 10000 is sufficient to obtain acceptable non-intersecting confidence intervals of 95% on NORA and PPO for the considered instances, so this sample size was used for the benchmarking of the policies. Results of these experiments are provided in the next sections with some information on the parametrization of the experiments in terms of PPO hyper-parameters.

Policy	Instances	1	2	3	4	5	6
FIFO	Mean	59.07	237.21	275.01	875.73	1188.04	1578.34
	St. dev.	10.91	23.83	140.17	131.01	193.25	276.82
	95% c.i.	0.21	0.46	2.75	2.57	3.78	5.42
NORA	Mean	23.48	125.89	222.38	231.44	266.29	368.81
	St. dev.	2.62	17.56	28.27	28.06	31.07	45.21
	95% c.i.	0.05	0.34	0.55	0.54	0.60	0.69
PPO	Mean	16.78	114.68	158.17	233.85	271.04	389.01
	St. dev.	1.12	33.46	38.91	47.84	46.16	59.13
	95% c.i.	0.02	0.65	0.76	0.93	0.90	1.15
Action space		discrete	discrete	discrete	continuous	continuous	continuous

Table 3.3: Average total reward with R_{int} per episode per experiment

3.4 Performance results

This section addresses the main performance results of the research, against which we measure the success of the DRL application in addressing the cost-efficient allocation decisions. Therefore, here we concentrate on the reward function results, which is based on the primary metric in this research - total operational costs (as per section 3.1).

Table 3.3 presents results across multiple instances of the problem for R_{int} reward function design. As reward is represented as costs, the reward shall be minimal - and this is how the table shall be read.

First evident result is that FIFO shows inferior performance across all the experiments. The comparison between NORA and PPO then is more interesting. Simple instances of the problem (1-3) show PPO to outperform NORA by 9% on average (and 22% average performance on Instance 1) in terms of costs reduction. More complex instances with more stochasticity (4-6) demonstrate the performance of PPO to be on par with performance of NORA, being slightly below the baseline. The results for more complex instances are based on a setting with continuous action space - as the results with discrete space (in training) for these instances were inferior. In addition, one can refer to the comparative results in Figure 3.1, where results for PPO and NORA are presented in relation to FIFO, as the latter always performs worse for each instance.

Table 3.4 shows similar setting of results, but for the R_{mix} reward design, which also directly optimized for the downtimes through contract penalties. Unfortunately, we were not able to obtain any PPO agent (except instance 1, which the simplest setting in problem size) being able to outperform the main baseline - NORA. The difference between rewards (also in terms of magnitude) for FIFO and NORA is related to

Policy	Instances	1	2	3	4	5	6
FIFO	Mean	432.67	482.40	673.87	1354.44	1519.92	2348.33
	St. dev.	24.50	46.63	120.04	169.69	272.88	384.93
	95% c.i.	0.48	0.91	2.35	3.32	5.34	7.54
NORA	Mean	176.12	210.41	302.68	572.65	589.83	939.51
	St. dev.	4.72	29.21	44.20	71.31	102.79	113.18
	95% c.i.	0.09	0.57	0.86	1.39	2.01	2.21
PPO	Mean	164.99	258.38	405.21	631.90	639.62	1334.78
	St. dev.	5.07	49.24	66.18	85.57	120.06	228.52
	95% c.i.	0.09	0.96	1.29	1.67	2.35	4.47
Action space		discrete	discrete	discrete	continuous	continuous	continuous

Table 3.4: Average total reward with R_{mix} per episode per experiment

the fact, that the inference in Table 3.4 was also performed with downtime penalties.

The results presented are the results for converged PPO agents. The convergence is however a relative term, therefore, the following criteria were used to assess whether the agent has converged to a particular policy or not.

- Convergence on miss factor: shows that the agent learned valid actions. Median miss factor over all the episodes in the training iteration is assessed here and values below 3 for several iterations in a row show that the agent has managed to learn available actions.
- Acceptable level of the explained variance of the value function: depending on the instance, a value over 0.4 can be considered acceptable. It means that the model can relatively well predict the value of the state in terms of future rewards, if the sample size is acceptable. The value was inferred empirically upon
- Unchanged or degrading behavior over last 10-30 steps, depending on the instance complexity. Specifically, this can be reliant if at the later stages of training there can be higher values for SGD iterations K used.

One of the main results here was the inability of PPO to learn properly when optimizing for the terminal reward setting with R_{hyb}. In order to combat that, the mixed reward function R_{mix} was introduced. However, while these results presented an improvement, as per Table 3.4, one can see that PPO did not manage to outperform NORA in that setting. The latter is partially explained by the heightened variance of rewards and larger magnitudes of rewards with the down time optimization. Also, in accordance with

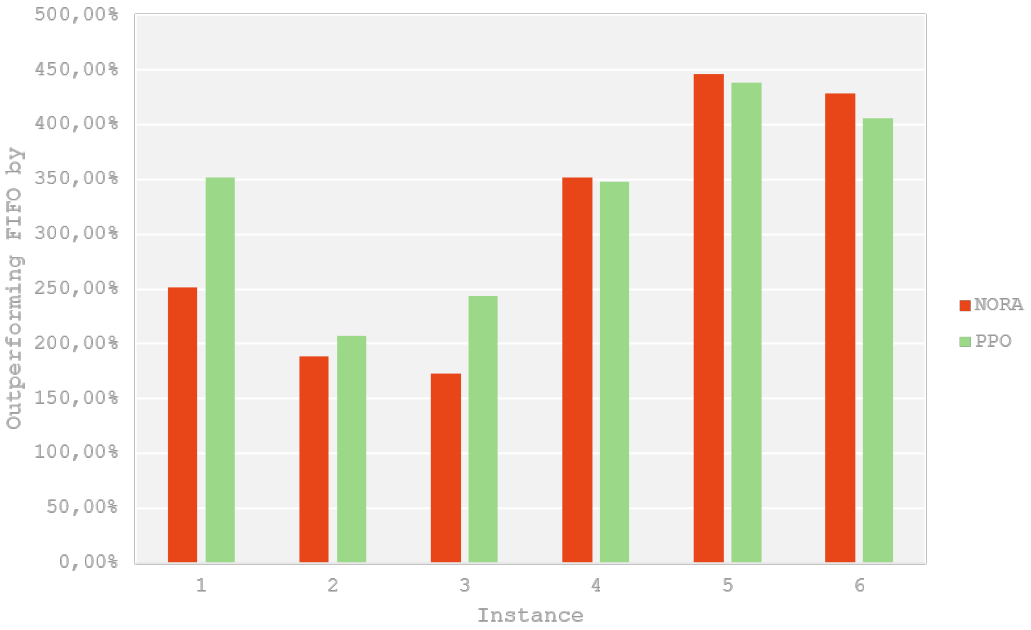


Figure 3.1: Comparing by how much NORA and PPO outperform FIFO per instance with R_{int}

Scenario 1, we assess how PPO agents handle the growth of the problem complexity. The general trend is conclusive: PPO scales with the degradation of performance. Even more complex instances show decrease in PPO average total reward per episode. A less conclusive, but interesting point for further research is the increased performance degradation with the expansion in the number of SKUs $|I|$. This is explainable as it not only grows the state and action space sizes, but also rise stochasticity in the model, while the growth in the number of local warehouses seems to be more manageable. More SKUs automatically means more randomness per each local warehouse in the model and an impact on the exogenous supply calculations. Increase in $|J|$ potentially can be mitigated partially by lateral transshipment mechanics, allowing to cope with the additional demand in the system.

Baseline performance

Two algorithmic approaches are used within this research. Both make the allocation decisions structurally with respect to a particular metric, used for prioritizing the replenishment per each local warehouse in the network. Therefore, these methods have interpretable structure, which can explain their performance, to be further compared with the performance of the PPO method. PPO learns such structure and it is not directly interpretable upon the application of learned policies.

FIFO method simplistically prioritizes replenishment of the immediate most frequent demand. Such strategy assumes no planning for the future and is highly susceptible to potential stock-outs, requiring lateral or emergency shipments. As the result, FIFO tends to perform approximately twice as bad as

NORA on average. NORA uses demand information to make predictions on the risks of potential stock-outs and tends to minimize them by respective prioritization. Unlike FIFO, NORA tends to have less overall variance in the total incurred operational costs. The latter can be explained by the fact that FIFO changes its choices with each new occurring event, while NORA tends to maintain most of its decision sequences stable, due to NAV risks experiencing less fluctuations in the overall amount of demand events per each supply lead time. Also, if we compare instances 2 and 3, it is noticeable that FIFO struggles with both increase in the number of SKUs (instance 2) and increase in the number of local warehouses (instance 3). NORA at the same time manages rather well the growth of the SKU set size $|I|$, while the growth in $|J|$ leads to a substantial growth in the cost variance and average total costs per episode. Such comes directly from the ability of NORA to mitigate additional demand rates, as it has the access to this information. Increasing the amount of warehouses at the same time makes the prioritization list of local warehouses longer and increases a risk of potential assessment error in case NAV risks for all warehouses are very close to each other.

Consequent baseline comparisons are mostly performed between NORA and PPO due to clear disadvantage of the FIFO method.

Variance in total costs

While the magnitude of the rewards changes, most of the instance PPO policies exhibit similar behavior, in terms of relative performance between the baselines. From the performance point of view, NORA retains stable numbers and decreased variability. PPO agents per instance do have quite diverse training processes and higher variance of average total reward per episode in general: some can be trained successfully in an end-to-end fashion, while the others have to be gained with the refining approach 2.3.

