Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

LONG-TERM RAPID SCENARIO PLANNING IN THE SEMICONDUCTOR INDUSTRY USING DEEP REINFORCEMENT LEARNING

Bibi de Jong[1], Kai Schelthoff[2], Riccardo Lo Bianco[1], and Willem van Jaarsveld[1]

[1]Dept. of Industrial Eng. and Operations Research, Eindhoven University, Eindhoven, NETHERLANDS
[2]NXP Semiconductors N.V., Eindhoven, NETHERLANDS

ABSTRACT

The creation and maintenance of effective production plans is a central problem in supply chain planning. NXP Semiconductors N.V. uses a Mathematical Programming (MP) model to generate offline production plans on both short and long-term horizons. However, production plans need online updates in response to unforeseen events. This process is carried out manually, making the updated plans not optimal. This paper proposes the use of Deep Reinforcement Learning (DRL) as a method to produce close-to-optimal updated plans. DRL is suitable for large-scale problems and can produce close-to-optimal scheduling solutions quickly, whereas MP models ensure optimality at the expense of high computational complexity. The performance of DRL is compared against traditional optimization methods and a MP model using a simulation model that mimics NXP's situation. The results of this study highlight the usefulness of DRL as a tool for short and long-term decision-making in supply chain planning within the semiconductor industry.

1 INTRODUCTION

Supply chain planning is pivotal for semiconductor companies due to the industry's dynamic nature, marked by variable demand, global manufacturing, and multi-tier suppliers (Banerjee 2007). These factors contribute to the complexity of planning processes, making them dynamic, intricate, and highly sophisticated (Mönch et al. 2018). Semiconductor supply chain planning presents unique challenges due to its NP-hard nature, necessitating quick and high-performance solutions (Mönch et al. 2018). Many companies rely on mathematical models for planning, such as NXP's hierarchical Mathematical Programming (MP) model, IBM's SCO processes supported by MP, and Intel's supply chain planning supported by MP (Degbotse et al. 2013), (Denton et al. 2006), (Bean et al. 2005). Other semiconductor companies, like Infineon and Texas Instruments, have also explored using MP models for supply chain planning (Ziarnetzky et al. 2019), (Deng et al. 2010). MP models guarantee optimal plans at the expense of a high computational complexity. This makes MP models a viable solution for weekly planning, where the entire plan is produced based on a set of static variables, but not for daily planning, where the solution algorithm needs to adapt swiftly to unforeseen events.

In NXP, the process of rapidly adapting plans to unforeseen events is known as rapid scenario planning. Currently, the company manages rapid scenario planning manually, requiring supply chain planners to update plans generated over the weekend by an MP model. However, this slow process can lead to sub-optimal resource utilization and operational inefficiencies. In practice, supply chain planners frequently respond to unforeseen events by canceling orders and addressing them again in the following week while leaving the rest of the current plan unchanged. Later, when the new (weekly) plan is generated, it includes the unsatisfied orders from the previous week. This results in underutilization, as planners do not modify the current week's plan but cancel and delay orders.

Reinforcement Learning (RL) and meta-heuristics are popular techniques for quickly producing close-to-optimal scheduling solutions. RL is a method used to learn sensible policies in sequential decision-making problems. In the context of scheduling problems, a policy is a function that, given the current schedule and the remaining orders, assigns one of the remaining orders to a location and a time in the schedule. On the other hand, meta-heuristics explore the space of all possible complete schedules to find a close-to-optimal one.

This work will focus on a Deep Reinforcement Learning (DRL) optimization method for the supply chain planning problem experienced at NXP. DRL involves using advanced deep neural networks to approximate policies or values in RL algorithms. In contrast to RL, DRL is suitable for handling large-scale problems, which is the case for NXP's scenario.

Recent approaches include DRL for task scheduling (Lu et al. 2021), for capacity planning (Tseng et al. 2023), and for supply chain synchronization (Jackson 2022). There is a gap in deep reinforcement learning (DRL) research when it comes to long-term strategic decision-making in supply chain scheduling. This issue needs to be addressed because current planners do not take into account long-term consequences when being pressured to make rapid decisions. A decision made early in the planning process can lead to significant delays, especially if the company is planning to fully utilize all resources. Planners usually choose the path with the least resistance, with minor short-term impacts, ignoring long-term effects. This paper addresses this gap by comparing DRL algorithms against traditional optimization methods and an MP model to produce long-term scheduling plans. Note that this paper focuses on the question of whether it is possible to apply DRL in this context, and if it can match the performance of other methods. It does not however focuses on proving that DRL would be the best option.

We will evaluate the effectiveness of the suggested optimization methods using a simulation model that replicates the conditions at NXP. This optimization challenge involves three distinct physical supply chains, with each chain consisting of three types of sequential resources: assembly resources, wire-bonding resources, and final testers. Among these resources, the final testers are the bottleneck resources. The Customer Order Decoupling Point (CODP) is situated between the front-end and back-end processes, assuming infinite stock within the CODP. Three supply paths are available: dual-sourcing path capable of producing both product types A and B, and two dedicated supply paths, each exclusively producing either type A or B. Lead times from CODP to end-product storage differ, with the dual-sourcing path taking one week and the dedicated paths taking two weeks. Despite similar weekly capacities across supply paths, the dual-sourcing path exhibits faster production for both product types. The system is depicted in Figure 1.

