

Reinforcement Learning, Optimization and Scheduling in Industry 5.0: Overview and Case studies

Alberto Sinigaglia³, Riccardo De Monte³, Patrick Stöckermann⁴, Alessandro Immordino⁴, Ilayda Celenk¹, Willem van Jaarsveld¹, Ivo Adan¹, Eric Weijers², Cas Rosman², Bibi de Jong², Kai Schelthoff², Bram Hoekstra^{1,2}, Laura Leja⁵, Gian Antonio Susto³, Oskars Teikmanis⁵, and Karlis Freivalds⁵

Abstract—This chapter reviews recent advances in the application of Deep Reinforcement Learning (DRL) to industrial scheduling and supply chain optimization. We first introduce foundational RL concepts and leading DRL algorithms, as well as various other machine learning algorithms. A central component of the chapter are various case studies, where DRL and other methods are applied to company use cases. The chapter concludes with perspectives on scalability, deployment, and research opportunities within Industry 4.0 and 5.0 contexts.

Index Terms—Reinforcement Learning, Scheduling, Optimization, Industry 4.0, Industry 5.0, Supply Chain, Logistics, Operational Research

I. INTRODUCTION

Recent years have witnessed a surge in interest around Reinforcement Learning (RL) and other machine learning (ML) techniques for solving decision-making problems in complex industrial environments. In this chapter, we explore the use of ML—particularly Deep Reinforcement Learning (DRL)—as a paradigm-shifting approach to operations planning and scheduling, where conventional methods struggle to meet scalability, adaptability, and real-time responsiveness requirements.

This review is structured to offer both theoretical insights and practical applications. After introducing foundational concepts in DRL and ML, we provide a comparative overview of state-of-the-art methods. A key case study is presented in the context of semiconductor supply chain planning, where DRL has been applied to address limitations of traditional optimization models. Finally, we highlight promising research directions and implementation considerations for deploying DRL in real-world industrial settings.

This work was funded by the European Union in the context of the Horizon Europe project “AIMS5.0 - Artificial Intelligence in Manufacturing leading to Sustainability and Industry5.0” Grant agreement ID: 101112089.

¹ Eindhoven University of Technology, The Netherlands

² NXP Semiconductors N.V., The Netherlands

³ University of Padua, Italy

⁴ Infineon Technologies AG, Germany

⁵ EDI - Institute of Electronics and Computer Science, Latvia

II. OVERVIEW OF METHODS

A. Introduction to Reinforcement Learning

Reinforcement Learning deals with a two-entity framework, the agent and the environment. With reference to Fig. 1, the environment dictates the dynamics, and the agent acts on it to maximize a specific reward function. This setting is usually formalized under the umbrella of Markov Decision Process (MDP). An MDP consists of five main parts:

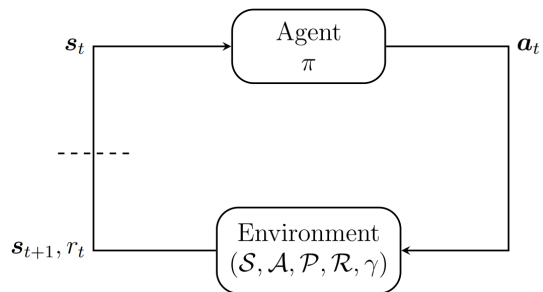


Fig. 1. Agent-environment interaction in a Markov Decision Process.

- 1) $\mathcal{S}$, called state space, is the set of states in which a reinforcement learning agent can be.
- 2) $\mathcal{A}$, called action space, is the set of actions that an agent can perform;
- 3) $\mathcal{P}$, called the transition function, rules the dynamics and is defined as $p(s_{t+1}|s_t, a_t)$. For the rest of the chapter, we will consider the discretized time RL setting, which transition from the current state and the new state will be dictated by the transition distribution;
- 4) $\mathcal{R}$, called the reward function, which acts as the supervision in the RL problem, and is defined as $p(r_t|s_t, a_t, s_{t+1})$. Poorly speaking, it defines the immediate reward an agent should receive given a certain decision;

- 5) γ , called the discount factor, which rules the trade-off between the importance of the immediate reward and the future rewards.

The behavior of an agent is identified in a policy that defines how the agent acts in a specific state. It is defined as a distribution $\pi(a_t|s_t)$. The support for such a distribution can change between states. Such policy aims at maximizing the return (expected cumulative sum of discounted rewards), defined as:

$$G(\pi) = \mathbb{E}_{s_0, a_0, s_1, \dots}^{\pi} \left[\sum_{i=0}^{\infty} \gamma^i r_i \right]$$

Given a specific policy π and discount factor γ , we can define two other quantities, the value function $V^{\pi} : \mathcal{S} \rightarrow \mathbb{R}$ and the action-value function $Q^{\pi} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$:

$$V^{\pi}(s_t) = \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, s_{t+2}, \dots}^{\pi} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \mid s_t \right]$$

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, s_{t+2}, \dots}^{\pi} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \mid s_t, a_t \right]$$

All RL algorithms can be derived from such quantities, however, an additional one is defined since is extensively used in literature, which is the Advantage A , which is defined as:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

Intuitively, it models the improvement obtained thanks to action a_t from the policy π compared to the average return in such state.

The challenge in RL is learning these functions in order to derive a policy improvement algorithm. In the naive setting with a finite set of possible states and actions, and these sets being reasonably small, a tabular setting is usually employed, where the table represents all possible combinations of states and actions $\mathcal{S} \times \mathcal{A}$. However, in real world scenario, the size of these sets makes such approach prohibitive, or impossible in the case of continuous settings. In order to solve this criticality, RL usually relies in approximations of these functions.

All the algorithms presented in the following sections, will take advantage on a subset of these tools ($\pi, \mathcal{A}, \mathcal{V}, \mathcal{Q}$) approximated with some function approximator, which usually are Neural Networks, in which case are referred as Deep Reinforcement Learning (DRL) algorithms.

Most of recent advancements in DRL fall under the umbrella of Q-learning algorithms, and Actor Critic algorithms. The former aims at distilling a policy directly from the Q-function, this requiring the learning of only that component. Actor Critic algorithms instead aim at learning the policy π , and exploit either $\mathcal{A}$ or $\mathcal{V}$ or $\mathcal{Q}$ to do so.

B. State of the Art of Deep Reinforcement Learning

Having introduced the fundamental concepts of Reinforcement Learning (RL)—value functions (V^{π}), action-value functions (Q^{π}), advantage functions (A^{π}), and stochastic or deterministic policies (π, μ)—we now focus on the principal deep RL algorithms that have driven recent progress in the field. Each of the following methods leverages neural networks to

approximate one or more of the core functions, and incorporates various techniques to stabilize learning and improve performance.

1) *Deep Q Networks*: Deep Q Networks (DQN) [1] were among the first successful attempts to combine deep learning with Q-learning, enabling agents to learn directly from high-dimensional inputs such as raw pixels in Atari games.

a) *Objective*.: Let $Q_{\theta}(s, a)$ be the parameterized action-value function (Q-function) approximated by a neural network with parameters θ . DQN minimizes the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s, a, s', r) \sim \mathcal{D}} \left[(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_{\theta}(s, a))^2 \right], \quad (1)$$

where θ^- are the parameters of a slowly updated target network and (s, a, s', r) are transitions sampled from a replay buffer $\mathcal{D}$, where agent's experiences are stored. The replay buffer improves data efficiency, namely, the same transition can be used for multiple gradient updates, and decorrelates samples.

b) *Target Network and Double DQN*.: The use of a second target network, namely keeping a distinct set of parameters for the Q-learning target in equation 1, has shown improved training stability. Double DQN [2] further improves DQN performance by proposing the following target:

$$r + \gamma Q_{\theta^-}(s', \arg\max_{a'} Q_{\theta}(s', a')) \quad (2)$$

This allows reducing overestimation of the target by decomposing the max operation in the target into action selection and action evaluation.

2) *REINFORCE*: Instead of learning the action-value functions (Q^{π}) and derive a greedy policy from it, Policy Gradient methods, as REINFORCE [3], directly optimize the parameters θ of the policy $\pi_{\theta}(a|s)$ by sampling entire trajectories from the environment. Such an approach is effective in high-dimensional or continuous action spaces, contrary to DQN-based methods.

To do so, REINFORCE seeks to maximize the expected return $J(\theta)$ defined as:

$$J(\theta) = G(\pi_{\theta}) = \mathbb{E}^{\pi_{\theta}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \quad (3)$$

where T is the length of an episode (assuming an episodic setting where for any trajectory ends in a final state S_T). The gradient of $J(\theta)$ with respect to θ is given by the so-called Policy Gradient (PG) Theorem [4]:

$$\nabla_{\theta} J(\theta) = \mathbb{E}^{\pi_{\theta}} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) G_t \right], \quad (4)$$

$$G_t = \sum_{k=t}^{T-1} \gamma^k r_{t+k}$$

with G_t representing the return (or discounted return) following time step t . In practice, to mitigate high variance, REINFORCE implementations subtract to G_t a learned baseline e.g. the value function $V^{\pi_{\theta}}(s_t)$.