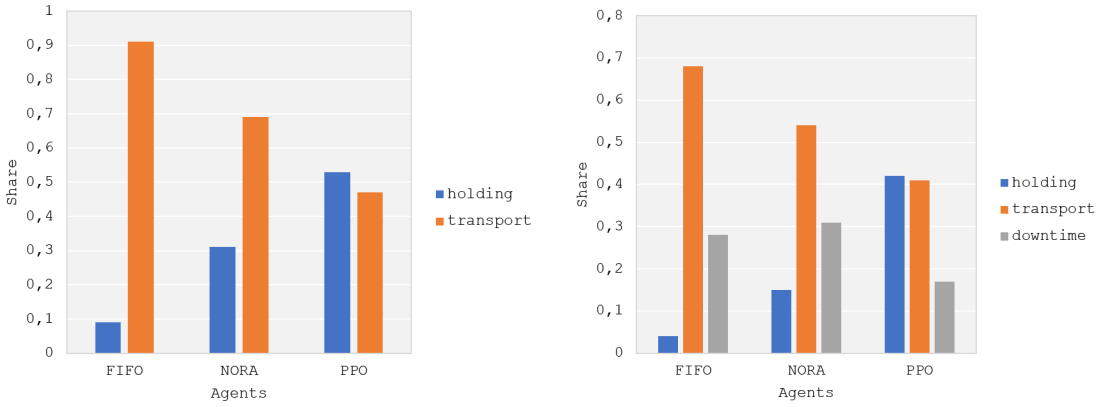
With interim reward design R_{int} the standard deviations ratio between NORA and PPO differ for instances 1-3 and 4-6. First group of instances have an average of 23% higher standard deviation for PPO over NORA, while for the second group this gap is at 62%. Such difference is a clear effect from the growth in complexity per instance. Moreover, this can be the result from applying continuous actions, while training the PPO agent for instances 4-6 due to the absence of acceptable results with discrete actions. At the same time, reward design R_{mix} with the additional downtime constraint showed an increased standard deviation for PPO in relation to NORA at 41% and 46% for instances 1-3 and 4-6 respectively. Taking into consideration, that this design did not allows PPO to outperform NORA in general, such stability is not a valuable result and can be explained by a general growth of reward variance for all the methods. This variance grew due to a more diverse reward distribution with additional cost factor introduced.

Operational cost factors distribution

As per Figure 3.2, one can observe the overall relationships between the cost factors per reward function. Both reward functions show a very clear trend across the applied policies. FIFO attempts to always push

the stock to the most frequented demand locations. Such behavior drastically decreases holding costs, but results in high numbers of additional shipments, leading to the increase in transport costs share in the total costs. NORA is able to plan ahead, but still tends to use more shipments then PPO (more detail in consequent chapter, section 4.2). R_{int} and R_{mix} reward designs do not have significant differences in the distribution of holding and transport cost, and the introduction of downtime associated penalties in R_{mix} still maintains this ratio in tact across methods. The only additional insight is the fact that PPO manages with decreased downtime penalties in relation to the other costs types.

Now, an important note to the interpretation of the aforementioned results and Figure 3.2. We show here relative shares between the cost factors per method. Therefore, while, for instance, PPO does have decreased downtime and transport shares in comparison with NORA in sub-figure 3.2b, in absolute numbers PPO still produces inferior results to NORA with this particular reward setting. With the same logic, while PPO transport costs share is lower than the respective share for NORA in sub-figure 3.2a, absolute values show even a larger difference. Such can be cross-referenced from the overall results tables in the previous tables 3.3 and 3.4.



(a) R_{int} reward function

(b) R_{mix} reward function

Figure 3.2: Distribution of costs factors per agent (average over instances 1-3)

Summary of performance results

It can be safely concluded that using interim reward design and instance-dependent training approach (in state and action definition, training paradigm, hyper-parameters choice) PPO manages to acceptably outperform NORA method by 9% on average and largely outperform FIFO in terms of the operational costs for simpler instances (1-3) of the problem. This result is considered to be an acceptably converged. Complex instances of the problem (4-6) result in a slight edge for NORA in terms of the operational costs (nearly 1.5%), while PPO is able to perform at nearly the same level as the NORA benchmark. Possible explanations and interpretations of such can be found in the next chapter of the report.

Chapter 4

Analysis and Discussion

This chapter directly follows from the previous chapter and is aimed at analyzing the obtained experiment results in order to gain useful insights, unrelated to the primary performance metric. Most of the analysis and discussion here concentrates on the interpretability of the experiment results. These insights, in general, can be grouped into two categories. First category relates to potentially interesting observations, collected during *the training process* of the DRL agent. These observations can be used for further research ion the direction of the thesis and contain practical interpretations of the aspects of successful training. Second category includes all the insights regarding the *behavior of the DRL agent* and comparative performance for secondary metric from 3.1 in relation to the baselines. This chapter extends and the answers on research sub-questions RQ6 and RQ7.

4.1 Training process analysis

Insights into the training process of a DRL agent on an inherently stochastic problem with sufficiently complex action space is one of the contributions of this paper. Alongside with the defined MDP, there are several factors, contributing significantly to the obtainment of the benchmark performance.

First, we managed to use the miss factor mf, introduced in 2.2.2, for tracking the progress of the agent training. Applying a flexible factor with changing penalty on invalid actions allowed for a guided learning of action availability for the agent. It is important for the agent to properly learn valid action performance before it actually can optimize for proper allocation policy with respect to the long-term planning. Miss factor, employed for that eventually served as one of the metrics, signifying success of the training: if we observe a convergence of the miss factor to small values, it is possible to say that the agent has learned valid actions, which is demonstrated in Figure 4.1.

Another important values to observe during the training process are the ratio between policy and value function loss and the explained variance of the value function. The latter gives the feedback on how

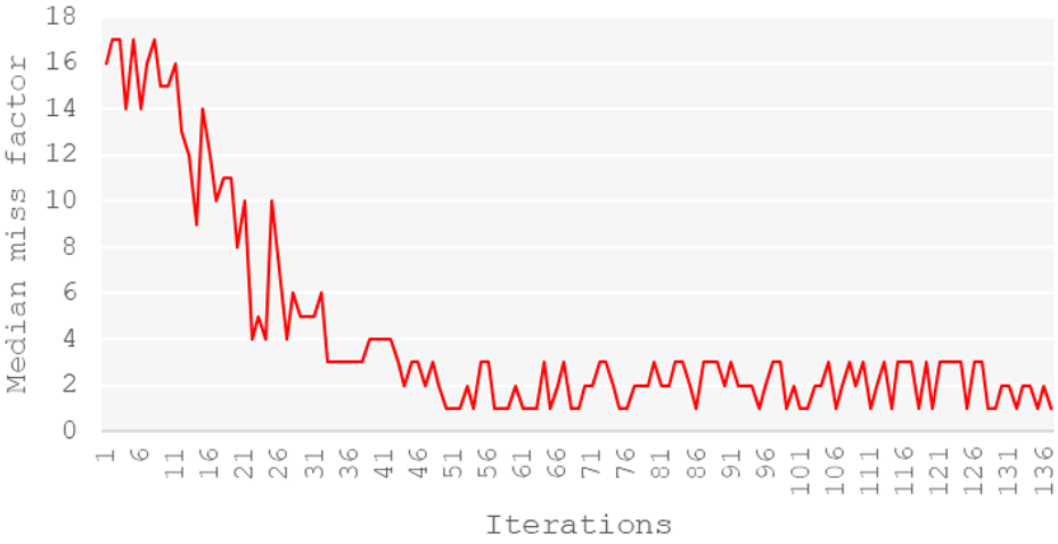


Figure 4.1: Progression of median miss factor in training through episodes

well the value function can make predictions on the basis of data it observes. Now, this factor does not necessarily converge appropriately due to diverse state space caused by the inherent stochasticity of the problem. but it can serve as a flag point: empirically, we can observe that training can be considered successful if the explained variance reaches values of 0.4 and higher. As PPO procedure forbids/prevents extreme updates, we can expect the explained variance to indicate an acceptable performance level, reached by the Critic model. Further reductions of this factor are normal, as fluctuations in the sample data may cause principally different training data supplied to the model. It is also shall be noted that this factor is only usable with a sufficient training data size. Otherwise, a large gradient update may occur at any moment of the training, causing the value function to heavily overfit on a small part of the state space (current training dataset $\mathcal{D}_k$) and then being unable to explain the other training samples well.

It is also quite evident, that the reward design R has a decisive effect on the success of DRL optimization and performance results. It also affects the efficiency of training and addresses various challenges to the stability and convergence of the learning. One of such challenges was the regulation of the *convergence to local optima*. The latter is a persistent problem, and is widely recognized as one of the main downsides of PPO algorithm alongside with poorly controlled exploration. In order to combat the convergence to the local optima, there are several tools available. With the end-to-end training paradigm a good balance between learning rate α, batch sizes and number of SGD iterations have to be reached. It is also advised to designated an average value function loss coefficient with relatively low clipping parameter on the value function. The latter prevents a more quick convergence of the critic model, but results in the higher stability at the mid and end stages of training.

4.2 Behavior analysis

In this section we address the secondary metrics from section 3.1 apart from operational costs, as these are considered in the review of performance results in section 3.4. Here, a number of interesting insights, gained from analysis of scenarios from 3.2 and experiments 3.3, are presented for demand satisfaction, stock-outs and downtime metrics. We mainly use reference experiments to showcase patterns in PPO behavior. These, with some relaxation, can be generalized for most of the trained policies within this research. All of the data considered here relates average and/or cumulative values per episode of a simulation run. We also use single representative episodes to show general behavior patters. Representative episode is chosen as the one with average costs performance for PPO with corresponding performance of NORA and FIFO exactly the same demand events.

4.2.1 Analysis of stock-out situations

First, it is important to address the stock-out situations and shipments statistics. As per Scenario 1 in 3.2.1, the increased complexity of the problem causes different amount of lateral m^{lat} and emergency m^{em} shipments to occur, where the total number of stockouts is the sum of cumulative numbers of shipments at the end of the episode $m_T^{lat} + m_T^{em}$. Consequently, we still distinguish between the stock-outs, replenished via a lateral transshipments, and the ones, satisfied with an emergency shipment. These two types of stock-outs contribute differently to the cost metric, influencing the optimization process. Here we find a conceptual contrast between NORA and PPO methodologies. NORA directly optimizes for the minimization of stock-outs, but does not receive a direct feedback on the costs, associate with each individual (potential) stock-out. At the same time, PPO is able to make such distinction with the reward function design and its transport costs factor (detailed in 2.2.2).