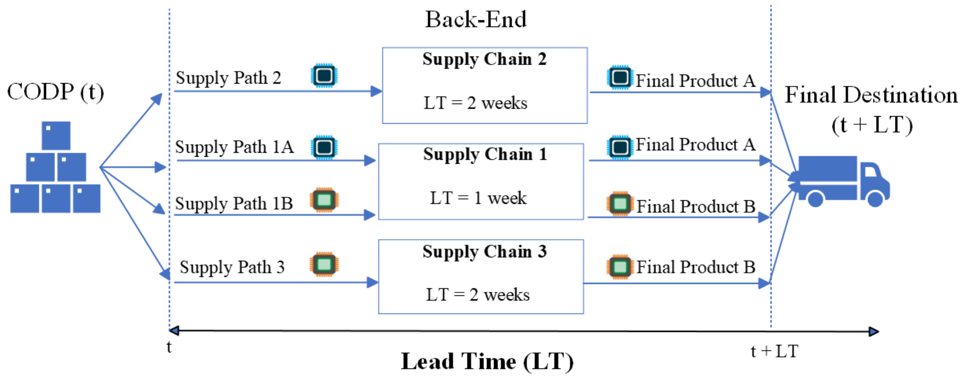

Figure 1: Simulation of NXP's scenario

Our planning horizon spans 52 weeks to create annual production schedules. Confirmed orders from a five-tier customer priority system (1 being the highest, 5 being lowest) populate the order book. Assumptions include an infinite wafer buffer, a static order book, and fixed lead times. In the next section, we illustrate the performance of traditional and tailored dispatching prioritization rules (DPRs) compared to the MP model and the DRL agents.

2 METHODOLOGY

This section discusses the considered planning methods, namely, DPRs, MP, and DRL. We will highlight each method's advantages and drawbacks. This paper utilizes two (policy-based) DRL algorithms: the well-established Proximal Policy Optimization algorithm (Schulman et al. 2017) and the recently proposed Deep Controlled Learning (Temizöz et al. 2020). We will show how the latter yields superior performance.

2.1 Dispatching Prioritization Rules (DPRs)

DPRs aim to construct schedules from scratch, allocating partial orders to supply paths for specific weeks while minimizing order lateness and magnitude. We introduce generic and tailored DPRs for the context of supply chain. The generic DPR is based on the Earliest Due Date (EDD); this benchmark assesses capacity for all product type-supply path combinations, assigning partial orders based on confirmed due dates and customer priorities (i.e. order release decision). However, early commitments of capacity may result in inefficiency.

The tailored DPR differentiates from the generic benchmark by storing a potentially optimal action choice for each supply path and product type combination, totaling four combinations. Unlike the benchmark, it is not constrained by sequential decision-making that necessitates utilizing all capacity in the current week to schedule for the next week. Instead, in each iteration, the decision rules determine an action choice, comprising information about the supply path (plus product type): the week from which capacity is deducted and the items will arrive (which may not align with the scheduling week), and the confirmed due date of the order to be scheduled. The overarching approach remains similar: iteratively prioritize orders by each week and priority until unmet demand is addressed. Similar dispatching rules are applied by (Horiguchi et al. 2001), within the semiconductor industry context. The DPR selects the supply path (plus product type) to commence production, targeting the ”First Available Week,” ideally aligning with the Just-In-Time (JIT) week to maximize efficiency. In cases where JIT scheduling is not feasible, orders are scheduled earlier, or when that is not feasible, later. The JIT date for each supply path synchronizes with the confirmed due dates of products A and B that we are currently planning following the stacked approach. Orders are assigned to the alternate supply path if actions exhibit similar customer priorities and confirmed due dates and are equally early or late. Opting for the alternate supply path, rather than the preferred or dual-sourcing path, when comparable factors enhance flexibility. It accommodates potential production needs for the other product type.

2.2 Mathematical Programming (MP) model

In this work, we use the results of the MP model as a baseline. In each subsequent iteration, less urgent customer orders are scheduled, using the left-over, available capacity of resources that are not already occupied by higher-priority customers (in hours). While this method can speed up convergence (as the solution space decreases after each iteration), there’s a risk of reaching a local optimal solution and not exploring the entire solution space. To tackle this, the decision variables are reset and optimized for each iteration, allowing a more thorough exploration of different solution regions. This approach increases computational time but is better suited for achieving a ”global optimum.” Additionally, the paper retains the previous objective value by saving the minimized cumulative shortage for priority 1 orders after the first iteration (i.e., the sum of lateness multiplied by the size of priority 1 orders). This saved value is then added as a new constraint for subsequent iterations to maintain consistency in the initial solution and prevent converging to local optima. Therefore, the next iteration minimizes both priority 1 and 2 orders while adhering to the new constraint, ensuring that the cumulative shortage for priority 1 orders does not exceed the previously saved cumulative shortage for priority 1. The same model is then used for five iterations, with each iteration becoming more complex to solve due to additional constraints from previous iterations and an expanding solution space by allowing more order priority types to be simultaneously planned. The final iteration solves for all orders (priorities 1 through 5) and has four additional constraints relative to the first iteration, ensuring that the saved cumulative shortages for priorities 1 through 4 (from the previous iterations) do not exceed the cumulative shortages for priorities 1 through 4 in the current model. We will not have the MP model results available for all cases due to time constraints. We can only run it a few times, so the results can only be used as a reference for the deterministic problem and problems with low levels of randomness. It can't be used for results when a high level of randomness is introduced because of the variability of the solutions and the number of runs needed to have a good confidence interval.