3) *Advantage Actor Critic (A2C / A3C)*: Actor-Critic methods combine policy-gradient (the *actor*) with a value function estimator (the *critic*). Actor-Critic methods estimate the advantage function $A^{\pi_{\theta}}(s_t, \mathbf{a}_t)$ by bootstrapping. In particular, provided an estimate of the value function for any state $\hat{V}^{\pi_{\theta}}(s)$, an estimate of the advantage $A^{\pi_{\theta}}(s_t, \mathbf{a}_t)$ is given by the so-called k -steps Temporal Difference error (TD(k)) error:

$$\hat{A}^{\pi_{\theta}}(s_t, \mathbf{a}_t) = \sum_{i=0}^k \gamma^i r_{t+i} + \gamma^{k+1} \hat{V}^{\pi_{\theta}}(s_{t+k+1}) - \hat{V}^{\pi_{\theta}}(s_t) \quad (5)$$

Advantage Actor Critic methods usually estimate $V^{\pi_{\theta}}(s)$ by training a neural network as a function approximator. Therefore, by denoting with ϕ the learnable parameter of such a network, called *Critic network*, $\hat{V}_{\phi}^{\pi_{\theta}}(s)$ is used as an estimate of the value function. As consequence, Advantage Actor Critic approaches jointly update the Actor network (π_{θ}) and the Critic one. As for REINFORCE, policy parameters θ are updated according to PG:

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | s_t) \hat{A}^{\pi_{\theta}}(s_t, \mathbf{a}_t) \right] \quad (6)$$

Regarding the Critic network, the parameters are updated by minimizing the Mean Squared Error (MSE) between the current estimate $\hat{V}^{\pi_{\theta}}(s)$ and the TD(k) target. In the simple case where $k = 0$, the loss for the Critic is given by:

$$\mathcal{L}(\phi) = \mathbb{E} \left[\left(r_t + \gamma \hat{V}^{\pi_{\theta}}(s_{t+1}) - \hat{V}^{\pi_{\theta}}(s_t) \right)^2 \right] \quad (7)$$

where $r_t + \gamma \hat{V}^{\pi_{\theta}}(s_{t+1})$ is the TD(1) target at given state s_t .

a) *Synchronization.*: In the literature, there are two popular implementations for Advantage Actor Critic: Asynchronous Advantage Actor Critic (A3C) [5] and the synchronous variant A2C. In A3C, multiple agents explore in parallel on different threads, asynchronously updating the shared parameters, while for A2C updates are collected from parallel actors and applied in a batch.

4) *Proximal Policy Optimization (PPO)*: In Reinforcement Learning, as in traditional Machine Learning, the choice of the learning rate is crucial. A small learning rate often requires multiple updates on the same samples. However, in RL, this can lead to large policy updates that are not justified, as the advantage function used in the Policy Gradient update depends on the policy itself.

To address this, TRPO [6] introduces a surrogate objective function along with a constraint to prevent excessively large updates. However, TRPO is challenging to implement in frameworks like PyTorch and is incompatible with techniques like Dropout or parameter sharing between the critic and the policy.

To overcome these limitations, PPO [7] introduces a clipped surrogate objective, making it both simpler to implement and more broadly applicable while still controlling policy updates. Defined the following likelihood ratio:

$$r_t(\theta) = \frac{\pi_{\theta}(\mathbf{a}_t | s_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | s_t)}, \quad (8)$$

where $\pi_{\theta_{\text{old}}}$ denotes the policy used to sample, the clipped surrogate objective (the most used PPO objective) is:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E} \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right], \quad (9)$$

where $\hat{A}_t = \hat{A}^{\pi_{\theta_{\text{old}}}}(s_t, \mathbf{a}_t)$ is the estimate of the advantage at time step t concerning the sampling policy $\pi_{\theta_{\text{old}}}$ (as done for A2C/A3C, a critic network is used for this purpose), and ϵ is a hyperparameter (typically small, e.g., 0.1 or 0.2). This *clipping* prevents excessively large policy updates while preserving incentive to improve on-policy actions.

5) *Deep Controlled Learning*: Deep Controlled Learning (DCL) [8] is a Deep Reinforcement Learning (DRL) algorithm tailored to high-variance environments such as inventory management and supply chain scheduling. It operates within the Approximate Policy Iteration (API) framework, where the goal is to iteratively improve a policy π by approximating the optimal value function and deriving improved policies.

Unlike conventional DRL algorithms, DCL decouples the learning of the value function from policy learning. It focuses on the policy learning step as a supervised classification task:

$$\pi(s) = \arg \min_{a \in \mathcal{A}} \mathbb{E}[c(s, a)]$$

where $c(s, a)$ is the estimated cost of taking action a in state s , obtained via simulation.

To estimate $\mathbb{E}[c(s, a)]$ efficiently and robustly, DCL combines two algorithmic techniques:

- **Sequential Halving (SH)**: An adaptive sampling method that allocates simulation budget to promising actions, discarding suboptimal ones early.
- **Common Random Numbers (CRN)**: A technique that reduces variance by using shared randomness across simulations for different actions, ensuring fair comparisons.

Given a set of actions $\mathcal{A}$, each action a is simulated under identical scenarios $\omega \in \Omega$, and the empirical mean cost $\hat{c}(s, a)$ is computed. The selected action minimizes this estimate:

$$\pi(s) = \arg \min_{a \in \mathcal{A}} \hat{c}(s, a)$$

This process is repeated for each state in a dataset, and a policy neural network is trained to predict the optimal action given any state.

DCL's design ensures scalability and stability. Empirical studies have shown that DCL performs consistently well across diverse inventory control scenarios—including lost sales, perishable goods, and random lead times—achieving average cost reductions with an optimality gap below 0.2% compared to known optimal policies.

The algorithm's generality, low variance, and robust performance with fixed hyperparameters make it a compelling choice for dynamic supply chain and inventory planning under uncertainty.

6) *Preconditioned Proximal Policy Optimization (P3O)*: While the clipping PPO objective aims to prevent large policy

updates, it might cause loss of exploration. For this reason P3O [9] proposes the following objective:

$$\mathcal{L}^{\text{p3o}}(\theta) = \mathbb{E} \left[\mathcal{L}^{\text{scp}}(\theta) - \beta \cdot \text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_{\theta}(\cdot|s_t)) \right] \quad (10)$$

$$\mathcal{L}^{\text{scp}}(\theta) = \sigma \left(\tau \cdot (r_t(\theta) - 1) \right) \cdot \frac{4}{\tau} \hat{A}_t \quad (11)$$

where τ and β are two hyperparameters, $\sigma(\cdot)$ denotes the sigmoid function, $\text{KL}(\cdot, \cdot)$ denotes the Kullback–Leibler (KL) divergence and $r_t(\theta)$ denotes the likelihood ratio defined in equation 8. $\mathcal{L}^{\text{scp}}(\theta)$ is called *Scopic objective* and, contrary to the clipped surrogate objective used in PPO, allows the policy to be optimized in a much larger space promoting exploration. On the other hand, the KL divergence keeps the importance sampling ratio $r_t(\theta)$ close to one, to better balance exploration and exploitation.

7) *Deep Deterministic Policy Gradient (DDPG)*: DDPG [10] extends Q-learning to continuous actions by following the Actor-Critic paradigm, namely, maintaining both a parameterized critic (Q-function) $Q_{\phi}(s, a)$ and a parameterized, deterministic actor $\mu_{\theta}(s)$. As for DQN, DDPG exploits a replay buffer $\mathcal{D}$, where samples (s, a, r, s') are collected according to the exploration policy $\mu(s) = \mu_{\theta}(s) + \mathcal{N}$, where $\mathcal{N}$ is any random noise, e.g. Gaussian noise.

a) *Critic Update.*: Similar to DQN, DDPG uses a target network for both Q , Q_{ϕ^-} , and μ , μ_{θ^-} . These are updated with an Exponential Moving Average update, and allows to derive a target for the critic that guarantees stable updates:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[(r + \gamma Q_{\phi^-}(s', \mu_{\theta^-}(s')) - Q_{\phi}(s, a))^2 \right]. \quad (12)$$

b) *Actor Update.*: Following the Deterministic Policy Gradient Theorem [11], the deterministic policy gradient is computed by backpropagating through the Q-function with respect to the policy:

$$\nabla_{\theta} J(\mu_{\theta}) = \mathbb{E}_{s \sim \mathcal{D}} \left[\nabla_a Q_{\phi}(s, a) \big|_{a=\mu_{\theta}(s)} \nabla_{\theta} \mu_{\theta}(s) \right]. \quad (13)$$

8) *Twin Delayed Deep Deterministic Policy Gradient (TD3)*: TD3 [12] improves upon DDPG by mitigating overestimation bias in Q-learning. It uses two critics and delays policy updates, leading to more stable training.

a) *Overestimation Mitigation.*: When computing the target, TD3 takes the minimum of the two critic networks:

$$y = r + \gamma \min_{i=1,2} Q_{\phi_i^-}(s', \mu_{\theta^-}(s') + \epsilon), \quad (14)$$

where ϵ is a small noise added for smoothing (target policy smoothing). The critics are updated by minimizing the difference to this target, and the policy update is performed less frequently (e.g., every two critic updates).

9) *Soft Actor Critic (SAC)*: SAC [13] incorporates an entropy term to encourage broader exploration. Instead of simply maximizing the expected return, SAC optimizes a *soft* value function that balances return and policy entropy.

a) *Soft Q Function.*: The soft Q-function $Q_{\phi}(s, a)$ is learned by minimizing

$$L_Q(\phi) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[(Q_{\phi}(s, a) - (r + \gamma V_{\bar{\psi}}(s')))^2 \right], \quad (15)$$

where $V_{\bar{\psi}}(s)$ is a parameterized value function network (or, in some variants, omitted in favor of a second Q-network).

b) *Entropy-Augmented Policy Update.*: The policy $\pi_{\theta}(a|s)$ aims to minimize the Kullback–Leibler divergence between itself and an exponential distribution defined by the soft Q-function. One way to write the objective is:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_{\theta}} \left[\nabla_{\theta} \alpha \log \pi_{\theta}(a|s) - \nabla_{\theta} Q_{\phi}(s, a) \right], \quad (16)$$

where α is the temperature parameter that controls the trade-off between exploration (entropy) and exploitation (return). An adaptive scheme for α can also be employed to dynamically adjust the entropy bonus.

c) *Advantages of SAC.*: By emphasizing entropy in the objective, SAC avoids prematurely collapsing to a near-deterministic policy, often leading to more robust and stable learning across a wide range of continuous control environments.