One would observe that instances of higher complexity have a growing number of emergency shipments m^{em} across all the applied policies - PPO, NORA and FIFO. While the growth of absolute values is logical, we cannot detect a conclusive change in the ratio $m_T^{lat} : m_T^{em}$m as m^{lat} grows respectively. Increased state space complexity in 3.2.1 for Scenario 1 does not lead to principal changes in the progression of cumulative shipments for both laterals and emergencies from instances 1 to 6. At the same time, when we introduce different demand patterns (compare slow vs. fast movers Scenario 3) in 3.2.3, we observe an increased relative share of emergency shipments in the range from 5 to 15 % across observed policies on average. Such result, however, does not contribute to the distinguishable features of PPO policy, but rather states the fact that all methods struggle with the growth of problem complexity and stochasticity. These can additionally be explained by the higher throughput demand arrivals. Consequent analysis showed that the plan group with highest aggregate demand rate tends to be the cause of the majority of emergency shipments, e.g. $\sim 60\%$ on Instance 2 for PPO policy. Comparable results are obtained for NORA as well, while FIFO tends to have increased emergency shipments for the groups with low aggregate demand. Such

behavior is expected for both of the algorithmic methods. NORA attempts to adapt towards anticipated risks and is able to manage low demand rates well with occasional, but impactful non-availabilities. FIFO concentrates on satisfying the high demand rates, causing excessive amounts of emergencies there.

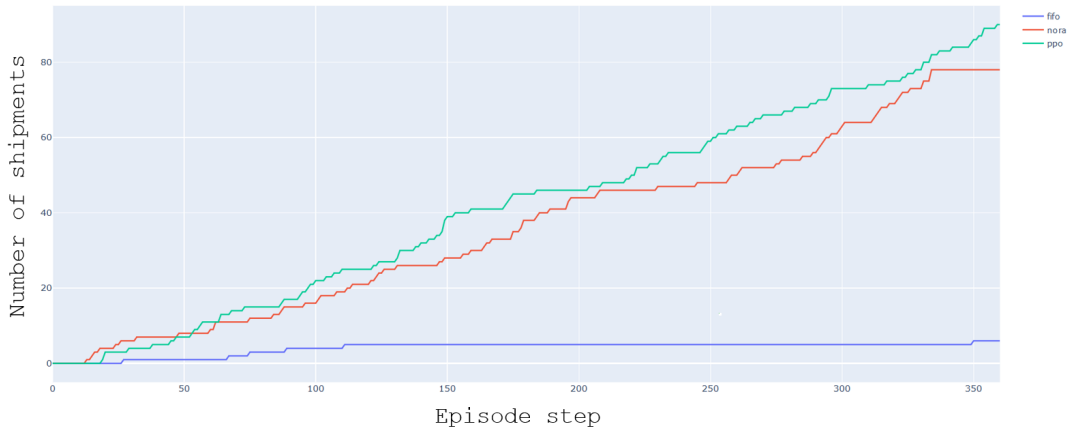


Figure 4.2: Cumulative lateral shipments across episode on Instance 2

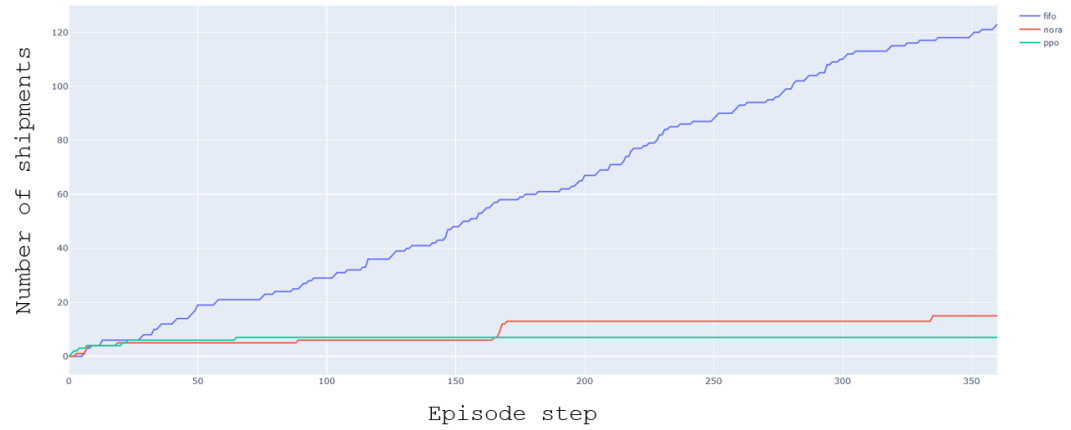


Figure 4.3: Cumulative emergency shipments across episode on Instance 2

In terms of Scenario 2 and alternating reward designs, the model showed no interpretable differences, except for different values. Figures 4.2 and 4.3 showcase comparison between the methods in the dynamics of a representative episode. In the figures, one can find PPO to manage with fewer emergency shipments across the episode without any strong spike, which it compensates with higher number of lateral shipments. NORA has one definitive spike in emergency shipments number on this graph. The spike constitutes to the near-simultaneous concentrated demand arrival, which NORA was not prepared to handle. Roughly at the same time during the episode there is a spike of lateral transshipments for PPO, meaning that the agent already had enough stock allocated in the network, which just had to be allocated correctly.

Another consideration within the problem is the distribution of stock-out situations in the system across a large sample of episodes. While the stock-outs distribution change is not interpretable directly by the state complexity and joint optimization (Scenarios 1 and 2), we can interpret them from comparing SKUs with different demand patterns in Scenario 3. For simplicity of representation, we use the instance 2 to illustrate the difference between a slow and a fast mover, both optimized within one system. In Figure 4.4 one can observe the performance of the three policies for both SKU1 - a slow mover - and SKU2 - a fast mover. Expectedly, FIFO manages the fast mover SKU1 more efficiently than the slow mover, due to the greedy selection of a more frequent demand source. NORA controls the stockouts for both SKUs even more efficiently, as it has a direct access to the information about the demand rate and can optimize with the stockouts with this information. PPO, however, while outperforming FIFO, seems to have higher variance and decreased performance in relation to NORA - especially with the fast mover. It can be explained by the fact that fast mover demand more often leads to the states, rarely visited by the agent in training. So, unlike FIFO, PPO is more comfortable with operating in a sparse state space during search, and specifically optimize for slower demand rates.

Nevertheless, in order to make a simple conclusion here, PPO allows for more stock-outs than NORA, on average. However, as reviewed, in the beginning of this section, the quality of this stock-outs tends to differ. Overall number of stock-outs $m^{lat} + m^{em}$ for PPO tends be higher than for NORA due to a faster growth of m^{lat}, while for NORA the ratio $m^{em} : m^{lat}$ growth due to increased emergency shipments.

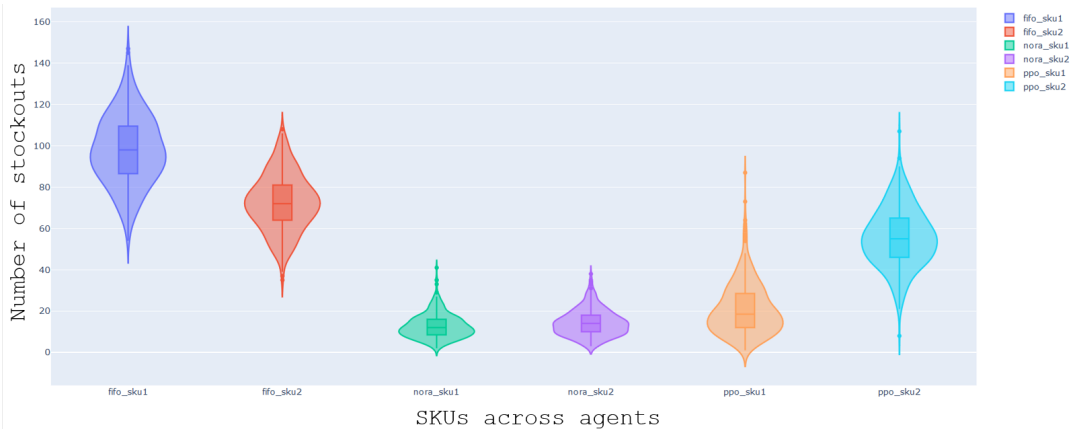


Figure 4.4: Distribution of stockouts in Instance 2

4.2.2 Analysis of the downtime

Logically, after the analysis of stock-outs in the model, it is interesting to observe the behavior of the applied policies. As mentioned in section 3.1, downtime metric is an important characteristic in this research, although not the primary one. An addition reason for that is the extraction of inconsistent results for the downtime measurements in the experiments.

