

PRODUCTION LINE CONTROL MODELING VIA MDP AND MULTI-AGENT DEEP REINFORCEMENT LEARNING

Rafael Gomes Paes de Lira

Instituição: UFPE - Universidade Federal de Pernambuco

Endereço: Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE, 50670-901

Email: rafael.gplira@ufpe.br

Lucas Matheus Nascimento

Instituição: UFPE - Universidade Federal de Pernambuco

Endereço: Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE, 50670-901

Email: lucas.matheusn@ufpe.br

Leandro Henrique Gomes da Silva

Instituição: UFPE - Universidade Federal de Pernambuco

Endereço: Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE, 50670-901

Email: leandro.hgsilva@ufpe.br

Dênis Leite

Instituição: UPE - Universidade de Pernambuco

Endereço: Av. Gov. Agamenon Magalhães - Santo Amaro, Recife - PE, 50100-010

Email: denisleite@poli.br

Márcio José das Chagas Moura

Instituição: UFPE - Universidade Federal de Pernambuco

Endereço: Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE, 50670-901

Email: marcio.cmoura@ufpe.br

ABSTRACT

This work presents a methodological proposal based on Multi-Agent Reinforcement Learning (MARL) for the distributed control of machine speeds in industrial production lines. The problem is formulated as a Markov Decision Process (MDP), in which local agents make decisions based on information from adjacent buffers and the machine itself, while respecting the natural delay in the impact of actions throughout the system. The approach aims to replace fixed operational rules with adaptive policies learned in simulated environments, aligning with Industry 4.0 trends. The relevance of the proposal is underscored by recent advancements in the literature, which highlight the potential of MARL to handle scenarios characterized by high complexity and variability.

KEYWORDS. Multi-Agent Reinforcement Learning, Simulation, Digital Twins.

Paper topics: Introduction, Methodology, Future Work, Conclusion and Acknowledgements.

1. Introduction

The growing need for flexibility, efficiency, and adaptability in industrial production systems has driven significant transformations in manufacturing environments. In a context marked by volatile demands, interdependent processes, and unexpected disruptive events—such as technical failures and abrupt variations in production rates—traditional methods based on fixed, sequential rules have proven increasingly inadequate to ensure agile and robust decision-making. In response, Artificial Intelligence (AI) techniques, particularly Reinforcement Learning (RL), have emerged as promising tools for improving real-time operational decisions, especially in autonomous and adaptive systems [Sutton & Barto, 2018].

In this scenario, Multi-Agent Reinforcement Learning (MARL) models have gained prominence due to their ability to coordinate decisions among multiple agents in dynamic and distributed environments, such as industrial production lines [Busoniu et al., 2008]. The decentralized nature of these systems, combined with their learning flexibility, allows them to capture local interactions between machines, fostering decisions that align more closely with the physical reality of the factory floor. When integrated with simulation environments or digital twins, these agents can learn from virtual experiences, reducing the need for human intervention and overcoming the limitations of purely heuristic approaches.

This work proposes a MARL-based methodology for the distributed control of machine speeds in a production line, formulated as a Markov Decision Process (MDP) [Puterman, 2014]. The agents act locally, using information from adjacent buffers and the machine's own status, while respecting the logic that the effects of an action propagate along the line with a delay. The focus lies exclusively on the methodological structure of the solution, without empirical validation or real-world testing, highlighting the steps of defining states, action spaces, decision policies, and reward functions.

Recent advancements in the literature demonstrate the increasing maturity of industrial MARL applications, particularly when combined with simulated environments. For example, Bollinga and Kubler (2024) propose a MARL framework for intelligent allocation of maintenance technicians, combining knowledge-based rules with multi-objective learning in scenarios of varying urgency. Bi et al. (2025) explore the capabilities of deep RL in digital twins to reconfigure production lines in response to failures and bottlenecks, while Tan et al. (2025) integrate stochastic failure models into preventive strategies optimized by MARL. These studies show that policies learned in simulation can outperform rigid empirical rules.