2.3 Deep Reinforcement Learning (DRL)

2.3.1 Markov Decision Processes (MDPs)

We propose an MDP that mimics the simplified semiconductor back-end supply chain shown in section 1. The Markov Decision Process is a mathematical framework for modeling decision-making problems with partly random and controllable outcomes. It is a framework that addresses reinforcement learning problems. An MDP is defined by a tuple $< S, A, P, R, \gamma >$, where: A: Finite action set, S: Finite-state set, R: Reward function, P: State transition probability, and γ: Discount factor $\gamma \in [0,1]$

The transition probability, $P(s'|s,a)$, refers to the likelihood of going to state $s' \in S$, given your current state $s \in S$, and a chosen action $a \in A$. The states capture the agent's perception of the environment. A policy π is a function mapping states to actions. The policy defines the behavior of an RL agent. A policy is a probability distribution on the set of actions A given the current state s. It provides a probability of picking an action a at state s. A value function is the *expected* reward over a state (state-value function) or action (action-value function). Reinforcement learning aims to find the optimal policy by iteratively updating value functions, illustrated through the Bellman equation (Puterman 2014): $v_\pi(s) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$. The Bellman equation decomposes the value function into the expected immediate reward and the expected value function of the next time step: $v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(s') | S_t = s]$. The Bellman equation guides the algorithm's action selection, providing the foundation for value-based algorithms that iteratively update the value function until convergence.

For direct comparison with MP based approaches, our MDP formulation involves the scheduling of a known set of orders for a period of 52 weeks. Our MDP formulation involves the sequential allocation of those orders to supply path-week combinations. We first allocate priority 1 orders, then priority 2 orders, etc; within each priority, we first allocate the orders with the earliest due date, etc. The MDP state space keeps track of the remaining capacity for each week, path, and the remaining orders and is discussed in detail in Section 2.3.2. The MDP action specifies the supply path and the week that the order should be produced, an action may prescribe that the product for the requested order is to be produced in week w using supply path sp. However, actions are contextual: the production week is specified *relative* to some focal week that depends on the order due date and the capacity available for the resources, this will be discussed in Section 2.3.3.

2.3.2 State Space

The state space encompasses all necessary information to represent a planning scenario, i.e. a situation in which some of the orders in the order book are planned, while orders still need to be planned. In particular, the state contains the unplanned orders (the *order book*), and the remaining capacity on each of the supply paths, for each of the weeks in the planning horizon. This information is primarily captured in matrices or vectors containing integers. For the order book, the matrix size (52 (weeks) X 5 (priorities)) for each product is fixed. Figure 2 illustrates how unfulfilled (i.e. still to be planned) demand is stored. The quantities are aggregated for a specific priority, with a specific confirmed due date. If there is no demand for this combination, the matrix is filled with a 0 for that particular index. To mimic the current way of scheduling (i.e. the stacked mathematical programming approach), higher priority orders (e.g. priority 1) are satisfied before lower priorities, and according to "Earliest-Due-Date". Therefore, the state space also includes information for the current customer priority and confirmed due date (refer to the green square in Figure 2) for the product A and B orders currently being scheduled.

Note that supply path 1 is a dual-sourcing supply path. Capacity is subtracted by choosing either to produce product A, or product B. Due to the unique usage rates, the amount of hours needed is different

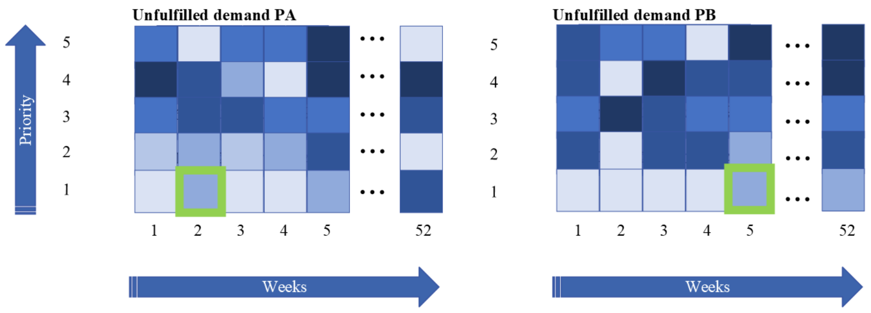

Figure 2: Matrices including unfulfilled demand for product A and B

to produce 2500 items (i.e. 2500 items = 1 lot). Due to order batching, orders are always expressed as a multiple of 2500. After each action, where 2500 items are scheduled for production, the capacity is adjusted accordingly. Capacity is subtracted from the moment of arrival.