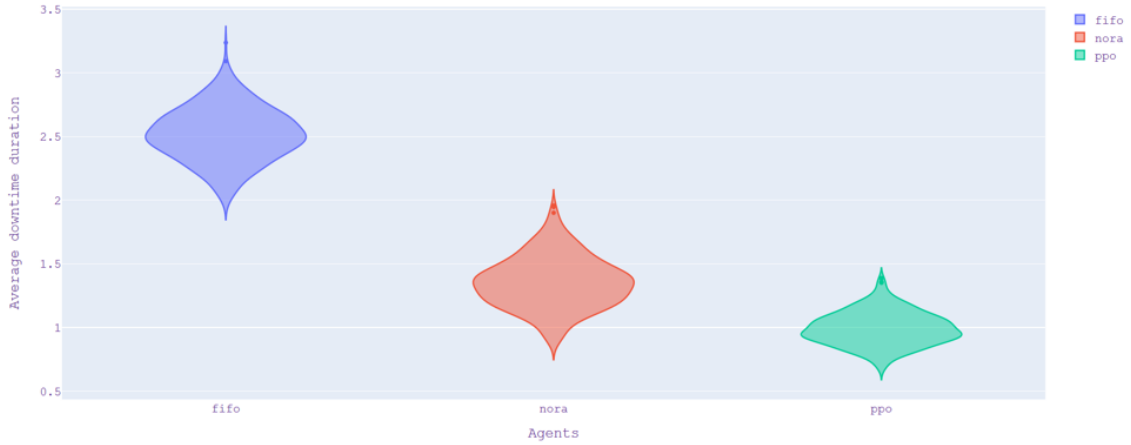


Figure 4.5: Average downtimes per day on Instance 2

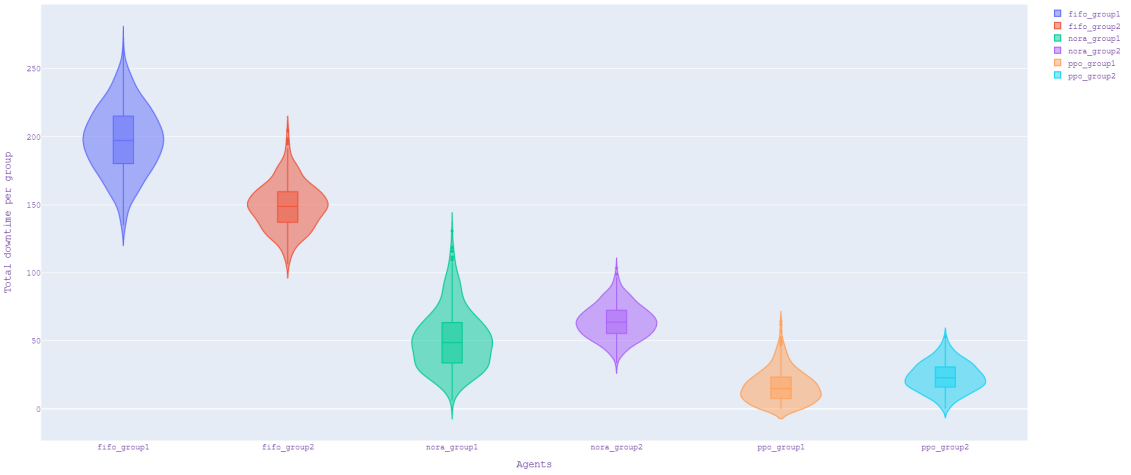


Figure 4.6: Total downtime per machine group on Instance 2

In general, as per section 3.4 we observed that the experiments with R_{int} reward design managed to show superior performance of PPO in terms of costs, while the mixed reward setting R_{mix} did not allow PPO policies to generally outperform NORA baseline. However, there was an expectation that since within R_{mix} we optimize for the downtime costs directly, the overall DTWP would be reduced. However, the downtime results with R_{int} were also more consistent and generally showed on the instances 1-3 PPO managed to decrease the overall downtime in relation to the other baselines. As an example we use a sample of experiments on instance 2 further on, which was chosen as it shows a very clear distinction in performance in DTWP between the considered policies and considers different demand patterns for two separate SKUs. Similar results can achieved for PPO on instances 1-3 with R_{int} reward design and discrete action space.

Figures 4.6 and 4.5 show the distributions of average stock-out duration (in other words, average DTWP

duration) and total downtime distribution per a plan group (and consequently, local warehouse) per policy. Total downtime per machine group on average is 190% higher for NORA in comparison to PPO, with only group 3 in this case being managed better by NORA. Standard deviation for these experiments are also at 48% higher for NORA. The results here are can be called statistically significant, as the confidence intervals of these measurements do not cross. As for the average downtime duration, NORA and PPO manage to achieve very close results. While the mean value of PPO is lower by 37%, the confidence intervals of these measurements do intersect, thus we can deduce, that both policies have approximately same performance in terms of the stock-outs, coupled with high variance for both methods. Nevertheless, together with the total downtime measurements, it can be said that PPO tends to have decreased downtime (i.e. stock-out duration) then NORA and FIFO.

Thus, despite having higher number of stock-out situations, PPO still manages to obtain increased performance over NORA in the considered instance. Consequently, one of the reasons for such situation is that PPO manages to maintain the average waiting times for spare parts lower then NORA's. Thus, despite the fact that PPO generates more stockouts then NORA, it has shorter stockouts, demonstrated by Figure 4.5. Furthermore, Figure 4.6 shows the distribution of downtime per machine group ($|N| = 4$) for the same distributions as demonstrated in Figure 4.4. One can observe that FIFO manages poorly in the groups 1 and 3 with lower aggregate rates, which correlates on the handling of stockouts as well. NORA generally performs better then FIFO, but has more uncertainty on the same "slow moving" groups, while on average total downtime on groups 2 and 4 with higher aggregate demand have higher total downtime. Such result logically comes from higher throughput in these groups. PPO, however, manages downtimes in groups 1,2 and 4 more optimally on average, then NORA.

4.2.3 Analysis of demand satisfaction

Having more stock-outs for PPO results in reduced costs, as these stock-outs are shorter. However, technically the cost performance depend on the DTWP indirectly. The other reason for that lies in how these stockouts are handled. We already discussed, that these stock-outs are different in nature, which is taken into account by PPO. Here, the topic of PPO behaves in relation to satisfying demand through secondary metric, defined in section 3.1. Here, in contrast to the previous section we still explore Scenario 3 in terms of various demand patterns, but we use instance 3 as an example as it showcases high performance of PPO and has increased number of local warehouses $|J|$.

Across multiple experiments we observed various behavior from the trained PPO policies. Most of these behaviors could not be attributed to a particular pattern. However, one noticeable pattern, a difference between the baselines and PPO was discovered for the simple instances 1-3. It can be presented as follows: PPO tends to concentrate most of its allocation actions (up to 50%) on one warehouse in the network, across multiple instances. Then this warehouse serves as a "hub" for lateral transshipments to the other warehouses. FIFO tends to allocate greedily and is directly guided by the demands, leading to more

stockouts on locations with lower overall demand rates, while NORA is more structured, but follows a similar approach. The difference from FIFO is that NORA is able to account for the future demand density, but still has a share of emergencies associated with ”slow” warehouses/plan groups.

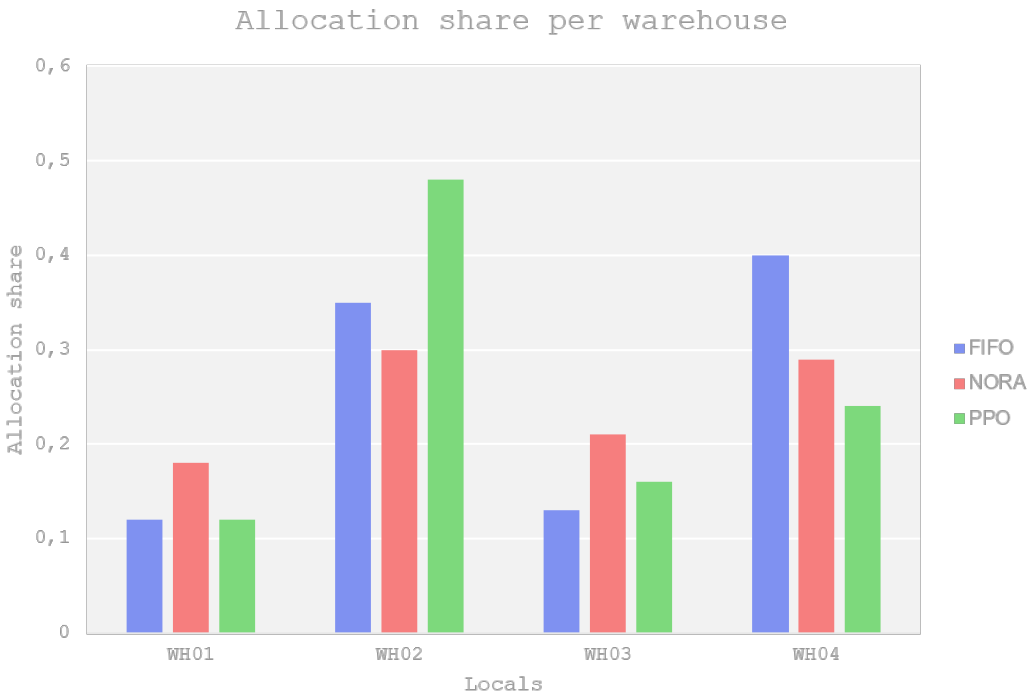


Figure 4.7: Relative allocation share of SKU1 from CW per local warehouse on instance 3

Figure 4.7 therefore shows the allocation shares per local warehouse for a representative episode within Instance 3. As mentioned above, PPO here concentrated most of the allocation in warehouse WH02. In this instance, WH02 and WH04 have highest aggregate demand, associated with fast moving items, while WH01 and WH03 are associated with low aggregate demand and slow movers. With such premise, FIFO concentrates in optimizing these ”fast” locations. As a result, it tends to have a lot of emergency shipments for the ”slow” locations, as the method is skewed towards high demand rates satisfaction. NORA behaves in a much more reserved manner. It also is demand rate oriented, but allocates more equally, which results in more stable performance. Finally, PPO tends to exploit WH02 for the most of its allocation decisions. It can be explained and hypothesized that PPO tends to perform such allocation due to the position of WH02. Inherently, within the formulation of experiments, we used a single definition of the grid (refer to section 3.3.1) for warehousing positioning. Technically, it was not expected that the positioning of the warehouses would have a direct effect on the optimization within the problem. However, it can partially explain the behavior of the method. We can hypothesize, that PPO mainly could consider both WH02 and WH04 for prioritized allocation, and the selection of WH02 is based on the fact that the lateral transshipments from it on average take less time than from WH04 to the other locations. While PPO was

not optimizing directly for the decrease of the lead times, it can be considered a peculiar result of the method using the existing underlying logic of the model to its advantage.