Overall, these algorithms showcase the evolution of deep RL, illustrating how different design choices—on-policy vs. off-policy, deterministic vs. stochastic policies, clipping vs. entropy regularization, or single vs. multiple critics—can greatly affect training stability and performance. While each method has its own strengths and trade-offs, together they form the core toolkit for solving complex RL tasks in both simulation and real-world applications.

C. Introduction to Evolutionary Methods

Evolutionary methods are optimization techniques inspired by natural evolution, emulating concepts such as mutation, selection, and recombination. These methods iteratively evolve candidate solutions to optimize complex, noisy, or non-differentiable functions. Unlike gradient-based optimization, evolutionary strategies do not rely on gradient information, thus providing robustness in scenarios where gradient information is either unavailable or unreliable.

At their core, evolutionary algorithms maintain a population of candidate solutions. Through repeated cycles of evaluation and variation, they progressively improve these solutions by favoring those exhibiting superior performance, typically measured by a fitness function. A general evolutionary approach includes the following steps:

- 1) **Initialization**: Generate an initial population of candidate solutions.
- 2) **Evaluation**: Compute the fitness of each candidate solution.
- 3) **Selection**: Select a subset of candidates based on fitness to form a new generation.
- 4) **Variation**: Apply mutation and recombination to create offspring solutions.
- 5) **Replacement**: Replace less fit solutions with better-performing offspring.

Contrary to traditional Reinforcement Learning (RL) approaches, Evolutionary Strategies (ES) do not rely on gradient estimates. Instead they treat the optimization of policies as a black-box optimization problem. This makes ES particularly suited for scenarios where the reward signals are sparse, delayed, or noisy, conditions frequently encountered in industrial scheduling and real-world RL problems. Prominent evolutionary methods, such as the Cross-Entropy Method (CEM), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Separable Natural Evolution Strategies (SNES), have demonstrated their efficacy in optimizing neural network policies directly, effectively handling high-dimensional and complex policy spaces.

D. State of the Art Evolutionary Methods

Modern evolutionary optimization methods have significantly advanced, providing powerful alternatives to classical optimization and reinforcement learning in complex industrial applications. Among these, three notable state-of-the-art methods include Cross-Entropy Method (CEM), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Separable Natural Evolution Strategies (SNES).

Each of these evolutionary strategies addresses different optimization challenges. CMA-ES stands out for its robustness in noisy, correlated parameter spaces, CEM offers simplicity and interpretability, and SNES excels in scalability and computational efficiency. Collectively, these methods form the cornerstone of evolutionary optimization, enabling effective exploration and exploitation in complex real-world scenarios.

1) *Cross-Entropy Method (CEM)*: Initially proposed for rare-event simulation and subsequently adopted for optimization, CEM is an iterative procedure optimizing a parameterized probability distribution of solutions. At each iteration, CEM samples solutions from a parameterized distribution (typically Gaussian), selects elite samples based on their performance, and updates the distribution parameters by minimizing the cross-entropy distance between the sampling distribution and the distribution of elite samples. Formally, the update rule for mean μ and covariance Σ can be expressed as:

$$\mu_{t+1} = \frac{1}{N_e} \sum_{i=1}^{N_e} x_i, \quad \Sigma_{t+1} = \frac{1}{N_e} \sum_{i=1}^{N_e} (x_i - \mu_{t+1})(x_i - \mu_{t+1})^T, \quad (17)$$

where x_i are elite samples, and N_e denotes the number of elite samples. The simplicity and adaptability of CEM make it appealing for problems where gradient computation is infeasible or costly, although it can be prone to premature convergence in multimodal landscapes.

2) *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)*: CMA-ES represents a significant advancement in evolutionary methods, particularly adept in handling continuous optimization problems with noisy fitness landscapes. CMA-ES explicitly models dependencies among variables by adaptively updating a covariance matrix of a multivariate Gaussian distribution. Formally, CMA-ES updates the mean and covariance matrix using information from elite solutions:

$$\begin{aligned} \mu_{t+1} &= \mu_t + \sigma_t \sum_{i=1}^{N_e} w_i (x_i - \mu_t), \\ \Sigma_{t+1} &= \sum_{i=1}^{N_e} w_i (x_i - \mu_t)(x_i - \mu_t)^T, \end{aligned} \quad (18)$$

where w_i are weights assigned according to the rank-based fitness of the samples x_i . This allows CMA-ES to adaptively adjust the step-size and the search direction, significantly enhancing its efficiency and robustness against ill-conditioned objective functions and improving convergence speed and robustness.

3) *Separable Natural Evolution Strategies (SNES)*: Separable Natural Evolution Strategies (SNES) simplify the computational complexity of covariance-based methods like CMA-ES by assuming a diagonal covariance structure. This method optimizes a Gaussian distribution characterized by its mean and a diagonal covariance, significantly reducing the number of parameters to be updated. SNES maximizes expected fitness through a gradient estimate derived from natural gradients:

$$\theta_{t+1} = \theta_t + \eta F^{-1} \nabla_{\theta} \mathbb{E}_{\theta}[f(x)], \quad (19)$$

where $\theta = (\mu, \Sigma)$ are the parameters of the Gaussian distribution, and the diagonal covariance ensures scalability to high-dimensional problems. SNES is particularly effective in large-scale optimization tasks where computational efficiency is critical.

E. Introduction to Differentiable Physics

Gradient-based learning with differentiable simulations, or differentiable physics (DP), is a growing field of machine learning that takes advantage of direct gradient propagation and offers significant improvements in sample efficiency, training stability, and computational performance in complex control tasks, compared to reinforcement learning [14], [15], [16], [17].

1) *Foundational Concepts*: Differentiable simulators process physical dynamics as end-to-end differentiable computational graphs [14]. Unlike conventional simulators, which treat state transitions such as

$$s_{t+1} = f(s_t, a_t)$$

as black-box functions, differentiable implementations enable gradient calculation through the state transition process. This allows for a direct computation of expressions such as

$$\frac{\partial s_{t+1}}{\partial a_t} \quad \text{and} \quad \frac{\partial s_{t+1}}{\partial s_t},$$

creating a continuous optimisation pathway from actions to outcomes. For example, in robotic manipulation tasks, this property enables backpropagation of positioning errors through the entire sequence of manipulation tasks.

In contrast, reinforcement learning (RL) generally relies on stochastic policy exploration and reward signal propagation, requiring extensive environmental interactions to estimate value functions. The policy gradient approach of the

REINFORCE algorithm [3] exemplifies this trial-and-error paradigm, where agents must discover successful action sequences through temporal difference learning and Monte Carlo sampling. Although effective in diverse domains, this methodology becomes prohibitively expensive in scenarios with high-dimensional state spaces or delayed rewards. Gradient-based learning circumvents RL's exploration bottleneck by using analytic gradients from differentiable models to directly optimise policy parameters.

F. State of the Art in Differentiable Physics (EDI)

1) *Overview of Differentiable Simulator Toolkits:* Early work on DP for control by Belbute-Peres et al. [18] shows that end-to-end differentiability with analytical gradients comes with a number of benefits, including parameter identification and sample-efficient learning by embedding differentiable physics within deep networks for more accurate predictions of physical behaviours from minimal observational data across diverse applications, including fluid-flow dynamics [19], [20], [21], [22], [23], and robotics tasks [24], [25]. Since then, there has been a rise of diverse differentiable simulation engines over the past years, including tools such as Brax [26], Dojo [27], NimblePhysics [28], Taichi [29] and Warp [30]. An overview of their features can be seen in Table I.

Compared to general-purpose ML libraries such as TensorFlow and PyTorch, differentiable simulation toolkits feature diverse physics solvers and may come with optimisation and learning tools or easy integration with existing machine learning libraries. For example, BRAX has Built-in RL algorithms in JAX, while Warp and Nimble can be easily integrated with PyTorch or Tensorflow.

2) *Applications in Robotic Manipulation Tasks:* Bridging the reality gap is one of the most important steps in robotics in order to operate in the real world. One method for accomplishing this is by parameter identification. Lutter et al. [17] demonstrate that knowledge of physics can significantly reduce the required number of measurement samples for achieving accurate control. This is accomplished by comparing the offline learning performance of an analytically differentiable white-box model of a robotic arm with black-box models of the same robot in a ball-in-cup task. The authors find that the white-box model consistently perform better than black-box models, despite being trained on significantly less data, while also complying with nonholonomic constraints (e.g. ball-on-string). Collins et al. [31] demonstrate similar results in an online learning setting by iterating through real-to-sim optimisations and sim-to-real transfers for continuous improvements of the robot model.

Accurate contact modelling is another important requirement for contact-rich robotic manipulation tasks. This is a challenge, since contact simulations are inherently difficult to differentiate due to potentially numerous discontinuities. Still, even these challenges can be addressed with differentiable physics by either mollifying (softening) contact forces [32], [24], [25], integrating collision detection and contact simulation into one unified differentiable framework [33] and introducing adaptive-horizon rollouts to limit gradient errors

from stiff contact dynamics within differentiable simulations [34].