The state space keeps track of *focal week* or *First Week Available* (FWA) for each specific order. The focal week for each supply path-product type combination is determined depending on the current due date of product A and product B, as well as the weekly capacity left. Idealistically the FWA is equal to the JIT date. This means that the orders arrive from production in the same week as when they are needed. However, this is not always possible due to capacity constraints. When production capacity in that week is insufficient, the focal week is moved to an earlier week with sufficient production capacity (staying as close as possible to the JIT date), or, if there is no such week, it is moved to a later week (but closest to the JIT date).

All in all, the state space includes 2 matrices, one for each product type, of size 52(weeks) multiplied by 5(priority types), where each index represents the aggregated quantities for a specific due date and priority. Another matrix includes the available capacity (in hours) for each week, for each final tester (i.e., bottleneck), and for each supply path of size 52(weeks) multiplied by 3(physical supply paths). We also keep track of the current due date and priority being scheduled for each product type (i.e. 4 scalers). For each combination of product type and supply path, we track the "focal week" and previous weeks that still have capacity available to schedule an order for 2500 items (this will be explained in more detail in subsubsection 2.3.3). The information includes the week numbers (from 1 to 52) and the difference (measured in number of weeks) from a specific week to the "focal week." This results in 24 values: 4 (product type-supply path combinations) x 3 weeks x 2 (absolute week numbers and relative differences respective to the focal week).

2.3.3 Action Space

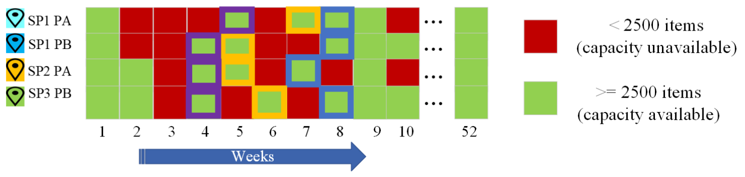

Figure 3: Matrix including the available capacity (in number of items) for each week, for each supply path, based on the bottleneck resource. For supply path 1, products A and B, it is based on 100% of the capacity.

Figure 3 is an illustrative example highlighting the information derived from the state and illustrates the action space for a specific state. It tracks the production capacity for each supply path and product type combination, along with the optimal scheduling week based on the Just-In-Time approach. Note that the capacity is subtracted from the moment of arrival. In this figure, suppose that there is demand for either product A or B due in week 8. The action space consists of 12 actions. For each combination of product type and supply path, the agent can decide to assign a partial order to the *focal week* (see the blue squares in Figure 3), the first available week earlier before the focal week (see the orange squares in Figure 3) or the second available week before the focal week (refer to the purple squares in Figure 3). Note that the lead time for supply chain 1 is one week, and for supply chains 2 and 3 is two weeks. Our annual production plan starts in week 1. Our planning horizon is spanning from week 1 until week 52; for products to arrive at week t, we need to start production in a week $t - leadtime$.

Long-term decision-making in this scheduling problem necessitates action masking to align with prioritization based on due dates and customer priorities. Specifically, actions are masked to restrict the neural network (NN) to selecting actions involving the product type with the highest priority or the lowest confirmed due date (comparing the current due date and priority for product A and B orders), maintaining fidelity to the stacking methodology. Additionally, capacity constraints dictate further action masking to prevent the selection of weeks with insufficient capacity. Actions concerning product types that are already fully satisfied are masked to avoid unnecessary noise that may hinder the NN's learning ability. In essence, action masking is implemented to constrain the selection of actions by the neural network to those pertaining to the product type exhibiting the highest priority or the earliest confirmed due date, thereby upholding the stacking methodology. Moreover, action masking takes into consideration capacity constraints, thereby precluding the selection of weeks characterized by insufficient capacity.

2.3.4 Reward Function

Because of the assumed fixed lead times, it is known at the moment of (partially) scheduling an order, whether this (partial) order is late, and by how many weeks. Therefore, we can provide the agent with feedback when scheduling orders late immediately after the decision is made. W_{plan} is the week selected by the JIT approach, and W_{DD} is the confirmed due date. The feedback signal is 0 when orders are scheduled early or just in time. The optimization problems reach the "final state" when all the orders in the order book have been allocated, or when all capacity is utilized. There are always orders left unsatisfied when the state terminates. The cumulative demand (i.e., CD in Equation 1) of unsatisfied orders is given as a final reward at the end. Equation 1 weights the unfulfilled demand based on quantity and lateness (i.e. week 52 - $week_{DD}$). This reward function directly aligns with the objective value of the MP model, except for the priority weights. We provide the agent with priority weights (i.e., w_{prio}) to communicate that the customer priorities matter and not to risk lateness of higher prioritized orders to avoid lateness for lower prioritized orders.

$$CD_{t,prio} = \sum_{1}^{t} \left(\sum_{t'=1}^{t' \leq t} D_{t',prio} \right) \tag{1}$$

$$R(x) = \begin{cases} 2500 * (W_{plan} - W_{DD}) * w_{prio}, & \text{if } W_{DD} < W_{plan} \\ \sum_{prio=1}^{5} (\sum_{t=1}^{t=52} (CD_{t,prio})) * w_{prio}, & \text{if state is final} \\ 0, & \text{otherwise} \end{cases}$$

2.3.5 Algorithms

Reinforcement learning (RL) is a machine learning paradigm wherein an agent learns to solve a task through iterative trial and error. The fundamental steps in RL involve the agent being in a state s, taking an action a, receiving a reward r, and transitioning to a new state s'. This process forms the foundation of all RL algorithms. However, algorithms differ in methods for updating the policy.