Consequently, we want to consider how demand is satisfied within the system in accordance with the metric for demand satisfaction from section 3.1. Here, WH04 is considered from the data described above. Figure 4.8 shows how the demand was satisfied for this warehouse. As a rule, two main differences were noticed for the baselines of the research. FIFO method tends to use comparatively a lot of emergency shipments, θ demand share takes from 40% to 50%, depending on the complexity of the instance. At the same time, it does not require as much of laterals. NORA is able to satisfy most of it demand locally, as well as PPO. The difference between the two is that PPO can use a very small amount of emergencies (5%), compensating with lateral shipments. NORA uses less lateral, but has to resort to more expensive emergency shipments. The latter might explain the edge, gained by PPO - one one hand it is not large, as PPO still has to balance the network out through internal routings, but on the other hand it manages to avoid large costs, associated with emergency shipments.

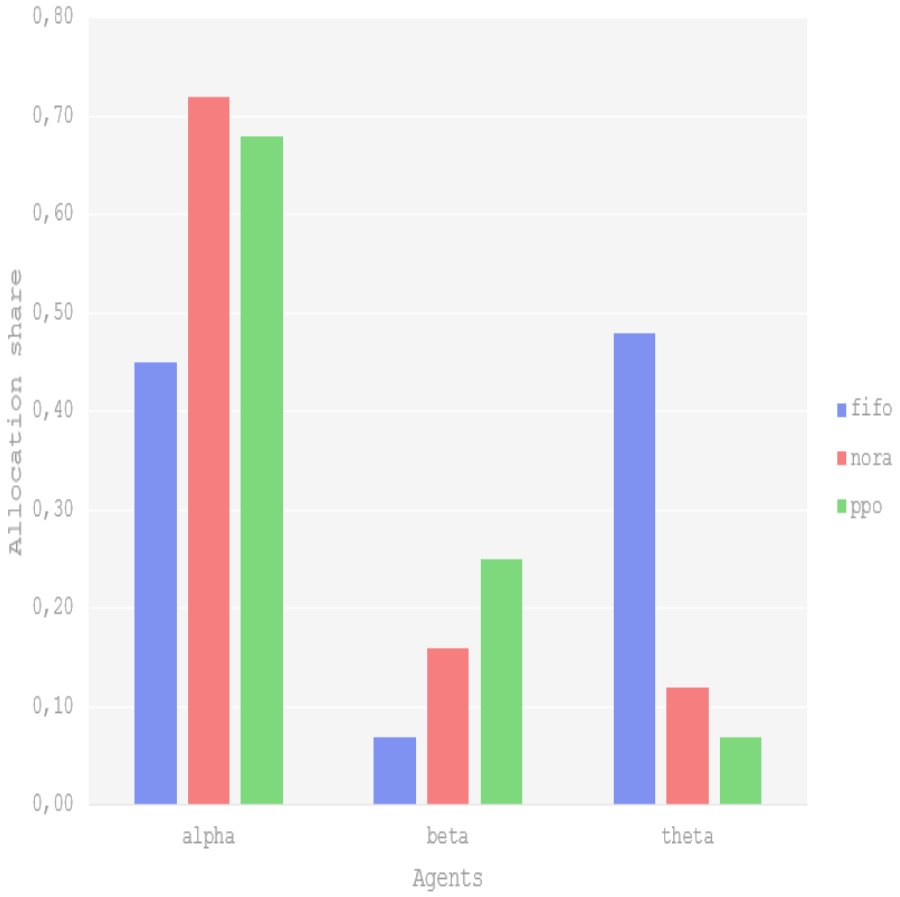


Figure 4.8: Demand satisfaction from local, emergency and lateral methods for US03 in instance 3 for SKU 1

4.3 Summary of the discussion

This chapter attempts to make a number of generalized conclusions about the results of PPO training and behavior. These statements are based on the numerical experiments and allow for an extent of interpretation, but are still applicable to the specific instances of the problem, as well as show possible analysis in how the optimization procedures can be improved.

Training of PPO, apart from extensive search in the hyper-parameter space, requires at times interference and manual adjustments. These are caused by the trends of PPO for convergence towards local optima. We can analyze the convergence and progress in training with various parameters, e.g. reward function miss factor.

PPO generally tends to concentrate stocks in a particular node in the network and use lateral transshipments in order to distribute stock over the network. In such way, the DRL policy manages to maintain enough stock in the network at all times and avoid expensive emergency shipments. Such picture is supported by the amounts of demand satisfied via the 3 available methods. In addition to that, PPO consequently allows for more stockouts in absolute numbers, but these stockouts are both on average and in total DTWP lower than with NORA - as less emergencies are use.

In general, PPO behavior is not directly driven by the demand patterns, as the agent does not inherently have the information on the demand (as NORA) or reacts on it greedily (as FIFO). Instead, PPO optimizes for the total system costs with the inferred information on the network structure and comes to a state, when it manages to use lateral transshipments to cope with the uncertainty in the network, leading to higher gross costs and longer downtimes.

Chapter 5

Conclusions

5.1 Results summary

In this paper, we considered a business problem within ASML service supply chain, corresponding to the inefficiency of allocation decisions in terms of total operational costs. Such problem presents a challenge to the existing traditional methodologies, applied in the field. Hence, we presented an application of a novel methodology of Deep Reinforcement Learning for spare parts inventory control. Namely, the goal here was to address the main research question (refer to section 1.3):

How can we decrease operational costs, associated with allocation decisions in the ASML service supply chain network, with application of Deep Reinforcement Learning?

The end result of this application is a policy for making allocation decisions, optimized for the total incurred costs for holding and transporting spare parts, as well the downtime of customer machines, associated with waiting for the part to be delivered. The main question on "how" therefore is addressed in several steps. Initially, considered business problem was formalized and translated into a model and, further on, a simulated environment in accordance with RQ3 and RQ4 from 1.3 (here and further). This model is evaluated with the existing baseline method, applied in the company - NORA system, as well as a basic FIFO heuristic (answering on RQ1 and RQ2). Consequently, Deep Reinforcement Learning agents were trained to represent an optimized policy for the allocation decisions within the network. This training included an iterative process of Proximal Policy Optimization (PPO) algorithm, making use of an artificial neural network as a powerful function approximator for the policy (RQ5 and RQ6). We describe various factors, contributing to the success of the training, such as application of Actor-Critic approach, dynamic management of hyper-parameters (especially related to the definition of training data and value function) and the training paradigm. Consequently, we show that on several base instances of the problem, our PPO agents are able to gain an edge over the baselines with average of 9% and max of 22% decreased costs in relation to NORA (and address RQ7). On more complex instances it is able to

perform almost as good as the baseline - within 3% of cost increase. Further investigation showed that under given design choices on action validation and definition of Markov Decision Process, PPO exhibits a particular type of behaviors, associated with exploiting the mechanics of the model and environment, not directly presented to the agent. Namely, some PPO policies managed to indirectly use cost and positioning information from the experiences in the simulated environment, leading to unique strategies in optimizing the total operational costs under the customer Service Level Agreements.

This research cannot be reviewed as an exhaustive analysis neither of the problem setting, nor the method applied. Rather, it is an attempt to prove the viability of applying DRL approach for the problem of spare parts allocation and showcase how it can be the optimal methodology to handle this type of problems. Consequently, applications of these methodology bear extensive possibilities for optimization due to its flexibility and generalized, problem-agnostic approach. Moreover, these research shows various ways of interpreting the behavior of the agent, represented with a neural network. These interpretations are important for further understanding on how the overall performance of the agent can be improved and evaluated.

5.2 Recommendations

Average advantage PPO manages to provide (in comparison with the version of the current system within the company, NORA) in terms of holding and transport costs is 9%. These results primarily relate to the instances with 1 or 2 SKUs. While current approach struggles with scalability for the amount of SKUs used, it is still possible to make small-scale optimization for a number of most important SKUs in the system in the as-is state of the method. However, the main recommendation is to continue the exploration of DRL applications in the planning. With sufficient training and tuning, we managed to derive policies, beating the state-of-the-art heuristic. Therefore, potentially similar results can achieved for a larger scale instances of the problem, however further research is required to either support or dismiss such hypothesis.

One of the main strengths of the DRL approach (PPO in particular) are its adaptability and flexibility. One can encode different types of information in the state space for the method to make the decisions, and use various design rewards to optimize for the chosen purpose. Therefore, DRL approach has a good perspective for the integration with existing systems within the company. For instance, the method works successfully with the provided base stock levels, computed by the system within ASML (an adapted version was used in this research). Potentially, PPO can be trained flexibly for various time periods and demand forecasts. Unlike NORA, PPO is more adaptive towards unfavorable situations and still is able to generate relatively acceptable results with sub-optimal base stock levels, as well as less accurate forecast predictions. DRL is a generalized learning framework, and this research shows that it can be adaptively applied within the close-to-real-life scenarios at smaller scale.

As an end product, a system, based on a DRL agent would be close to the current algorithmic method operationally. There are various ways to deploy the policies, obtained with the DRL methods. It can be used as a decision-making system on its own or can help to derive generalized business rules to be then used for the decision making. Conversely, periodic re-training of the agent under new input data (new base stock levels, demand forecasts, changed targets and cost model, etc.) can have a computational toll on the resources of the company and potentially require prolonged time before it can be applied. At the same time, obtained policies can provide long term cost-efficient decision sequences.

5.3 Further research

Extensive follow-up research is required for the given problem setting in combination with the DRL methodology. This paper can be reviewed as an entrance point into the topic, while a variety of issues can be considered in more detail in the future.