III. DRL AND OPTIMIZATION: APPLICATION AREAS IN INDUSTRY 5.0

DRL and Optimization are broadly applied in Industry 4.0 and Industry 5.0, e.g. in the areas:

- Supply chain and logistics
- Autonomous robotic systems and adaptive control
- Energy and resource management
- Quality control, predictive maintenance
- Collaborative robots, adaptive workflows
- Resilient production systems

Next, we present a detailed discussion of applications in specific use cases:

- Wafer Lot Scheduling in Semiconductor Frontend Fabs
- Master Planning for Semiconductor Supply Chain Simulation
- Reel Utilization in Production by 3D printing
- Long-Term Rapid Scenario Planning in the Semiconductor Industry using DRL
- Differentiable physics training method for robot motion planning

A. Wafer Lot Scheduling in Semiconductor Frontend Fabs

1) *Problem Description:* Semiconductor Frontend Manufacturing (SCFM) is one of the most complex industrial processes in the world. Silicon wafers are going through an intricate process flow which can entail hundreds of steps and can take multiple months. Those steps can include diffusion, etching processes, lithography, ion implementation, measuring steps and more. During those processes, the intended electrical structure is build layer by layer on the wafer. After the wafer processing in the frontend of line, the wafers are tested and move on to the backend of line where they are cut into piece, packaged and bonded to create the interconnections between the integrated circuit (IC) and its packaging. As the tools are very expensive and can cost many millions of USD each [35], we want to utilize them as much as possible. In order to achieve a high utilization, we accept queues of waiting lots in front of the machines. To achieve good WIP (Work-In-Progress)-flow and balancing of load across the routes of the different products as well as across the machines, the queues of waiting lots have to be optimized. [36]

This optimization is described by the Semiconductor Factory Scheduling Problem (SFSP), which is regarding the efficient assignment of wafer lots to specific machines in a designated order. The problem is often formalized as a Complex-Job-Shop-Scheduling Problem (CJSSP) in academia. The CJSSP is based on the Job-Shop-Scheduling Problem (JSSP), which describes the efficient assignment of n jobs $\mathcal{J}$ to m machines $\mathcal{M}$. In the basic version, each job of a unique type has to be processed once on each machine in a specific order. The CJSSP extends this problem by introducing machine groups with parallel machines. Furthermore, batching processes are considered, where multiple jobs can be processed

TABLE I
FEATURES OF DIFFERENTIABLE SIMULATION TOOLKITS.

Feature/Engine	BRAX	Dojo	Nimble	Taichi	Warp
Physics domain	Particle/Rigid/CFD	Robotics	Robotics	Multi-domain	Rigid/Soft
Differentiation	JAX AD	Implicit diff.	Chain rule AD	Kernel-grad AD	CUDA AD
Contact handling	Convex approx.	NCP/SOC	LCP	Hybrid	Compliant
Contact differentiation	Smooth approx.	Analytic	Limited	TOI-Velocity	Semi-analytic
Acceleration	GPU/TPU	GPU/CUDA	CPU/GPU	GPU	GPU

on the same machine in parallel if their setup requirements are matching. The CJSSP also incorporates the concept of reentrant flow, meaning that the jobs can revisit machines multiple times in their process flow. [36]

An important intrinsic property of SCFM is the strong stochasticity. Frequent machine failures are normal in day to day operations and directly affect the SFSP. To prevent unscheduled downtime, some manufacturers do preventive maintenance. The waiting times of wafer lots at individual steps, transport times as well as processing times are stochastic. Quality checks can trigger optional rework steps. [36]

2) *Heuristics*: Traditionally, simple heuristics, often in an hierarchical order, have been used in the SCFM industry to control the dispatching order of queues of waiting lots. Those can be as simple as First-In-First-Out (FIFO). This approach is good for comprehensibility as well as keeping a good overviews. The system is however limited to manual human intervention when it comes to dynamically reacting to arising bottlenecks or other material-flow related problems.

Researchers have presented solutions to optimize metaheuristics to better fit specific problem instances. This can be done by use of WIP-flow simulations. Linear combinations of weighted lot properties can be used to score and sort queues of weighting lots. The coefficients of that linear combination can be optimized though iterative perturbation and simulation runs.

An alternative approach in academia is the use of population-based metaheuristics called swarm algorithms. Some popular example are the Artificial Bee Colony (ABC) swarm algorithm as well as Ant Colony Optimization (ACO) which aim at a decentralized optimization. The idea is to observe properties in the neighboring environment, such as the utilization of the machines in the following steps. [37] and [38] show the effectiveness of these algorithms in smaller scheduling instances regarding semiconductor manufacturing. Another population-based metaheuristic approach are Evolutionary Algorithms (EAs) such as Genetic Algorithms (GAs), which have been shown to be effective for complex scheduling problems [39].

3) *Classical Mathematical Optimization*: Scheduling problems are often solved with exact mathematical methods such as Constraint Programming (CP), Mixed-Integer-Programming (MIP), Linear Programming (LP) or Answer-Set-Programming (ASP) [40], [41], [42], [43], [44], [45]. Those methods aim at finding the exact best solution or to at least estimate the solution gap. One of their advantages is that they can dynamically adapt to the local situation, i.e., to rebalance the

load across parallel machines.

As already the simple JSSP is NP-hard [46], the SFSP is often decomposed into subproblems, such as the Parallel-Machine-Scheduling-Problem (PMSP), which considers only the queue of waiting lots in front of a set of parallel machines of the same type. An acceptable solution has to be found by the solver within a specified interval time. This interval times depends on the dynamics of the manufacturing environment which can require rescheduling from one minute to the other if, i.e., tools are going down or rework has been triggered. An alternative is to accept strong abstractions of the problem, not considering all constraints in detail and disregarding important aspects such as stochasticity.

4) *Deep Reinforcement Learning*: Similar to metaheuristics, DRL can iteratively optimize a scheduling policy using a simulation based approach. This enables the direct optimization towards global fab level KPIs. Furthermore, DRL enables the training of a dynamic policy which is changing based on the state of the environment. A metaheuristic can learn a policy defined by a static linear combination of input features which works well on average. A stochastic policy trained by RL can be defined by a Deep Neural Network and dynamically change a nonlinear combination of input features to adapt to the dynamically changing situation in the manufacturing environment which is subject to an every changing load mix and stochastic machine failures.

One of the main challenges in DRL is the problem formulation through the characterization of the main components of the DRL paradigm, namely reward signal, state and action space. The choice of a suitable DRL algorithm is heavily influenced by this preliminary settings. In the context of lot scheduling in semiconductor manufacturing, the main objectives are usually makespan minimization [47], [48], utilization [49] or on time delivery [50]. Thus, the reward signal is usually related to these metrics. In this context, the state provided to the agent is dependent on the type of event that triggers the interaction between agent and simulation. A discrete event simulation may either halt after a predefined time interval or when a machine under the agent's control becomes idle. Regarding the former approach, some studies have incorporated an embedded graph structure into the observation space [47], [51], [52], which represents the relationships between jobs and machines or among the machines.

Alternatively, other approaches represent the state using multiple matrices [53], [54], [55], which define job-to-machine assignments, job processing times per machine, or the remaining processing times for each machine. In case the simulation

dynamically presents jobs waiting in front of an idle machine to the agent, the extracted features are typically more closely related to the characteristics of the lots [56], [49].

In this setting, the action space can be defined as assigning the position of the lot in the queue. Alternatively, actions can be defined as selecting or weighting predetermined dispatching rules commonly used in manufacturing contexts, such as shortest processing time (SPT) or first-in first-out (FIFO) [50], [54]. Another approach is to define actions space as the set of possible job-to-machine allocations [53], [52].

Once the main components of the MDP are defined, the choice of RL algorithm depends heavily on the complexity of the underlying simulation. If the simulation models a real semiconductor manufacturing process, the computational time required for the agent to interact with the simulator can be significantly higher compared to commonly used open-source benchmark models. In such cases, on-policy methods like PPO may be more suitable, as they compute updates on the fly without storing samples in a replay buffer. Asynchronous policy updates, such as in the A3C algorithm, can be beneficial when using multiple agents in parallel, as it allows for gradient calculation to begin without waiting for all agents to finish collecting their samples by interacting with the environment. This is particularly useful, as parallel agents can potentially complete episodes at very different times, i.e., due to stochastic events, such as machine breakdowns. Table II summarizes the major contributions regarding RL methods in the field of scheduling and dispatching which are cited in this section.

One limitation of RL approaches is the comprehensibility of the result, as the results have to be communicated to experts in the manufacturing domain who might not have in-depth knowledge of Deep Learning. Even for data scientists, the resulting policies are not fully explainable and can only be understood approximately. There is a tradeoff between the comprehensibility as well as the performance of the methods. Explainable Reinforcement Learning (XRL) is a growing field that aims to make RL agents more understandable and interpretable to humans by providing insights into the inner workings of an agent's policy. Several methods have been developed to address this challenge. One approach involves visualizing which input feature were most influential for the agent decision, i.e., with saliency maps for image based observations [57]. Another strategy is imitation-based explanations, where interpretable models such as decision trees or symbolic rules are trained to mimic the behavior of a more complex policy. Reward decomposition is another powerful technique [58]. Instead of treating the reward as a single scalar signal, it can be broken down into human understandable components that reveal what aspects of the environment the agent is prioritizing [59].