Other authors have expanded the boundaries of industrial MARL by incorporating techniques such as line balancing [Nikkerdar & ElMaraghy, 2025], transfer learning in volatile environments [Le & Zhang, 2024], dynamic decentralized scheduling [Yu et al., 2024], real-time control with PLCs [Zhao et al., 2023], and cooperative negotiation among agents [Karakostas & Vlahavas, 2024]. In industrial robotics, Park and Oh (2025) and Gao and Song (2023) demonstrate how deep learning combined with simulation enables the development of autonomous agents for tasks like palletizing and inspection. These studies collectively underscore the central role of simulation as a safe and effective environment for training and testing adaptive policies.

Against this backdrop, this work positions itself as a relevant methodological contribution by proposing a comprehensive MARL framework tailored to production lines, with a focus on local and coordinated decision-making among machines. By accounting for the delayed and limited impact of actions across the production line, the model respects the natural dynamics of the system and offers a solution compatible with future real-world industrial implementations. Thus, even without experimental results, the proposal advances MARL as a practical, scalable tool aligned with the operational challenges of Industry 4.0.

2. Methodology

2.1. General Approach Description

This study's methodological proposal is based on Multi-Agent Deep Reinforcement Learning (MARL) with Double Deep Q-Networks (Double DQN) to optimize speed control in a

production line for cans. The line comprises 27 machines connected by conveyor belts, which also serve as intermediate buffers. Currently, the line operates using empirical rules designed by experienced operators. While effective in routine scenarios, these rules show limitations in atypical operational conditions, such as demand fluctuations or internal blockages.

The proposed approach employs a multi-agent reinforcement learning framework, where each machine is controlled by an autonomous agent that learns to adjust its speed to maximize the line's global performance. The agents interact with a validated simulated environment, aiming to develop adaptive, robust control policies capable of generalizing to conditions not covered by traditional empirical rules.

2.2. Simulation Environment

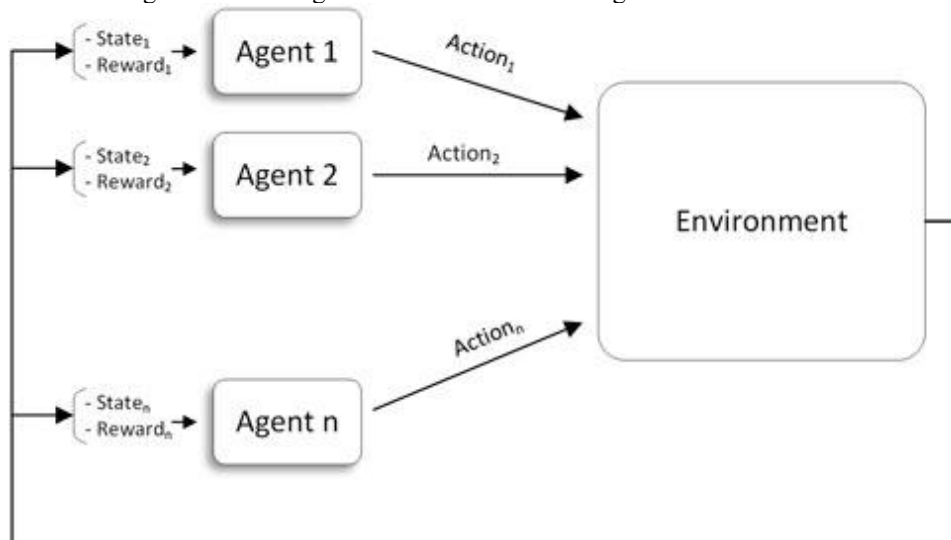
The RL Environment was built using Siemens Plant Simulation, a tool the company has used for over seven years to model and analyze its production systems. The model replicates the real production line's behavior with high fidelity and is considered validated for internal use. It includes 27 machines arranged in series, connected by conveyor belts that serve as both transport mechanisms and limited-capacity intermediate buffers.

The simulation incorporates stochastic process-time variability, minor stoppages, material accumulation, and demand changes, replicating the key operational challenges faced by the plant. The current empirical rules were programmed into the model as a benchmark for comparison with the MARL-based approach.

2.3. Problem Formulation as a Markov Decision Process (MDP)

The dynamic control of the production line is modeled as a multi-agent Markov Decision Process (MDP), where each of the 27 machines is represented by an autonomous agent making decentralized decisions. The MDP framework captures the system's sequential and stochastic nature, where the future state depends only on the current state and action, adhering to the Markov property (Figure 1).