Proximal Policy Optimization (PPO): Proximal Policy Optimization (PPO) is a prominent deep policy gradient algorithm that directly enhances a policy while incorporating a clipping mechanism to moderate large policy updates, as introduced by (Schulman, Wolski, Dhariwal, Radford, and Klimov 2017). PPO seeks to strike a balance between exploration and exploitation by optimizing a surrogate objective function, often leveraging probability ratios to regularize policy updates. Serving as a successor to Trust Region Policy Optimization (TRPO), PPO maintains the concept of policy optimization with a trust region constraint while simplifying its implementation.

Deep Controlled Learning (DCL): Deep Controlled Learning (DCL) is an approximate policy iteration algorithm that iteratively refines policies by defining RL as a classification problem (Temizöz, Imdahl, Dijkman, Lamghari-Idrissi, and van Jaarsveld 2020). Utilizing simulation, DCL accumulates state-action pairs to construct a dataset employed for neural network training. For each state in the dataset, DCL determines an estimated ”optimal” action by selecting the action with the least anticipated expected costs over a trajectory. While mapping states to estimate optimal actions, DCL progressively constructs datasets and refines policies.

Both PPO and DCL update the policy directly in each iteration (i.e., policy iteration). PPO is an on-policy method: the policy is updated based on the data collected by the current version of the policy. Once the parameters are updated, it discards the old policy. DCL, on the other hand, is an off-policy method. The data used for learning can come from a different policy than being updated. While PPO bases its parameter updates on a trajectory, DCL bases its parameter updates on multiple trajectories. During the exploration, the neurel network is learning a function that, given a state, returns an action. This is called a policy function.

2.3.6 Environment

To enhance the understanding of the environment and decision-making processes in reinforcement learning (RL), it is crucial to differentiate between the concepts of ”state” and ”features”. In reinforcement learning (RL), the ”state” is all the information the agent needs to navigate the environment effectively. It provides a complete picture of the current situation. ”Features” are derived from the state. They are carefully chosen or engineered to give to the neural network (NN) as input. The objective of feature selection is to ensure that the NN receives all the essential information needed for effective learning. The action space encompasses 12 distinct actions, necessitating deeper insight into their implications for the NN. Through exploration of the environment, the NN receives rewards for specific state-action pairs, prompting the inquiry: What information is requisite for actions (a) given a state (s)?

Deterministic Decision Making: Initially, the NN will undergo training in a deterministic context, where all samples are derived from trajectories sharing the same initial order book. For the deterministic case, additional information regarding the order book is not needed, as the NN should inherently recognize the environment through trial and error, given that all samples originate from the same order book. This indirect exposure to circumstantial information aids the agent's learning process.

Stochastic Decision Making: Suppose we observe that the agent effectively learns to optimize scheduling and maximize utilization of preferred supply paths by strategically producing products A and B. In that case, we can further enhance its learning by introducing randomness. One example is allowing orders to be delayed or expedited. Training a Deep Reinforcement Learning (DRL) algorithm with this uncertainty enables the development of an agent capable of rapid scenario planning. Once trained, this agent can generate new entire schedules from scratch in response to order delays or expedited requests. Depending on business needs, the agent can be trained on various stochastic elements, such as order delays. Validating the training of a DRL agent on order book variations with randomness confirms the potential for DRL in rapid scenario planning. The agent's learned strategy is tailored to specific order book characteristics. However, when the order book changes—due to delays, density fluctuations, or other factors—the strategy may need adjustment. Introducing more stochasticity enhances the agent's generalizability but may reduce accuracy.

The level of introduced randomness raises questions about the agent's learning capacity and adaptability. By experimenting with different levels of randomness, we aim to balance adaptability and accuracy.

2.3.7 Feature Engineering

In a stochastic environment, an agent may be unable to learn detailed information about the order book, which means that it may need to adjust its strategy in response to changes in the order book. However, depending on the level of randomness present, the agent may still be able to infer other circumstantial details, especially if the aggregated quantities remain consistent. As a result, the agent can ”learn” the distribution of orders for products A and B within the order book, which can help it anticipate future demand and improve its predictive capabilities.