5) *Evolution Strategies / Neuroevolution*: One of the main problems in RL is the credit-assignment problem. It is difficult to find a reward which offers direct feedback and correlates with the actual target KPIs. An individual scheduling or dispatching decision might have long lasting effects which cannot be seen in the immediate returns of the reward. Furthermore, the many stochastic effects and dynamic nature of the system overlay the reward signal with noise. Lastly, the environment

is practically not fully observable if we consider all relevant constraints and details. In case we are using only a partial observation of the environment's state, we violate the Markov-Property.

We can circumvent all of the above problems, especially the credit-assignment problem, by using a black-box-optimization-algorithm which only requires one reward at the end of an episode. In this case we do not have to focus on reward shaping but we can directly return the KPIs of interest at the end of each iteration. [7] have shown Evolution Strategies (ES) to be a scalable alternative to policy-gradient-based RL. It is able to optimize smaller Neural Networks without the intrinsic problems of policy-gradient methods described above. It can be highly parallelized and offers good scalability with the number of utilized CPU cores. by combining ES with Neural Networks, it becomes a method within the area of Neuroevolution. CMA-ES is an even more effective enhancement of ES which takes into account the covariance among the parameters under optimization. [56] show its effectiveness for the SFSP in a real world industrial setting.

B. Master Planning for Semiconductor Supply Chain Simulation

1) *Problem Description*: Semiconductors are essential components in modern electronics, such as computers or smartphones. Their manufacturing is intricate, involving multiple stages that require precise coordination and efficient planning across various production sites. To manage this complexity, supply chain master planning is employed to coordinate resources and optimize supply, production, and distribution to meet demand effectively. Traditionally, this planning relies on linear programming (LP) models which are unable to deal with uncertainty. To analyze the effects of uncertainty on the supply chain, rapid event simulation is performed, where a host of different scenarios are simulated and their effects can be analyzed. A supply chain simulation model needs a policy model in the simulation, to determine the quantities of each product that should be produced in each period. Thus a policy model needs to be able to solve the production inventory control problem. The specific production inventory control problem addressed in this section deals with capacity sharing, dual sourcing systems. This means that each product can be produced using multiple different combinations of machine groups, and these combinations may share capacities. Furthermore, since rapid event simulation is the goal, LP cannot be used as a policy model, as a single LP run can take several hours, and in a simulation dozens of sequential LP runs would need to be performed. A good policy model needs to have short computation times, such that a wide range of scenarios can be explored, whilst approximating optimal solutions at the same time, to reflect reality.

2) *Heuristics*: A wide range of algorithms has been developed to achieve good solutions for the inventory control problem. Base-stock policies offer rapid execution and can yield near-optimal solutions in relatively simple settings with limited uncertainty [60]. More sophisticated heuristics that

TABLE II
EXAMPLES OF RL APPROACHES APPLIED TO SCHEDULING PROBLEMS WITH THEIR RESPECTIVE MDP CHARACTERISTICS.

Publication	RL Algorithm	Action Space	Observation Space	Reward
[47]	P3O	Lot-machine assignment	Graph based schedule	Makespan
[48]	DQN	Lot-machine assignment	Graph based schedule	Makespan
[49]	DQN	Lot sequence	Lot features	Utilization
[56]	CMA-ES	Lot sequence	Lot features	Tardiness/Throughput
[53]	PPO	Lot-machine assignment	Lot-machine features	Makespan
[54]	DDPG	Dispatching rules	Lot-machine features	Makespan
[50]	DQN	Dispatching rules	Production status	Tardiness

have been applied to supply chain simulation optimization include genetic algorithms and meta-heuristics [61]. Recently deep reinforcement learning (DRL) has shown considerable promise in inventory control applications [62]. Whilst a wide array of DRL algorithms is available, proximal policy optimization (PPO) is widely considered to be easy to tune for such problems [63], though it may not be able to outperform heuristics in all circumstances [64]. Moreover, to prevent the state-action space from becoming prohibitively large in large-scale problems, PPO often requires customization of the method. An example is to reduce the action space by using knowledge of optimal solutions for small problem instances, and applying this knowledge to larger problem instances [65], [66]. In response to limitations of PPO, DRL algorithms tailored specifically to the challenges of inventory control have been proposed the difficulties of inventory control have recently been proposed [8], [67], which have been shown to support scalability when paired with algorithmic enhancements based on problem-specific insights [68]. DRL is able to learn adaptive, situation dependent strategies, in contrast to traditional models that often rely on static policies. The ability to learn adaptive strategies is especially useful in scenarios with non-stationary demand and/or complex supply chain dynamics [69], [70], [71].

3) *Implementation*: The implemented model addresses the capacitated production inventory control problem, striving to deliver solutions rapidly and with near-optimal quality. Consequently, a DRL approach utilizing the PPO algorithm is developed as the policy generator within the supply chain simulation framework. This DRL model receives input in the form of state information for each item, including available production capacities, current inventory levels, and forecasted demand for multiple upcoming periods. Based on this input, the model determines production decisions—specifically, the quantity of each product to produce and the resources or supply chains to employ. To enhance computational efficiency and manage complexity, both the action and state spaces are discretized. This discretization reduces the complexity and size of the problem, facilitating faster convergence of the DRL model toward an effective policy. Additionally, action masking is applied to exclude infeasible production decisions due to capacity constraints. Action masking identifies infeasible actions, setting their selection probability to zero, thereby preventing the DRL agent from exploring irrelevant or impossible state-

action combinations. This targeted exploration significantly accelerates the training and convergence process [66]. To assess the effectiveness of the implemented DRL model, it is bench-marked against a (S,Q) base-stock policy in a realistic multi-objective setting. This policy orders items in each period, until the inventory position is above the safety stock level S, it can only order in multiples of Q items. In situations with ample capacity, this policy will mimic the behavior of LP models closely.

4) *Results*: To compare the two approaches, fifty agents were trained using the same PPO algorithm and subsequently evaluated against a baseline base-stock policy. The DRL models consistently achieved higher cumulative rewards relative to the base-stock counterparts. In addition, the variance in overall performance was lower for the DRL models, indicating more stable aggregate outcomes. However, variance across individual objectives was notably higher, suggesting that DRL agents converged to multiple distinct strategies, each capable of attaining similar near-optimal rewards. This phenomenon can be attributed to the inherent stochasticity of DRL training, which arises from factors such as distributional action sampling, variability in the training environment, and the complexities of non-convex optimization landscapes [72]. While the ability to discover diverse high-performing strategies enhances the model's exploratory capacity, it comes at the cost of reduced transparency. In particular, the absence of a clear, explainable policy may undermine user trust in the model's recommendations [73].

Another finding is that the DRL agents effectively leverage the future demand forecast embedded within the state-space to anticipate and proactively plan for substantial fluctuations in demand. In contrast, the benchmark model lacks this capability; as a result, when demand shifts abruptly, its response lags, leading to suboptimal planning and stock-outs.

5) *Scalability*: The current implementation has been applied to a small subset of real-world data; however, for the model to provide tangible business value, it must be scaled to life-sized datasets. To achieve this, several strategies can be employed. Transfer learning offers a promising solution by allowing a model, trained on a generalized smaller dataset, to be adapted for larger datasets, thus reducing training time without compromising performance [74]. Initial experiments applying transfer learning to this problem have produced encouraging results. Furthermore, graph-based approaches, such as the

integration of graph databases and graph neural networks, can enhance both generalization and scalability, outperforming traditional neural networks that rely on relational databases [75]. Another viable strategy involves the use of mixture of expert (MoE) models. By activating only subsets of neurons relevant to specific product groups, MoE models enable a single DRL framework to efficiently manage extensive and diverse product portfolios. This approach avoids the added complexity of multi-agent systems while maintaining high levels of performance [76].

C. Reel Utilization in Production by 3D Printing

1) *Problem Description:* In industrial additive manufacturing (AM), production relies on filament reels as the primary consumable. Each job depletes a portion of a reel, with varying consumption amounts depending on component complexity and size. Efficient utilization of these reels is essential to minimize material waste, avoid mid-print interruptions, and reduce downtime caused by frequent reel replacement.

The underlying optimization challenge mirrors a dynamic variant of the Cutting Stock Problem (CSP), where each incoming component request must be assigned to one of the available reels, which may have partial remaining capacity. However, unlike classical CSPs, this is an online decision-making problem as component requests arrive sequentially in real time, and each assignment must be made in real time without knowledge of future demands. Furthermore, the objective function explicitly incorporates the cost associated with residual filament that remains unusable for subsequent print jobs due to insufficient remaining capacity.

2) *Heuristics:* Conventional industrial practice employs simple greedy heuristics, such as First Fit or Best Fit Decreasing, to assign jobs to reels. A commonly used rule prioritizes assigning a job to the smallest available reel that can accommodate it, thereby conserving larger reels for future tasks. While computationally efficient and interpretable, these heuristics disregard long-term consequences, such as the accumulation of unusable leftovers and the stochastic nature of job sequences. This often results in suboptimal resource utilization and increased material waste.

Offline strategies consider a list of component requests in advance and solve the problem with integer programming techniques. However, these methods are inapplicable in real-time production environments characterized by high product variability, dynamic demand, and uncertainty, where immediate, adaptive decision-making is required.

3) *Reinforcement Learning Approach:* To address these limitations, we model the reel assignment problem as a Markov Decision Process (MDP). The state space comprises the current filament levels of all active reels, together with the filament requirement of the incoming component request. The action space consists of assigning the component request to one of the available reels. The goal is to minimize average filament waste while ensuring that all jobs are assigned and completed.

We introduce an index-based heuristic policy, assigning to each possible reel capacity a computed priority index that

reflects its future cost given the current job and reel state. We employ Deep Controlled Learning [8], an approximate policy iteration framework, where the index policy is refined through repeated evaluation and improvement steps using Deep Reinforcement Learning techniques. This allows us to maintain a low-dimensional, interpretable control policy while incorporating long-term reward optimization.