Main issues, that are to be addressed by the research are:

- Scalability. The complexity of the problem increases exponentially with the number of locations and SKUs in the network, as it results into the growth of state and action spaces we have to perform search through. Larger instances of the problem were experimented on without PPO agents being able to get close to convergence and level with the performance of NORA baseline due to problems in training. PPO experienced particular problems with the convergence to local optima, overflow of explored data and convergence of the Critic model. Primarily (and instinctively) these problems can be addressed by an extended training procedure with more extensive computational resources, which would also support more complex neural network structures. However, such solution is a shallow attempt to oversee the primary issues and rely on the machinery rather than the method design. Further research shall be performed in order to find how each of the scalability issues can be handled at realistic costs in computing power and time, as well as with a boost to performance.

  A separate issues with scalability for the problem at hand can be the exploration how SKUs can be optimized jointly. Current research on the problem ([42], [41]) apply a system approach towards inventory control optimization, but they still consider SKUs individually. DRL method allows for tractable solutions to be obtained, however the quality of these solutions can and shall be improved. In its core, the problem setting requires extensive number of experiments to be carried out with different sets of model, MDP and PPO parameters to find the general pathways towards optimal large-scale solutions.

- Extensive search through the parameters of the model. In order to draw more substantiated conclusions, a larger number of experiments across more problem instances shall be performed. Such search should also include different cost models, new sources of stochaticity and changes in the

network structure, making it an incomplete graph.

One of the results of this research is the idea of studying how the actual physical positions of local warehouses in the network can be accounted for while taking the allocation decisions. Another perspective direction of parameter search is the study in how the capacity of the machine groups can influence the final results and be related to the changes in the performance of PPO. Additionally, further research in the stochasticity of lead times can be performed, as such setting comes very close to the real world randomness from a multitude of factors.

- Complex decisions. This problem settings includes a number of decisions, which were out of scope for this research, but are directly involved with the allocation decision - such as supply order, timing of allocation, pro-active lateral transshipments, etc. These decisions were modeled in accordance with well defined rules and presented the dynamics of the model, but in other cases they can be also modeled to be a part of the decision making by the agent.

  The current system of NORA is able to advise on pro-active lateral transshipments within local networks in order to minimize the potential stock-outs. Within this research such feature was disabled to avoid added complexity and study the basics of the model, but it can be included in the follow-up research. Deciding on the quantities to be supplied to the network is another major decision, which can be managed via a DRL approach. Current research considered a base stock policy approach to perform this decision, but it can be optimized jointly with the allocation. Such decision making is complex and is expected to be more difficult to train a DRL agent for, but can potentially deliver synergetical performance results.

- Reward design. A separate investigation should be performed in how the design of the reward function can be improved in order to receive a boos in the performance of the agent. Various components and factors can be tested in order to find the optimal combination of cost factors or other types of optimization objectives.

Consequent research can concentrate on specific problems, highlighted in this paper and basically provide a deeper exploration at the application of DRL methods in complex stochastic environments.

Bibliography

[1] ASML. About asml - the world's supplier for the semiconductor industry. 2

[2] Sven Axs. *Inventory control.* Springer Science+Business Media, New York, NY, 2015. 6, 9, 10, 11, 15

[3] Christopher A. Boone, Christopher W. Craighead, and Joe B. Hanna. Critical challenges of inventory management in service parts supply: A Delphi study. *Operations Management Research*, 1(1):31–39, September 2008. 12, 13

[4] Richard J. Boucherie, Geert-Jan van Houtum, Judith Timmer, and Jan-Kees van Ommeren. A TWO-ECHELON SPARE PARTS NETWORK WITH LATERAL AND EMERGENCY SHIPMENTS: A PRODUCT-FORM APPROXIMATION. *Probability in the Engineering and Informational Sciences*, 32(4):536–555, October 2018. 14

[5] Steven J Bradtke and Michael O Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in neural information processing systems*, pages 393–400, 1995. 17

[6] G. Cano Lopes, M. Ferreira, A. da Silva Simões, and E. Luna Colombini. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pages 503–508, 2018. 39

[7] T Chakrabarty, Bibhas Chandra Giri, and KS Chaudhuri. An eoq model for items with weibull distribution deterioration, shortages and trended demand: an extension of philip's model. *Computers & Operations Research*, 25(7-8):649–657, 1998. 12

[8] Bowen Cui, Zili Wang, Qiang Feng, Yi Ren, Bo Sun, Dezhen Yang, and Cheng Qian. A Heuristic Hybrid Optimization Approach for Spare Parts and Maintenance Workers Under Partial Pooling. *IEEE Access*, 7:137835–137847, 2019. 14

[9] Ton de Kok, Christopher Grob, Marco Laumanns, Stefan Minner, Jörg Rambau, and Konrad Schade. A typology and literature review on stochastic multi-echelon inventory models. *European Journal of Operational Research*, 269(3):955–983, 2018. 15

[10] Ton de Kok, Christopher Grob, Marco Laumanns, Stefan Minner, Jörg Rambau, and Konrad Schade. A typology and literature review on stochastic multi-echelon inventory models. *European Journal of Operational Research*, 269(3):955–983, September 2018. 10, 11, 12, 54

[11] Ilaria Giannoccaro and Pierpaolo Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2):153–161, July 2002. 21

[12] Joren Gijsbrechts and Robert N Boute. Can Deep Reinforcement Learning Improve Inventory Management? Performance on Dual Sourcing, Lost Sales and Multi-Echelon Problems. page 26. 12, 22

[13] Joren Gijsbrechts, Robert N Boute, Jan A Van Mieghem, and Dennis Zhang. Can deep reinforcement learning improve inventory management? performance on dual sourcing, lost sales and multi-echelon problems. *Performance on Dual Sourcing, Lost Sales and Multi-Echelon Problems (July 29, 2019)*, 2019. 34

[14] Suresh Kumar Goyal. Economic order quantity under conditions of permissible delay in payments. *Journal of the operational research society*, 36(4):335–338, 1985. 12

[15] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017. 38

[16] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 20

[17] Qiwei Hu, John E. Boylan, Huijing Chen, and Ashraf Labib. OR in spare parts management: A review. *European Journal of Operational Research*, 266(2):395–414, April 2018. 13

[18] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 17

[19] M Khan, MY Jaber, AL Guiffrida, and S Zolfaghari. A review of the extensions of a modified eoq model for imperfect quality items. *International Journal of Production Economics*, 132(1):1–12, 2011. 12

[20] A. A. Kranenburg and G. J. van Houtum. A new partial pooling structure for spare parts networks. *European Journal of Operational Research*, 199(3):908–921, December 2009. 12

[21] Douniel Lamghari-Idrissi, Rob Basten, and Geert-Jan van Houtum. Spare parts inventory control under a fixed-term contract with a long-down constraint. *International Journal of Production Economics*, 219:123–137, January 2020. 13, 14

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 20

[23] Luckeciano C. Melo and Marcos R. O. A. Maximo. Learning humanoid robot running skills through proximal policy optimization, 2019. 39

[24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016. 19

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 19

[26] Fredrik Olsson. An inventory model with unidirectional lateral transshipments. *European Journal of Operational Research*, 200(3):725–732, 2010. 15

[27] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019. 22

[28] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. A deep q-network for the beer game: A deep reinforcement learning algorithm to solve inventory optimization problems. *arXiv preprint arXiv:1708.05924*, 2017. 22

[29] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. A Deep Q-Network for the Beer Game: A Deep Reinforcement Learning algorithm to Solve Inventory Optimization Problems. *arXiv:1708.05924 [cs]*, February 2019. arXiv: 1708.05924. 34

[30] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *CoRR*, abs/1804.02717, 2018. 40

[31] S. Rahimi-Ghahroodi, A. Al Hanbali, I. M. H. Vliegen, and M. A. Cohen. Joint optimization of spare parts inventory and service engineers staffing with full backlogging. *International Journal of Production Economics*, 212:39–50, June 2019. 14

[32] I C Reijnen, T Tan, and G J van Houtum. Inventory planning for spare parts networks with delivery time requirements. page 27. 14, 51, 86

[33] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015. 19, 39

[34] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 41

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 38, 39, 40

[36] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017. 38

[37] Tim Stockheim, Michael Schwind, and Wolfgang Koenig. A reinforcement learning approach for supply chain management. 22

[38] Ruoying Sun, Gang Zhao, and Chunhua Yin. A multi-agent coordination of a supply chain ordering management with multiple members using Reinforcement Learning. *2010 8th IEEE International Conference on Industrial Informatics*, 2010. 21

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. v, 6, 15, 16, 17, 19, 40

[40] Hamdy A. Taha. *Operations research an introduction*. Pearson, Boston, tenth edition edition, 2017. 10

[41] Martijn van Aspert. *Design of an integrated global warehouse and field stock planning concept for spare parts*. Technische Universiteit Eindhoven, 2015. OCLC: 8087189398. v, 4, 13, 14, 26, 34, 35, 51, 53, 54, 55, 74, 83, 86, 89

[42] Geert-Jan van Houtum and Bram Kranenburg. *Spare Parts Inventory Control under System Availability Constraints*, volume 227 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2015. vii, 12, 13, 14, 15, 22, 23, 27, 28, 34, 53, 54, 55, 74, 83, 85, 86, 87

[43] A. C. C. van Wijk, I. J. B. F. Adan, and G. J. van Houtum. Optimal lateral transshipment policies for a two location inventory problem with multiple demand classes. *European Journal of Operational Research*, 272(2):481–495, January 2019. 12, 14, 34

[44] Xun Wang and Stephen M. Disney. The bullwhip effect: Progress, trends and directions. *European Journal of Operational Research*, 250(3):691–701, May 2016. 10

[45] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016. 39

[46] Gang Zhao and Ruoying Sun. Application of multi-agent Reinforcement Learning to supply chain ordering management. *2010 Sixth International Conference on Natural Computation*, 2010. 21

Appendix A

Research questions

The following research questions (RQs) shall expand the main research question:

- RQ1: What is the current state-of-art for the spare parts allocation in ASML service supply chain?
  - What is the structure and dynamics of ASML service supply chain?
  - What is the current allocation strategy/rules applied for spare parts replenishment and allocation?
  - How can we model the current allocation rules in ASML?
- RQ2: What is the current state-of-art for spare parts management in the field of OR?
  - What are the current order policies and how are they obtained in inventory control?
  - What are the current policies and allocation strategies for spare parts supply chains?
  - What are the current applications of stochastic optimization to the allocation decision making?
  - Are there use cases of DRL applications spare parts management?
- RQ3: How can the problem be formulated and solved with DRL?
  - What are the current applications of RL and DRL in OR?
  - Why do we need RL and specifically DRL for the problem at hand?
  - Which RL setting shall be applied (e.g. single agent vs. multi-agent, continuous vs. discrete Markov chain)?
  - How can we formulate the problem as an MDP, including definitions of state and action space?
  - What are the possible extensions/changes to the state and action space of the MDP?
  - What is the optimal reward design for the MDP?