It can be challenging to identify the features that assist the algorithm in learning and those that may hinder it. The workings of the neural network (NN) and the associations it creates when mapping states to actions are not transparent. Therefore, it is necessary to experiment with different feature combinations to find the optimal configurations. One feature combination consistently outperformed others, indicating its crucial role in mapping states to actions. In Figure 4, feature information is illustrated. The purple, orange, and blue squares represent the action space of the agent with 12 actions. This is based on the confirmed due dates in week 8. The yellow boxes indicate the ”first available week” after the targeted week by following the JIT methodology. The agent is provided with detailed information for 16 squares, including utilization, absolute week numbers, relative week numbers (against the current confirmed due dates), capacity left in hours, and how much capacity is measured in the number of buckets of 2500 items. The number of buckets is determined by the usage rates of each product-resource combination. Additionally, the number of buckets already reserved for products A and B regarding the dual-sourcing supply path are also given to the NN to recognize which weekly combinations of product A and B buckets eventually lead to better performances when the states terminate. Lastly, the aggregated information is provided for the capacities before the purple squares and after the yellow squares in Figure 4. This aggregated data, presented in terms of hours or bucket quantities, furnishes the agent with crucial insights into remaining capacities for each supply path-product type combination, indicative of various facets such as proportionate demand scheduling and indirect details concerning product distribution. Additionally, the utilization metrics of weeks available for scheduling partial orders in specific supply paths offer vital insights into the feasibility of accommodating pending orders within the current action space constraints. This information, customized to reflect differing usage rates across supply paths, helps agents make informed decisions by providing standardized utilization metrics across all actions.

While incorporating order book features (e.g., information about the distributions of the unsatisfied orders) did not disrupt learning, it introduced noise, resulting in a marginal degradation in objective value. Because of the constrained sequence of assigning orders to supply paths, detailed information about the order book is unnecessary. For the NN to learn to anticipate, it uses circumstantial information. This is possible as we keep the total aggregated quantities consistent with the demand for products A and B; we only manipulate the distributions.

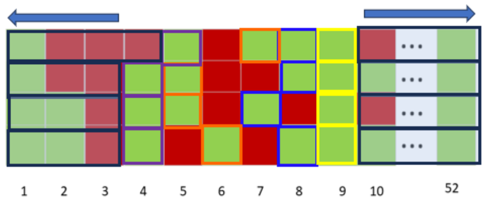

Figure 4: Example visibility features

3 RESULTS

3.1 Computational Time

The running time is more or less the same for all stochastic levels. However, the level of randomness does influence the training time of the agent. Furthermore, for the agent to learn properly, more samples (and thus simulations) must be done before training the agent, increasing the time needed. Training and sampling, however, only need to be done once. After training, the generations of new plans for both DPRs and DRL agents are significantly quicker than relying on MP models, Table 1. However, the savings in computation time are achieved at the cost of solution quality.

Table 1: Comparison of the relative run times

Optimization Technique	Time Decrease (in %)
MP model	Reference
DPRs	- 99.17%
DRL (once trained)	- 98.14%

3.2 Low Level of Randomness

Low stochastic behavior involves shuffling an entire order by 1 week, with a $\beta\%$ chance that the confirmed due date does not change, $\frac{(1-\beta)}{2}\%$ chance that the due date is one week earlier, and $\frac{(1-\beta)}{2}\%$ chance that the due date is one week later. Therefore, the lower β, the more randomness is introduced. Both agents trained using the DCL algorithm, performing better or worse, exhibit marginal changes compared to the tailored DPRs. This similarity might imply that both the DPR and DCL agents experience changes in the objective value influenced by factors such as order distribution and available capacity, potentially leading to late orders due to capacity limitations. Despite differences in mean objective performance between the DCL agent and the tailored DPR, the difference between the DCL agent and tailored DPR is insignificant, so no conclusion can be drawn about which of the two methods is more optimal regarding the objective value. While DCL agents consistently match performance against customized DPRs, PPO agents do not demonstrate this consistency. Nevertheless, PPO agents consistently show a minimal margin of error.

Table 2: Results of the use case discussed throughout this paper solved using the different optimization methods

Optimization Method	**Mean Objective β = 0.5**	**Standard Error of the Mean β = 0.5**	**Mean Objective β = 0.9**	**Standard Error of the Mean β = 0.9**
MP	reference	-	reference	-
Tailored DPR	+14.5%	+/- 0.6%	+16.3%	+/- 0.5%
Generic DPR	+295.3%	+/- 0.2%	+342.38%	+/- 0.1%
PPO	+164.6%	+/- 0.02%	+70.7%	+/- 0.02%
DCL	+14.3%	+/- 0.6%	+16.5%	+/- 0.5%

3.3 High Level of Randomness (different order books)

In this scenario, the distribution for products A and B orders is modified across priority levels one through five, and across due dates while upholding a consistent aggregate quantity for both products A and B across all priority levels and due dates. Due to the high variance in objective values (the cumulative shortage) for each trajectory, the MP model fails to produce consistent results for this scenario. The method results in varying aggregated item numbers for different priorities within each order book, contributing to significant fluctuations in the final cumulative reward. Therefore, the results of the MP model are not considered. The results shown in Table 3 show a pattern similar to those of Table 2. Interestingly, the learned DRL agent, utilizing Deep Controlled Learning, demonstrates performance comparable to, or even slightly better than, the tailored DPR. In contrast, the benchmark based on traditional EDD DPRs exhibits notably inferior performance.