4) *Results:* We assess the performance of our proposed approach through a comparative analysis involving three policies: a baseline random policy, the index policy derived from structural problem analysis, and a DRL-enhanced index policy obtained via approximate policy iteration. The random policy, which assigns jobs to reels uniformly at random, serves as a naive benchmark. The index policy introduces a rule-based reel selection that leverages the filament requirement distribution of the component requests to guide decisions. Finally, the DRL policy learns to adaptively refine this structure by estimating long-term value under uncertainty.

Experimental and theoretical results demonstrate that the index policy significantly outperforms the random baseline in terms of waste reduction. Furthermore, the DRL-enhanced policy consistently outperforms the static index policy.

5) *Research Implications:* This case study highlights the potential of integrating structural modeling with reinforcement learning to address dynamic, real-time scheduling challenges in resource-constrained manufacturing systems. The proposed index-based formulation, combined with DRL-driven policy refinement, offers a promising balance between interpretability, scalability, and performance. It demonstrates how reinforcement learning can be effectively tailored to discrete, resource-constrained environments by embedding problem structure into the policy design.

From a broader perspective, this work aligns with the objectives of Industry 5.0 by enabling intelligent and sustainable decision-making practices. In particular, it illustrates how adaptive control strategies can reduce material waste, enhance resource utilization, and support more flexible, demand-driven production systems in additive manufacturing.

D. Long-Term Rapid Scenario Planning in the Semiconductor Industry using DRL

1) *Problem Description:* In the semiconductor industry, effective supply chain planning is crucial given the complexity and global nature of manufacturing processes. Traditional planning methods often rely on Mathematical Programming (MP) models that provide optimal solutions but suffer from high computational complexity. At NXP Semiconductors N.V., the challenge is compounded when rapid adjustments are required due to unforeseen events, where planners manually update production plans generated by MP models. This manual approach frequently leads to suboptimal resource utilization and inefficiencies. Thus, there is a critical need for automated, near-optimal solutions that can swiftly respond to dynamic changes in supply chain conditions.

2) *Current Approaches and Limitations:* Presently, NXP utilizes MP models primarily for weekly planning. However, daily updates required by unexpected occurrences cannot be

efficiently managed by these models due to computational delays. Dispatching Prioritization Rules (DPRs), both generic and tailored, offer faster solutions but lack the capability for long-term, strategic decision-making, often leading to inefficiencies and underutilized resources. Consequently, rapid scenario planning at NXP requires an agile, scalable solution capable of handling dynamic conditions effectively.

3) **Deep Reinforcement Learning Approach:** To overcome these limitations, a Deep Reinforcement Learning (DRL) framework is proposed [55], leveraging its strength in handling large-scale, sequential decision-making problems. The approach is formalized as a Markov Decision Process (MDP), with clearly defined states, actions, and rewards:

- **State Space:** Incorporates detailed information regarding remaining capacities for each week and supply path, as well as unplanned orders categorized by priority and due date.
- **Action Space:** Defines actions as allocations of orders to specific supply paths and production weeks, considering capacities and Just-In-Time (JIT) scheduling constraints.
- **Reward Function:** Immediate feedback is provided based on lateness of scheduled orders and final cumulative shortages, weighted by customer priority.

Two key DRL algorithms were explored:

- 1) **Proximal Policy Optimization (PPO):** Balances exploration and exploitation with clipped policy updates.
- 2) **Deep Controlled Learning (DCL):** An approximate policy iteration algorithm [8] transforming RL into a supervised learning classification problem.

4) **Results:** The DRL framework, particularly the DCL algorithm, demonstrated significant advantages. In scenarios with varying degrees of stochasticity, DCL consistently matched or slightly outperformed tailored DPRs while drastically reducing runtime compared to MP models. For instance, with low randomness, DCL showed objective values comparable to tailored DPRs, while PPO's performance lagged behind. Under higher stochastic conditions, DCL maintained superior performance, reinforcing its robustness and adaptability to dynamic order distributions.

5) **Implications and Scalability:** The DRL-based solution proves especially effective for rapid, scenario-driven adjustments in supply chain planning. Its ability to adapt to stochastic environments ensures sustainable and efficient resource allocation. Although DRL requires substantial initial training, its subsequent operational deployment significantly reduces decision-making latency.

Future enhancements to this DRL approach could involve:

- Shortening planning horizons for improved learning efficiency.
- Integrating real-time order arrivals for dynamic responsiveness.
- Expanding supply path options to address the dimensionality challenge while retaining flexibility.

Overall, leveraging DRL in semiconductor supply chain planning presents a promising direction, effectively balancing long-term strategic objectives with the agility required for rapid decision-making.

E. Differentiable physics for robot motion planning

1) **Description:** This chapter introduces the application of differentiable physics (DP) for trajectory optimization in robotic (UR5) manipulation and provides a systematic comparison with reinforcement learning (RL) method Proximal Policy Optimization (PPO), a widely recognized baseline in robotic learning tasks. Using DP, the robotic arm adaptively learns to reach the goal state and optimizes control parameters through gradient-based learning. The optimization process is governed by a loss function measures the deviation between the current state and the target state, and gradients are propagated through the simulator to update the control variables.

The proposed algorithm is integrated into a broader robotic system that includes: a code synthesis tool for robot control, a robot control system that integrates machine vision, robot motion planning and execution, and 6D object pose estimation [25]. The algorithm is evaluated via comparison with Proximal Policy Optimization (PPO), a method widely recognized as one of the most suitable approaches for robotic learning tasks. DP-based training is expected to reduce the sim-to-real gap, produce smoother trajectories, and improve placement accuracy. Incorporating DP into the training process further facilitates rapid adaptation to new tasks, thereby enhancing the robustness and flexibility of motion planning in robotic manipulators.

2) **Task Setup and Simulation:** To assess the effectiveness of DP in improving task performance, two bottle-insertion scenarios are considered. The first scenario involves insertion into a stationary socket, while the second introduces a dynamic setting in which the socket moves along a conveyor belt and stops at unpredictable positions. This dual setup enables evaluation under both controlled and partially uncertain environmental conditions [24].

A UR5 robotic manipulator, pre-equipped with a bottle, initiates each task from a randomly sampled position above the conveyor system. This randomized initialization increases variability across trials and prevents overfitting to a fixed starting configuration. The interaction between the robotic arm and the environment is modeled using a PD simulator Nimble Physics (NP). At each control step, the current joint states of the robot, together with the position of the target socket, are provided as inputs to a feedforward neural network. The network outputs joint torques that drive the manipulator toward successful insertion. To promote smooth and adaptive control behavior, the network architecture incorporates sinusoidal input encoding, GELU activation functions, and instance normalization. These components contribute to stable gradient propagation and enable the generation of continuous and responsive control signals throughout task execution.

3) **Training:** The model is trained in the Nimble Physics simulator, allowing gradients to propagate over time using backpropagation through time. This allows the optimization process to account for long-horizon dependencies between control actions and the resulting system states. The objective function consists of multiple components designed to enforce task specific constraints. A primary term minimizes positional error between the bottle and the target socket, ensuring accurate alignment. An additional term maintains

upright orientation during insertion. A velocity-regularization component penalizes excessive end-effector speed, promoting controlled motion, while a smoothness term reduces abrupt variations in joint torques to generate continuous trajectories. The objective is defined by the composite loss function shown in the Fig. 2 :

$$\mathcal{L} = \mathcal{L}_{pos} + \lambda_o \mathcal{L}_{orien} + \lambda_c \mathcal{L}_{contact} + \lambda_v \mathcal{L}_{vel}.$$

Each term of the loss function enforces a task-specific constraint. The position term is defined as

$$\mathcal{L}_{pos} = \|P_{bottle} - P_{target}\|^2,$$

where P_{bottle} denotes the position of the bottle and P_{target} represents the position of the socket target. This term minimizes positioning error. The orientation term maintains the correct upright orientation:

$$\mathcal{L}_{orien} = \|R_{bottle} - R_{upright}\|^2,$$

where R_{bottle} is the current orientation and $R_{upright}$ denotes the desired upright configuration. To address contact interactions, rigid collision responses are replaced with differentiable penalty functions:

$$\mathcal{L}_{contact} = \phi(d),$$

where d denotes the signed distance between interacting bodies and ϕ is a smooth penalty function. This "soft-contact" formulation preserves gradient flow while guiding the bottle toward insertion from above without introducing non-differentiable discontinuities typical of rigid-body contacts. To promote stable motion, velocity regularization is introduced:

$$\mathcal{L}_{vel} = \|\mathbf{v}_e\|^2,$$

where $\mathbf{v}_e$ represents the end-effector velocity. To address contact interactions, rigid collision responses are replaced with differentiable penalty functions. This soft-contact formulation preserves gradient flow while guiding the bottle toward insertion from above without introducing non-differentiable discontinuities typical of rigid-body contacts.

Optimization is performed over 500,000 training iterations using the Adam optimizer, enabling stable convergence under gradient-based updates.

4) *Results and Analysis:* The robot is trained using the DP and RL approaches in two variant tasks: with a fixed socket and a moving socket. In both cases, the objective is to insert a bottle with high precision, speed, and stability (see in the Fig. 3)[24].