  - What techniques can be applied for the reduction of traversed state and action spaces of the problem?

- RQ4: How can the MDP formulation translated into a simulated environment for DRL?
  - What type of simulation shall be used?
  - What technical environment shall be used?
  - What are the main assumptions/limitations/design decisions for the MDP based simulation?
  - How do we deal with stock-out (apply lost sales or backordering)
  - How do we model spare parts demand and supply?
  - How do we model the key sources of stochasticity in supply lead times and parts failure arrivals?
  - Which statistical metrics shall be collected during the simulation run?
- RQ5: Which DRL technique shall be applied to the problem formulation?
  - What is the current state-of-art in DRL for allocation decision making?
  - Which DRL technique(s) yield the (near) optimal result in the frameworks, similar to the considered formulation?
  - Which architecture shall be used?
  - What are the hyper-parameters of the applied approach?
- RQ6: How can the DRL application be analyzed, improved and experimented on?
  - How can the applied DRL approach be fine-tuned in terms of hyper-parameters?
  - How can we parametrize the reward function?
  - How can the algorithm adjust to various problem instances?
  - How can we adapt the architecture to the possible changes to the state/action space of the problem?
- RQ7: How the results of DRL application can be evaluated and analyzed?
  - How can we compare the current ASML allocation strategy with the policy, devised by a DRL agent?
  - What is the baseline performance of current ASML allocation strategy?
  - How can this baseline be incorporated into the MDP simulation?

Appendix B

Implementation of simulation environment

B.1 Implementation of a simulation environment

In 2.2.1 we defined the general dynamics of the model and provided the relevant notation. This appendix contains some specific information on the practical side of the implementation process.

The implementation of the simulation model was performed in a form of a discrete event simulation. It was implemented in Python with the help of a publicly available dedicated library SimPy. There is a number of key design ideas, underlying the current implementation of the simulation model. Here, specific figures for some parameters are provided, as consequently these parameters were not subject to analysis. Choice for this parameters is based on the test-beds of van Aspert [41] and van Houtum [42].

The main idea of the implementation was to model several main aspects of the model from 2.2.1. Naturally, the modeling of stochastic demand processes can be realized through generating certain sequences of events, attributed to the key objects in the simulation. These objects naturally are derived from the model description. We model such entities as SKU, Local Warehouse, Central Warehouse, Customer and Machine Group explicitly. As per Figure 2.4, each of these entities contains a number of parameters, used within the simulation and defining the relationships between the entities. Therefore, we model series of generated demand events sequences, accompanied with fixed scheduled events on the processes of supply and allocation.

B.1.1 Time domain and events

Before, addressing the simulation events in more detail, first we address the notion of time. We apply a continuous time domain, which is motivated by several major details. First, continuous time domain

allows for precise measuring of the downtime, incurred by the machines - as well as the collection of other statistics. Second, it allows for flexible and natural definition of Poisson processes for the parts failure. Another issue related to the practical side of the MDP implementation, is the notion of *horizon* T. This figure signifies the number of discrete steps, taken within the simulation. As per the problem definition, the time domain itself is continuous in the technical implementation and definition of the simulation logic - thus, all the events in the simulation are scheduled precisely on the time spectrum as real numbers. At the same time, in order to keep the problem under the definition of MDP and not semi MDP, we still keep the action steps discrete. That means that actions are taken through constant intervals in time. We also distinguish supply steps, where supply orders are performed, while actual steps of the simulation model always correspond to discrete events.

The events within the simulation can be divided into scheduled and logical. Scheduled events correspond to the events, which directly have a unique timestamp of occurrence and drive the progression of the simulation.

- Demand events. These are scheduled in accordance with Poisson processes per each SKU-plan group pair.
- Supply order. Previously referred as supply step, this event is associated to ordering of the supply to the central warehouse.
- Supply receipt. This event is scheduled by the previous event, and occurs after designated lead time per SKU.
- Action. This event (action step) is also scheduled in a constant rate, and is handled by the current allocation policy.
- Lateral transshipment initiation and receipt. This event is scheduled by the out-of-stock situation for local stock point. Lateral shipments are scheduled in accordance with the distance metric within the instance definition.
- Emergency shipment initiation and receipt. This event is scheduled by the out-of-stock situation for local stock point. Emergency shipments have a fixed lead time, and are scheduled accordingly.

Logical events are not essential for the simulation logic, but are important for keeping track of the simulation progression. These are for instance, occurred repairs, recording of certain statistics, shipment initiations, etc.

Upon summarizing, each simulation run corresponds to an episode in RL training procedures (further we refer to runs as episodes). Each episode runs until the designated time T, on course of which we have periodic fixed action $\tau_a = 1$ and supply $\tau_{sup} = 1$ steps and random events with demand occurrences. The rest of the events are scheduled by the internal model dynamics.

B.1.2 Stochasticity sources

The main stochasticity source within the simulation are independent Poisson processes per SKU per machine group for the spare part failures. These have constant arrival rates $\lambda_{i,n}$. There is an implementation challenge associated with the constant demand rate, as the machine groups within the problem definition are capacitated. Therefore, when the group runs out of capacity (all machines are broken), the demand can still be generated at a constant rate. In order to handle this challenge, two consequent measures are taken. First, the demand rates per plan group are kept constant, as per the current procedure of demand planning in ASML - where the demand rates are predicted for a fixed period of time (usually a month, but we assume constant rates during longer periods of time). Second, once a plan group has capacity of 0, the events for Poisson process are still scheduled, but are ignored until the capacity becomes strictly positive.

B.1.3 Summary of simulation mechanics

Simulation is based on scheduled events. Poisson processes per SKU per machine group schedule failure events. Once there is a failure, we have three options, applied sequentially if the option above is not available.

1. Resupply from assigned local warehouse j^n. Each machine group has a default (assigned) local warehouse, which stock is first checked for spare parts availability. In case of a success, the failure is repaired with no downtime. Thus, we have decrease $IL^t_{i,j^n} - 1$ and BO^t_{i,j^n} remaining unchanged.

2. In case there is no stock in the assigned warehouse, all the other warehouses in the network are checked one by one in lateral order list v_n, and if we strike a warehouse j_k with sufficient stock, we launch a lateral shipment from that warehouse to the assigned one. The definition of this list can be either fixed manually or based on a distance measure. We sort this list for each location on the premise of Euclidean distance (j^n, j_k) for $m \in |J| - j^n$. This shipment costs a fixed amount in costs c^{lat}_i and takes lead time $k_{dist}(j_0, j_{end})$, where k_{dist} is an adjustable coefficient on time. We fix k_{dist} at 2 for all considered network. We apply changes $BO^t_{i,j^n} + 1$ and $IL^t_{i,j_k} - 1$.

3. Finally, if there is no stock for this SKU anywhere in the network, we have to use an emergency shipment with costs c^{em} and fixed lead time $l_e m = 2$ [42]. We apply changes $BO^t_{i,j^n} + 1$.

In its essence, lateral and emergency shipments are used whenever we have the stock-out situation. Shipment costs are applied upon the dispatch of the event and not the receipt of the product - so we incur costs directly when the stock-out is recorded.

Spare parts within the simulation relate to the base stock levels at each of the locations. These base stock levels correspond to optimal levels, which provide certain service level to the customers. Base stock levels are directly related to the supply ordering, which incomes endogenously to the network within supply

lead time. As the simulation progresses through the episode, there are fixed points at which state of the system is evaluated and supply order is formed. By default, that happens each day as in reality in ASML. Supply is formed per SKU i as $supp_i = \sum_{j \in J} (b_{i,j} - IL_{i,j} + BO_{i,j}) - IL_{i,CW} - PL_i$. Once order is made, it is added to the pipeline stock $PL_i + supp_i$ and event of its arrival is scheduled.

B.1.4 Simulation validation

In order to ensure that the simulation model we apply is valid, a validation procedure is used. In order to validate our discrete-event simulation, we apply a model from van Aspert [41], which in turn is based on the model of Reijnen [32]. This model in the paper of van Aspert is used for field stock planning of ASML and forms a good fit with the model defined in this research, as it also takes the model from van Houtum [42] as its base. More than that, we also use this model in order to produce the optimized base stock levels for the local warehouses in our model. These base stock levels are consequently used by the model to generate supply orders to the central warehouse (refer to 2.2.1) and operations of NORA baseline (refer to 2.4). Here a brief description of the model and its algorithm are provided, while the full and extensive coverage can be found in the work of van Aspert [41].

The main idea of the validation model is to look at the system in the steady state. In contrast to the model employed in [41], we do not consider any contract service metric other then DTWP - which is also used in this research. We only require base stock levels for the local warehouses in the network, therefore the base stock level in the central warehouse is also out of scope here.