Table 3: Results of the use case discussed throughout this thesis solved using the different optimization methods

Optimization Method	**Mean Objective**	**Standard Error of the Mean**
Tailored DPR	reference	-
Generic Benchmark	+208.2%	+/- 0.7%
PPO	+35.1%	+/- 0.2%
Deep Controlled Learning	-0.02%	+/- 2.6%

4 CONCLUSION

Rapid scenario planning can be effectively conducted using deep reinforcement learning (DRL) algorithms, particularly Deep Controlled Learning (DCL), as evidenced by their satisfactory performance while significantly reducing runtime. This effectiveness holds true across low- and high-stochastic scenarios, with DCL demonstrating adaptability to diverse order books. Furthermore, training the DCL agent in various order books improves its robustness, allowing it to generate solutions quickly for different scenarios after a single training session. Notably, the DCL agent consistently matches the performance of tailored DPRs. However, it is unable to beat the tailored DPR consistently. DCL is better equipped to solve this use case than PPO as it reduces the effect of randomness in the environment.

The tailored DPRs are overengineered for the specific scenario outlined in this paper. Simple decision rules that revolve around having resources capable of producing only one or two product types will not be effective in real-world situations. The tailored DPRs will not hold up when expanding the supply chain for this use case, whereas the MDP can be extended. However, it's important to note that extending the MDP significantly increases complexity and, hence, training time. The DCL agent is trained on trajectories sampled following the tailored decision rules. Expanding the supply chain and thereby reducing the effectiveness of the tailored decision rules will once again impede the agent's efficient learning, necessitating more samples and, in turn, increasing the training time.

5 DISCUSSION

5.1 Long-Term Decision Making and Ambiguity

In reinforcement learning, different sequences of actions and states can lead to the same reward, making long-term decision-making challenging. For instance, in supply chain management, prioritizing between supply paths may not significantly affect rewards due to constraints, but there are still optimization opportunities within the preferred path. Finding the "sweet spots" of the preferred supply path, by combining the number of items for products A and B produced in the same week, can help maximize utilization. However, as different strategies produce the same rewards, this introduces noise for the neural network to recognize tiny differences in orders that can be beneficial.

Trajectory non-uniqueness is especially a problem when introducing randomness. Significant disparities in the reward function can impede the agent's ability to discern which policies are effective. Additionally, it becomes challenging for the agent to isolate the portion of reward signals attributable to differences in the (initial) order book that should be derived from the features presented to the neural network (NN). When the reward function varies significantly, the agent may struggle to understand the underlying patterns and correlations between actions and rewards. While it may still be possible for the agent to learn from many samples, such variations can disrupt its learning capabilities and hinder its ability to generalize effectively.

When a specific state offers multiple actions with identical immediate rewards, it is called ”action non-uniqueness.” This can further complicate decision-making, as the agent must navigate through states where multiple actions lead to the same immediate or final reward. To ensure convergence of the NN, we only selected state-action pairs to be included in the mini-batch samples that contributed to changes in the objective value. Thus, we dealt indirectly with the action ambiguity by discarding most of the state-action pairs that returned the same immediate reward, assuming being in a particular state.

5.2 Limitations

PPO is not well-suited for this MDP design because it updates its policy based on the most recent trajectory, whereas we require an algorithm that updates policy based on multiple trajectories, comparing similar states. The inherent randomness in the environment may overshadow good policies, leading to suboptimal outcomes. Although PPO may generate a good policy initially, fluctuations in the order distribution introduce additional noise in cumulative rewards, potentially degrading performance compared to a seemingly inferior policy that happened to benefit from favorable order-book conditions. DCL, on the other hand, aggregates similar states from different policies, making it a more robust method that is particularly effective in stochastic environments.

Considering an entire product portfolio can become complex due to the ”curse of dimensionality”. Adding NXP’s product portfolio creates a large state and action space. The method discussed in this paper only applies to small scenarios and can help manage competing products on a dual-source supply path. However, extending it requires modifications to the proposed current methodology.

This paper aims to replicate the scheduling modeling used at NXP using LP models with DRL. It is important to note that mathematical models and DRL are typically used for different types of problems. DRL is based on sequential decision-making and is suited for dynamic systems where not all information is known in advance and is instead revealed during simulation. Mathematical models make simultaneous decisions and are used for problems where all variables and options are known in advance. Given that our problem is restricted by the LP format of NXP and the stochastic elements are known in advance and do not change during simulation, it may be more effective to consider (meta-)heuristics that use mathematical formats but differ from MP models by searching the solution space more effectively, especially when the goal is to reduce run time. Meta-heuristics that use a mathematical format to search the solution space more effectively and reduce run-time include Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, Simulated Annealing, and Tabu Search (Salhi and Thompson 2022).

6 FUTURE RESEARCH

While this project has laid a foundational understanding of Deep Reinforcement Learning (DRL) within the context of a specific supply chain scenario, further enhancements of the simulation model are necessary for comprehensive applicability.