In the fixed socket scenario, the controller trained in DP achieved a success rate of 94%, compared to 92% for RL. The DP method also showed better accuracy, with an average positional error of 1.21 mm (vs 4.32 mm for RL) and an orientation deviation of 0.21° (vs 0.58° for RL). Additionally, the DP approach produced slightly shorter and smoother trajectories (0.35 m vs. 0.38 m). In the more challenging task of moving the socket, the DP controller maintained a success rate of 84%, which is close to the 87% achieved by RL. This is explained by the fact that RL training used a much larger number of steps (12 million) compared to the DP method

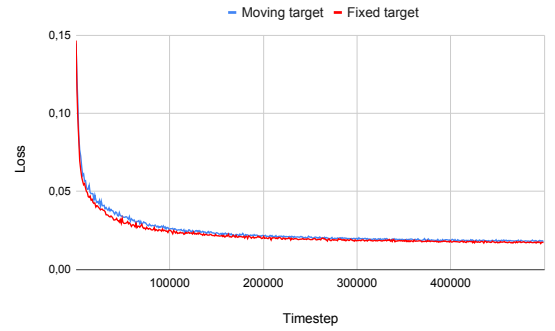


Fig. 2. DP training loss: red fixed socket scenario; blue moving socket scenario.

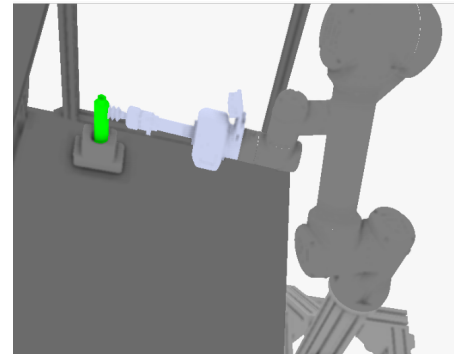


Fig. 3. Simulation of a UR5 robot arm that needs to insert a bottle into a designated socket.

(500K). The DP model was trained from scratch for both scenarios (500k iterations each). In contrast, the RL model was first trained on the fixed-socket task for 4 million steps and then fine-tuned for an additional 8 million steps on the moving-socket task.

However, the DP model again outperformed in accuracy, showing an average positional error of 1.51 mm (compared to 5.36 mm) and an orientation error of 0.23° (compared to 0.68°). DP trajectories were also more efficient, averaging 0.36 m compared to 0.43 m with RL.

These results highlight that while RL can achieve comparable success rates, DP consistently yields more precise, stable, and physically realistic trajectories, with lower deviation and faster per-action computation (0.3 ms), making it especially suitable for real-time deployment.

5) *Conclusions:* The DP training approach enables efficient learning of precise and physically consistent motion plans. Unlike RL, it requires less training time and provides more stable trajectories. This positions DP as a practical tool for industrial tasks involving contact-rich dynamic manipulation, with ongoing work focusing on integrating camera-based perception and extending the method to real-world hardware.

IV. IDENTIFICATION OF THE RESEARCH GAPS – FOR
INDUSTRY 4.0/5.0

Despite the remarkable progress in applying Deep Reinforcement Learning (DRL), Evolutionary Strategies (ES), and Differentiable Physics (DP) to industrial scheduling, control, and supply chain optimization, several key research gaps remain, particularly in the context of Industry 4.0 and 5.0:

1. Lack of Generalization Across Use Cases and Environments

Many DRL models and heuristic optimizations are highly tuned to specific problem instances or environments. This limits their transferability across different manufacturing setups, supply chain configurations, or robotic systems. There is a clear lack of reusable software, as well as techniques for making algorithms broadly applicable, in this domain.

2. Integration of Human-AI Collaboration

Industry 5.0 emphasizes human-centric design and human-AI collaboration. However, most current DRL or optimization-based methods operate as black-box solvers without mechanisms for interpretability, controllability, or integration with human decision-making. Research is needed into hybrid frameworks that allow operators to guide, intervene in, or understand the reasoning of the algorithm.

3. Scalability and Real-Time Constraints

As demonstrated in supply chain and scheduling use cases, scalability remains a limitation. While such limitations have been overcome in related domains [77], there remains a need for hardware-efficient learning frameworks and scalable DRL techniques.

V. CONCLUSION

This chapter has reviewed the recent advances in applying Deep Reinforcement Learning, Evolutionary Strategies, and Differentiable Physics to industrial scheduling, supply chain planning, and robotic control. We explored the foundational principles of these methods and showcased their application in high-impact use cases, from wafer lot scheduling in semiconductor manufacturing to supply chain master planning, 3D printing reel optimization, and robot motion planning.

These approaches promise more adaptive, scalable, and robust decision-making systems that align with the evolving goals of Industry 4.0 and 5.0—namely, increased automation, flexibility, and human-machine collaboration. However, transitioning from academic success to industrial impact demands further progress in several key areas: improving generalization, enhancing interpretability, reducing simulation dependence, and integrating uncertainty-aware learning.

The combination of DRL, ES, and DP offers a powerful toolkit. When aligned with domain expertise and industry requirements, these technologies can shift traditional optimization paradigms toward autonomous, self-adaptive, and intelligent industrial systems. As industrial AI evolves, continued cross-disciplinary collaboration between AI researchers, engineers, and domain experts will be crucial to closing remaining gaps and ensuring sustainable, human-centered innovation in manufacturing and logistics.

REFERENCES

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [2] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
- [3] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, pp. 229–256, 1992.
- [4] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in neural information processing systems*, vol. 12, 1999.
- [5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
- [6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
- [7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.
- [8] T. Temizöz, C. Imdahl, R. Dijkman, D. Lamghari-Idrissi, and W. van Jaarsveld, “Deep controlled learning for inventory control,” *European Journal of Operational Research*, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221725000463>
- [9] X. Chen, D. Diao, H. Chen, H. Yao, H. Piao, Z. Sun, Z. Yang, R. Goebel, B. Jiang, and Y. Chang, “The sufficiency of off-policy and soft clipping: Ppo is still insufficient according to an off-policy measure,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7078–7086.
- [10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.
- [12] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
- [13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Offpolicy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.
- [14] R. Newbury, J. Collins, K. He, J. Pan, I. Posner, D. Howard, and A. Cosgun, “A review of differentiable simulators,” *IEEE Access*, 2024.
- [15] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, “DiffTaichi: Differentiable programming for physical simulation,” *arXiv preprint arXiv:1910.00935*, 2019.
- [16] J. Heeg, Y. Song, and D. Scaramuzza, “Learning quadrotor control from visual features using differentiable simulation,” *arXiv preprint arXiv:2410.15979*, 2024.
- [17] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, “Differentiable physics models for real-world offline model-based reinforcement learning,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4163–4170.
- [18] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, “End-to-end differentiable physics for learning and control,” *Advances in neural information processing systems*, vol. 31, 2018.
- [19] K. Freivalds, L. Leja, and O. Teikmanis, “Learning to move objects with fluid streams in a differentiable simulation,” in *2024 7th Iberian Robotics Conference (ROBOT)*. IEEE, 2024, pp. 1–7.
- [20] O. Teikmanis, L. Leja, and K. Freivalds, “Applying a differentiable physics simulation to move objects with fluid streams,” in *International Workshop on Embedded Digital Intelligence (IWoEDI)*, 2023, p. 2.
- [21] L. Leja, O. Teikmanis, and K. Freivalds, “Shaping flames with differentiable physics simulations,” *Machine Learning and the Physical Sciences* at NeurIPS, vol. 202, pp. 1–6, 2024.
- [22] L. Leja and K. Freivalds, “Controlling shape of flame with differentiable physics,” *Engineering for Rural Development*, vol. 24, pp. 666–672.