Same notation as in Section 2.2.1 is employed for the model entities and sets. In addition we define for each plan group n an array v_n with all warehouses to satisfy the demand here in lateral order (refer section 2.2.1) with p_n defining the length of this array. Let $m_{i,n}$ to be the total demand for SKU i from plan group n (summed demand). We designate vector $S_i = (S_{i,1}, ..., S_{i,|J|})$ as the vector with basestock levels for SKU i in warehouse $j \in J$. These are our decision variables, on which the rest of the problem calculations rely. S then is the vector of this vectors for all SKUs. Furthermore, define :

- $\alpha_{i,n,j}$ to be a fraction of demand of SKU i for plan group n from warehouse j.
- $\theta_{i,n}(S_i) = 1 - \sum_{q=1}^{p_n} \alpha_{i,n,v_n(q)}(S_i)$ as a fraction of demand for SKU i by plan group n that is not satisfied by any of the warehouses or emergency hubs (in array v_n).

The following assumptions are applied, partly based on [32]:

- The demand streams for all SKUs are independent Poisson processes.
- For each SKU, the demand rate is constant.
- The replenishment lead times for SKUs are independent and identically distributed.
- A one-for-one replenishment strategy is applied for all SKUs.

- A First-In First-Out (FIFO) method is assumed as the allocation method in the dynamics of the model.

Cost model employed in validation model is similar to what we formulate in our model, it corresponds to holding costs, as well as costs for transport - sum of costs for emergency and lateral transshipments.

$$C_i(S_i) = \sum_{j \in J} c_i^h S_{i,j} + \sum_{n \in N} m_{i,n}(c_i^{em}\theta_{i,n}(S_i) + \sum_{j \in J_n} c_i^{lat}\alpha_{i,n,j}(S_i) \tag{B.1}$$

, where in comparison to original we use lateral costs c^{lat} fixed for all shipments and keep replenishment from original warehouse of n - j^n - to 0.

We pose down time constraint on the waiting times for parts, and it is possible to calculate waiting times for SKU i for customer n:

$$W_{i,n}(S_i) = l^{em}\theta_{i,n}(S - i) + \sum_{j \in J_n} t_{n,j}\alpha_{i,n,j}(S_i) \tag{B.2}$$

Then we can obtain total DTWP per group with :

$$DTWP_{i,n}(S_i) = \frac{W_{i,n}(S_i) \times m_{i,n}}{|n|} \tag{B.3}$$

Total DTWP is calculated as:

$$DTWP_n(S) = \sum_{i \in I} DTWP_{i,n}(S_i) \tag{B.4}$$

In order to determine optimal base stock levels, we first formulate an optimization problem. We want to minimize total costs in the system subject to down time constraints.

$$\begin{aligned} \min_{S_i} \quad & C(S) = \sum_{i \in I} C_i(S_i) \\ \text{subject to} \quad & DTWP_n(S) \leq DTWP_n^{obj}, \quad \forall n \in N, \\ & S_{i,j} \geq S_{i,j}^{start}, \quad \forall i \in I, j \in J, \\ & S_{i,j} \in \mathbb{S}, \quad \forall i \in I, j \in J. \end{aligned} \tag{B.5}$$

Here $\mathbb{S} = \{S = (S_{i,j}|S_{i,j} \in N_0, \forall i \in J \ \ \& \ \ j \in J\})$ denotes the set of all solutions, and $S_{i,j}^{start} \in \mathbb{S}$ denotes minimum base stock level for SKU i at location j. And since the problem is formed, we shall cover two procedures. The main optimization procedure is based on the greedy heuristic, used throughout the book of van Houtum and Kranenburg [42]. It is an iterative procedure of search of optimal S can be summarized in pseudocode in Algorithm 4. In order to apply it however, at any step we need to evaluate the costs and other metrics , originating from an input vector S. Such is realized with the approximate evaluation procedure, stated below in Algorithm 3.

Before the algorithm is discussed, there are two points for the approximate procedure.

- Overflow demand occurring due to stock-outs at local warehouses are assumed to be Poisson distributed.
- Stock levels at the warehouses are assumed to be independent of each other such that warehouse can be analyzed as separate stock points.

Algorithm 3: Approximate evaluation procedure for each SKU $i \in I$

Result: Obtained $\alpha_{i,n,j}(S_i), \theta_{i,n}(S_i)$

Input: S vector;

Initialization:

In validation procedure we make distinction of setting demand rates per each warehouse and updating them further iteratively. We start with just the original local warehouse to be assigned the overall demand rate and the other ones in the lateral order list are initialized with 0. $\forall j \in J,\ \beta_{i,j} \leftarrow 1 - L(S_{i,j}, l_i^{sup} \sum_{n \in N | v_n(1)=j} m_{i,n})$;

$\forall n \in N,\ m_{i,n,v_n(1)} \leftarrow m_{i,n}$;

$\forall n \in N, j \neq v_n(1),\ m_{i,n,j} \leftarrow 0$

;

while $M_{i,j}$ *change* $\leq \epsilon$ *for all* $j \in J$ **do**

Calculate new demand rates for the lateral order

$m_{i,n,v_n(q)} \leftarrow (1 - \beta_{i,v_n(q-1)}(S_{i,v_n(q-1)}) m_{i,n,v_n(q-1)})$;

For all the local warehouses $j \in J$:

$M_{i,j} \leftarrow \sum_{n \in N} m_{i,n,j}$;

$\beta_{i,j} \leftarrow 1 - L(S_{i,j}, l_i^{sup} M_{i,j})$

end

Now finalized with calculating the demand shares for all $n \in N, j \in J$:

$\alpha_{i,n,j}(S_i) \leftarrow \frac{\beta_{i,j}(S_i) m_{i,n,j}}{m_{i,n}}$;

$\theta_{i,n}(S_i) = 1 - \sum_{q=1}^{p_n} \alpha_{i,n,v_n(q)}(S_i)$

We also need to make several more definitions. $\beta_{i,j}(S_i)$ denotes fill rate of SKU i at warehouse j. Let $m_{i,n,j}$ denote the demand rate for SKU i from plan group n that is faced be the warehouse j and let $M_{i,j} = \sum_{n \in N} m_{i,n,j}$ denote the total demand for all plan groups. Furthermore, we use Erlang Loss probability , denoted as:

$$L(c, \rho) = \frac{\rho^c / c!}{\sum_{x=0}^{c} \rho^x / x!} \tag{B.6}$$

, which is the result from a M/G/c/c system in queuing theory and denotes the probability that all servers are occupied. In the validation model ρ is the demand during the replenishment lead time and c is the number of servers - base stock levels in the local warehouse. It is used as a function in approximate

evaluation.

After obtaining the distribution for how demand is satisfied with the given S, we can calculate $C(S)$ and $DTWP(S)$ in order to optimize for them further in the greedy optimization heuristic. We employ the idea of first satisfying the downtime constraints primarily and similarly adjusting the costs. This heuristic is a simplified version of the one from [41], as we have fewer metrics to optimize for. We define distances d to the sets of feasible solutions (as difference between the values obtained in evaluation and target values). We use objectives pre-defined for down time and within change target for the costs to describe the decrease in distance Δd and ΔC respectively for down times and costs.

$$\Delta_{i,j} d(S) = \sum_{n \in N} \left[\left(\sum_{i \in I} DTWP_{i,n}(S_i') - DTWP_n^{obj} \right)^+ - \left(\sum_{i \in I} DTWP_{i,n}(S_i') + DTWP_{i,n}(S_i + e_j) - DTWP_n^{obj} \right)^+ \right. \tag{B.7}$$

, where $e_j, j \in J$ is a row vector with j^{th} element is 1 and others are 0.

$$\Delta_{i,j} C(S) = C_i(S_i - e_j) - C_i(S_i),\ i \in I, j \in J$$

The we define the relation between the two decreases with $\Gamma_{i,j} = \Delta_{i,j} d(S) / \Delta_{i,j} C(S)$, and use it to perform the greedy selection on the best decrease towards the optimal solution with each step.

Algorithm 4: Optimization procedure

Result: Optimized S base stock levels

$S_{i,j} \leftarrow S_{i,j}^{start}, \forall i \in I, j \in J$;

for *For each* $j \in J, i \in I$ **do**

  Perform evaluation on current S;

  Calculate $\Delta_{i,j} C(S), \Delta_{i,j} d(S), \Gamma_{i,j}$;

  while $d_{DTWP>0}$ **do**

    Determine $\hat{i}$ *and* $\hat{j}$ *such that* $\Gamma_{\hat{i},\hat{j}} \geq \Gamma_{i,j}$;

    Set $S_{\hat{i},\hat{j}} \leftarrow S_{i,j} + 1$;

    Calculate $\Delta_{i,j} C(S), \Delta_{i,j} d(S), \Gamma_{i,j}$;

  end

  Finalize with overall evaluation and calculate all relevant metrics on obtained S.

end

As a result of such validation procedure, we obtain optimal base stock levels and performance metrics on how demand is satisfied in the system. Consequently, we can validate the performance of our simulation. For that we use simple instances of the problem from the test bed of van Aspert [41] and define our

# locals	$m_{i,n}$	S_i	$\alpha_{i,n,v(1)}(S_i)$		$\alpha_{i,n,v(2)}(S_i)$		$\alpha_{i,n,v(3)}(S_i)$		$\alpha_{i,n,v(4)}(S_i)$		$\theta i, n(S_i)$	
			A	O	A	O	A	O	A	O	A	O
2	0.001	1	0.980	0.980	0.020	0.020					0	0
2	0.040	2	0.219	0.220	171	0.177					0.610	0.603
4	0.040	2	0.133	0.130	0.115	0.111	0.100	0.107	0.087	0.089	0.566	0.567
4	0.040	2	0.335	0.329	0.223	0.219	0.148	0.147	0.099	0.105	0.195	0.194

Table B.1: Validation results

simulation along this instances. The comparison is made with the paper of van Aspert .We apply FIFO method (2.4) for the allocation decisions and observe the results in Table B.1. We run simulation for 500 times and use averaged results here. In table A denotes result from van Aspert, and O denotes obtained results from our simulation. Demands are showcased as per day rate.

One can see, that the results do correspond to each other with a small error of (≤ 0.007) which can be attributed to rounding of results and stochasticity, yet deemed acceptable. Hence the modeled is validated across simple instances in terms of demand, which are still comparable to the instances, used in this research.