- Reducing the planning horizon from one year to a shorter period can potentially alleviate inefficiencies in the agent’s learning process. With fewer actions between the trajectory’s start and end, the agent could better discern optimal policies and mitigate state non-uniqueness.
- Rather than predefining a static order book, introducing order arrivals with varying customer priorities might create a more dynamic environment for the agent. Real-time reaction to each arrival potentially promotes adaptability and reduces repetitive scenarios.
- An immediate improvement involves adding an additional supply path for one of the two product types. Adding another path could potentially address the uniqueness problem without fundamentally changing the properties of the system. It is not expected for the uniqueness problem to be an issue

when expanding the model for real-life scenarios. However, the curse of dimensionality remains a significant issue.

REFERENCES

Banerjee, A. 2007. "Global trends in supply chain planning in semiconductor industry". *Supply Chains and Firm Performance* 53.

Bean, J. W., A. Devpura, M. O'Brien, and S. Shirodkar. 2005. "Optimizing Supply-Chain Planning.". *Intel Technology Journal* 9(3).

Degbotse, A., B. T. Denton, K. Fordyce, R. J. Milne, R. Orzell and C.-T. Wang. 2013. "IBM blends heuristics and optimization to plan its semiconductor supply chain". *Interfaces* 43(2):130–141.

Deng, Y., J. F. Bard, G. R. Chacon, and J. Stuber. 2010. "Scheduling back-end operations in semiconductor manufacturing". *IEEE Transactions on Semiconductor Manufacturing* 23(2):210–220.

Denton, B., J. Forrest, and R. J. Milne. 2006. "Methods for solving a mixed integer program for semiconductor supply chain optimization at IBM". *Interfaces* 36(5):386–399.

Horiguchi, K., N. Raghavan, R. Uzsoy, and S. Venkateswaran. 2001. "Finite-capacity production planning algorithms for a semiconductor wafer fabrication facility". *International Journal of Production Research* 39(5):825–842.

Jackson, I. 2022. "Deep reinforcement learning for supply chain synchronization".

Lu, W., H. Tan, X. Yan, and C. Lv. 2021. "Supply Chain Scheduling Using Double Deep Time-Series Differential Neural Network". In *E3S Web of Conferences*, Volume 257, 03038. EDP Sciences.

Mönch, L., C.-F. Chien, S. Dauzère-Pérès, H. Ehm and J. W. Fowler. 2018. "Modelling and analysis of semiconductor supply chains". *International Journal of Production Research* 56(13):4521–4523.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Salhi, S. and J. Thompson. 2022. "An overview of heuristics and metaheuristics". *The Palgrave Handbook of Operations Research*:353–403.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov. 2017. "Proximal policy optimization algorithms". *arXiv preprint arXiv:1707.06347*.

Temizöz, T., C. Imdahl, R. Dijkman, D. Lamghari-Idrissi and W. van Jaarsveld. 2020. "Deep Controlled Learning for Inventory Control". *arXiv e-prints*:arXiv–2011 https://doi.org/https://arxiv.org/abs/2011.15122.

Tseng, C.-Y., J. Li, L.-H. Lin, K. Wang, C. C. White III and B. Wang. 2023. "Deep reinforcement learning approach for dynamic capacity planning in decentralised regenerative medicine supply chains". *International Journal of Production Research*:1–16.

Ziarnetzky, T., L. Mönch, T. Ponsignon, and H. Ehm. 2019. "Integrated planning of production and engineering activities in semiconductor supply chains: A simulation study". In *2019 Winter Simulation Conference (WSC)*, 2324–2335. IEEE.

AUTHOR BIOGRAPHIES

BIBI DE JONG is a doctoral candidate in the Department of Information Systems at NXP Semiconductors and in the Department of Industrial Engineering and Innovation Sciences at the Eindhoven University of Technology. She obtained her master's degree in Operations Management & Logistics at the Eindhoven University of Technology cum laude. Her research interest lies in the application of machine learning to support supply chain operations. Her email address is bibi.de.jong@nxp.com.

RICCARDO LO BIANCO is a doctoral candidate in the Department of Industrial Engineering and Innovation Sciences at the Eindhoven University of Technology. He obtained a double master's degree in Data Science and Entrepreneurship at Politecnico di Milano and Polytech Nice Sophia. His research interests lie in the application of AI to process management and decision making. His email address is r.lo.bianco@tue.nl.

KAI SCHELTHOFF is the head of the Supply Chain Innovation team at NXP Semiconductors B.V.. He obtained his PhD at the Karlsruhe Institute of Technology about data-driven cycle time estimation in semiconductor wafer fabrication using a concatenated machine learning approach, in collaboration with Robert Bosch GmbH. He is an expert in machine learning applications for estimation, classification and optimization in Supply Chain Operations and Manufacturing. His email address is kai.schelthoff@nxp.com.

WILLEM VAN JAARSVELD is Associate Professor in Stochastic Optimization and Machine Learning at Eindhoven University of Technology (TU/e). His main research interest is stochastic optimization, using a diverse set of methodologies including Deep Reinforcement Learning, Stochastic Programming and Dynamic Programming. Applications areas include data-driven inventory control, production planning, supply chain management, and maintenance logistics. His email address is w.l.v.jaarsveld@tue.nl.