- [23] L. Leja and U. Strautins, "Network model for thermal conversion of heterogeneous biomass granules," *Engineering for Rural Development*, pp. 737–743, 2021.
- [24] L. Leja, G. V. Strazds, C. Chronis, I. Varlamis, and K. Freivalds, "Differentiable physics training method for robot motion planning," *IFAC-PapersOnLine*, vol. 59, no. 27, pp. 220–225, 2025.
- [25] G. V. Strazds, K. Vitrake, G. Hollósi, L. Leja, A. Prieditis, D. Dupl, evska, M. Meiers, C. Chronis, I. Varlamis, G. Dimitrakopoulos et al., "Applied ai for adaptive robotic conveyor-feeding," *IFAC PapersOnLine*, vol. 59, no. 27, pp. 214–219, 2025.
- [26] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax-a differentiable physics engine for large scale rigid body simulation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.
- [27] T. A. Howell, S. Le Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable simulator for robotics," *arXiv preprint arXiv:2203.00806*, vol. 9, no. 2, p. 4, 2022.
- [28] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints," in *Robotics: Science and Systems*, 2021.
- [29] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand, "Taichi: a language for high-performance computation on spatially sparse data structures," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–16, 2019.
- [30] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," <https://github.com/nvidia/warp>, 2022, nVIDIA GPU Technology Conference (GTC).
- [31] J. Collins, R. Brown, J. Leitner, and D. Howard, "Follow the gradient: Crossing the reality gap using differentiable physics (realitygrad)," *arXiv preprint arXiv:2109.04674*, 2021.
- [32] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.
- [33] S. Le Cleac'h, M. Schwager, Z. Manchester, V. Sindhwani, P. Florence, and S. Singh, "Single-level differentiable contact simulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4012–4019, 2023.
- [34] I. Georgiev, K. Srinivasan, J. Xu, E. Heiden, and A. Garg, "Adaptive horizon actor-critic for policy learning in contact-rich differentiable simulation," *arXiv preprint arXiv:2405.17784*, 2024.
- [35] G. Tembey, A. Dahik, C. Richard, and V. Rastogi, "Navigating the costly economics of chip making," September 2023, [Online; accessed on 21-October-2024]. [Online]. Available: <https://www.bcg.com/publications/2023/navigating-the-semiconductor-manufacturing-costs>
- [36] L. Mönch, J. W. Fowler, and S. J. Mason, *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems*, ser. *Operations Research/Computer Science Interfaces Series*. New York: Springer, 2013, vol. 52.
- [37] M. Umlauf, M. Gojkovic, K. Harshina, K. Majbour, and M. Schranz, "Bees, bats and glowworms: Swarm algorithms for optimizing industrial plants from the bottom-up," in *Agents and Artificial Intelligence - 15th International Conference, ICAART 2023, Lisbon, Portugal, February 22-24, 2023, Revised Selected Papers*, ser. *Lecture Notes in Computer Science*, A. P. Rocha, L. Steels, and H. J. van den Herik, Eds., vol. 14546. Springer, 2023, pp. 3–25.
- [38] R. Ali, S. Qaiser, M. El-Kholany, P. Eftekhari, M. Gebser, S. Leitner, and G. Friedrich, "A greedy search based ant colony optimization algorithm for large-scale semiconductor production," in *Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SCITEPRESS - Science and Technology Publications, 2024, p. 138–149. [Online]. Available: <http://dx.doi.org/10.5220/0012813100003758>
- [39] B. Kovács, P. Tassel, and M. Gebser, "Optimizing dispatching strategies for semiconductor manufacturing facilities with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. *GECCO '23*. New York, NY, USA: Association for Computing Machinery, 2023, p. 1374–1382.
- [40] G. Da Col and E. Teppan, "Google vs ibm: A constraint solving challenge on the job-shop scheduling problem," *Electronic Proceedings in Theoretical Computer Science*, vol. 306, p. 259–265, Sep. 2019.
- [41] G. Da Col and E. C. Teppan, "Industrial-size job shop scheduling with constraint programming," *Operations Research Perspectives*, vol. 9, p. 100 249, 2022. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2214716022000215>
- [42] A. M. Ham and E. Cakici, "Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches," *Comput. Ind. Eng.*, vol. 102, pp. 160–165, 2016.
- [43] A. Ham and J. Fowler, "Constraint programming approach for scheduling batch operations with incompatible job families in fab and compatible job families in backend: Fa: Factory automation," in *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 2017, pp. 294–297.
- [44] A. Klemmt, J. Kutschke, and C. Schubert, "From dispatching to scheduling: Challenges in integrating a generic optimization platform into semiconductor shop floor execution," in *2017 Winter Simulation Conference, WSC 2017, Las Vegas, NV, USA, December 3-6, 2017*. IEEE, 2017, pp. 3691–3702.
- [45] R. Ali, M. M. El-Kholany, and M. Gebser, "Flexible job-shop scheduling for semiconductor manufacturing with hybrid answer set programming (application paper)," in *International Symposium on Practical Aspects of Declarative Languages*. Springer, 2023, pp. 85–95.
- [46] D. L. Applegate and W. J. Cook, "A computational study of the job-shop scheduling problem," *INFORMS Journal on Computing*, vol. 3, no. 2, pp. 149–156, 1991.
- [47] L. Hua, H. Liu, and Y. Pan, "Solving industrial chain job scheduling problems through a deep reinforcement learning method with decay strategy," *Information Sciences*, vol. 702, p. 121906, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025525000386>
- [48] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multiagent reinforcement learning," *IEEE Access*, vol. 7, pp. 39 974–39 982, 2019.
- [49] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek, "Optimization of global production scheduling with deep reinforcement learning," *Procedia CIRP*, vol. 72, pp. 1264–1269, 2018, 51st CIRP Conference on Manufacturing Systems. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S221282711830372X>
- [50] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Applied Soft Computing*, vol. 91, p. 106208, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1568494620301484>
- [51] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and C. Xu, "Learning to dispatch for job shop scheduling via deep reinforcement learning," 2020. [Online]. Available: <https://arxiv.org/abs/2010.12367>
- [52] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible job-shop scheduling via graph neural network and deep reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, p. 1600–1610, 2023, cited by: 143; All Open Access, Green Open Access. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85134304558&doi=10.1109%2FTHI.2022.3189725&partnerID=40&m d5=ea3b88863f1c3e57663a04ad2bab1c33>
- [53] L. Wang, X. Hu, Y. Wang, S. Xu, S. Ma, K. Yang, Z. Liu, and W. Wang, "Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning," *Computer Networks*, vol. 190, p. 107969, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128621001031>
- [54] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-critic deep reinforcement learning for solving job shop scheduling problems," *IEEE Access*, vol. 8, pp. 71 752–71 762, 2020.
- [55] B. de Jong, K. Schelthoff, R. L. Bianco, and W. van Jaarsveld, "Longterm rapid scenario planning in the semiconductor industry using deep reinforcement learning," in *2024 Winter Simulation Conference (WSC)*. IEEE, 2024, pp. 1818–1829.
- [56] P. Stöckermann, A. Immordino, T. Altenmüller, G. Seidel, M. Gebser, P. Tassel, C. W. Chan, and F. Zhang, "Dispatching in real frontend fabs with industrial grade discrete-event simulations by deep reinforcement learning with evolution strategies," 2023, p. 3047–3058.
- [57] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding atari agents," 2018. [Online]. Available: <https://arxiv.org/abs/1711.00138>
- [58] A. Kuhnle, J. Kaiser, F. Theiß, N. Stricker, and G. Lanza, "Designing an adaptive production control system using reinforcement learning," *Journal of Intelligent Manufacturing*, vol. 32, no. 3, pp. 855–876, 2021.
- [59] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," 2019.
- [60] F. Chen, Y. Lu, and M. Xu, "Optimal inventory control policy for periodic-review inventory systems with inventory-level-dependent demand," *Naval Research Logistics*, vol. 59, pp. 430–440, 09 2012.

- [61] H. Jalali and I. V. Nieuwenhuyse, "Simulation optimization in inventory replenishment: a classification," *IIE Transactions*, vol. 47, no. 11, pp. 1217–1235, 2015.
- [62] R. N. Boute, J. Gijsbrechts, W. van Jaarsveld, and N. Vanvuchelen, "Deep reinforcement learning for inventory control: A roadmap," *European Journal of Operational Research*, vol. 298, no. 2, pp. 401–412, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221721006111>
- [63] F. S. Francesco Stranieri and C. Kouki, "Performance of deep reinforcement learning algorithms in two-echelon inventory control systems," *International Journal of Production Research*, vol. 62, no. 17, pp. 6211–6226, 2024. [Online]. Available: <https://doi.org/10.1080/00207543.2024.2311180>
- [64] I. Kaynov, M. van Knippenberg, V. Menkovski, A. van Breemen, and W. van Jaarsveld, "Deep reinforcement learning for one-warehouse multi-retailer inventory management," *International Journal of Production Economics*, vol. 267, p. 109 088, 2024.
- [65] L. v. Hezewijk, N. Dellaert, T. v. Woensel, and N. Gademann, "Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem," *International Journal of Production Research*, vol. 61, no. 6, pp. 1955–1978, 2023. [Online]. Available: <https://doi.org/10.1080/00207543.2022.2056540>
- [66] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *The International FLAIRS Conference Proceedings*, vol. 35, May 2022. [Online]. Available: <http://dx.doi.org/10.32473/flairs.v35i.130584>
- [67] P. Harsha, A. Jagmohan, J. Kalagnanam, B. Quanz, and D. Singhvi, "Deep policy iteration with integer programming for inventory management," *Manufacturing & Service Operations Management*, 2025.
- [68] T. van Dijk, T. Fleuren, T. Temizoz, Y. Merzifonluoglu, M. Hendriks, and W. van Jaarsveld, "Inventory planning in capacitated high-tech assembly systems under non-stationary demand," *Available at SSRN 4843271*, 2024.
- [69] B. Rolf, I. Jackson, M. Müller, S. Lang, T. Reggelin, and D. Ivanov, "A review on reinforcement learning algorithms and applications in supply chain management," *International Journal of Production Research*, vol. 61, no. 20, pp. 7151–7179, 2023. [Online]. Available: <https://doi.org/10.1080/00207543.2022.2140221>
- [70] L. van Hezewijk, N. P. Dellaert, and W. L. van Jaarsveld, "Scalable deep reinforcement learning in the non-stationary capacitated lot sizing problem," *International Journal of Production Economics*, vol. 284, p. 109601, 2025.
- [71] T. Temizöz, C. Imdahl, R. Dijkman, D. Lamghari-Idrissi, and W. van Jaarsveld, "Zero-shot generalization in inventory management: Train, then estimate and decide," *arXiv preprint arXiv:2411.00515*, 2024.
- [72] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," 2015. [Online]. Available: <https://arxiv.org/abs/1412.0233>
- [73] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," 2019. [Online]. Available: <https://arxiv.org/abs/1811.10154>
- [74] F. Ezzahra Achamrah, "Leveraging transfer learning in deep reinforcement learning for solving combinatorial optimization problems under uncertainty," *IEEE Access*, vol. 12, pp. 181 477–181 497, 2024.
- [75] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," 2018. [Online]. Available: <https://arxiv.org/abs/1806.01261>
- [76] O. Sanseviero, L. Tunstall, P. Schmid, S. Mangrulkar, Y. Belkada, and P. Cuenca, "Mixture of Experts Explained," <https://huggingface.co/blog/moe>, December 2023, accessed 2025-04-22.
- [77] J. F. van der Haar, W. van Jaarsveld, R. J. Basten, and R. N. Boute, "Industrializing deep reinforcement learning for operational spare parts inventory management," *Available at SSRN 4999374*, 2024.