Figure 1: Multi-Agent Reinforcement Learning Modeled as an MDP



Source: Bahrpeyma & Reichelt (2022)

2.3.1. Agents and Decentralized Architecture

Each agent represents a specific machine and operates with information limited to its immediate surroundings: the machine itself, the preceding and succeeding machines (where applicable), and adjacent buffers. This design reflects two key factors:

- Operational feasibility: Industrial infrastructure often lacks real-time centralized data collection for all machines due to hardware limitations, network latency, or robustness constraints.
- Temporal impact: The effects of a speed change in one machine take time to propagate to distant machines. Focusing on local contexts enables faster responses to failures, bottlenecks, or sudden production flow changes.

2.3.2. State Space

Each agent's observed state at time t includes:

- Controlled machine variables: Current speed, operational status (active, stopped, or failed), and time since last maintenance.
- Adjacent machine variables: Speed and status of the preceding machine (except for the first agent) and the succeeding machine (except for the last agent).
- Buffer conditions: Occupancy percentage of input and output buffers.

This structure allows agents to detect critical local patterns, such as imminent blockages (full output buffer) or idleness due to lack of supply (empty input buffer), optimizing their actions accordingly.

2.3.3. Action Space

Each agent selects from discrete speed ranges: 30%, 50%, 80%, or 100% of the machine's nominal capacity. This discretization serves two purposes:

1. Ensures practical feasibility by respecting physical/operational limits and avoiding abrupt commands.
2. Simplifies learning by reducing action-space complexity.

2.3.4. Reward Function

The reward function balances productivity and operational stability. Each agent's reward r_t is defined as:

$$r_t = \alpha * Throughput_t - \beta * IdleTime_t - \gamma * Blocking_t \quad (1)$$

Where:

- Throughput: Units processed per time unit (encourages productivity).
- IdleTime: Unplanned idle time (penalizes underutilization).
- Blocking: Time the machine was blocked due to a full downstream buffer (penalizes congestion).

Coefficients α , β , and γ are calibrated based on operational priorities. To incentivize collaboration, a differential reward strategy is used: an agent reducing its speed to prevent a downstream blockage may be indirectly rewarded if the action resolves the bottleneck.

2.3.5. Training Strategy

Agent policies are learned using Double Deep Q-Network (DDQN), which enhances Q-value update stability. Key mechanisms include:

- Experience Replay: Transitions (state, action, reward, next state) are stored in a buffer for random sampling, breaking temporal correlations and improving generalization.
- Target Network: A secondary neural network, updated less frequently, estimates target Q-values to reduce instability risks.

Training occurs in a Siemens Plant Simulation environment replicating real production line dynamics (e.g., failures, demand variations). After this phase, policies are transferred to the real environment for controlled validation and fine-tuning.

2.4. Reinforcement Learning Architecture

The Double Deep Q-Network (Double DQN) was selected as the algorithmic foundation for this work due to its ability to mitigate Q-value overestimation—a common issue in traditional Deep Q-Learning methods [van Hasselt et al., 2016]. In industrial settings, where incorrect decisions may incur high costs, this stability is critical. Double DQN achieves this balance by employing two separate neural networks:

- A main network (Q_θ) to select actions.
- A target network ($Q_{\theta'}$) to evaluate them, updated periodically from the main network.

This approach is particularly suited for high-variability environments like production lines subject to unplanned stoppages and demand fluctuations [Zhao et al., 2023]. The neural network was designed to process a moderate-dimensional state space (12 variables per agent, including speeds, operational statuses, and buffer levels). The architecture consists of:

- Input layer: 12 neurons (one per state variable), with Min-Max normalization for uniform scaling.
- Three hidden layers: 128, 64, and 32 neurons, respectively, with ReLU activation, chosen for computational efficiency and performance in industrial control tasks [Le & Zhang, 2024].
- Output layer: 4 neurons (one per speed range), with linear activation for Q-value estimation.

Weight updates follow the equation:

$$y_j = r_j + \gamma * Q_{\theta'}(s'_j, \text{argmax}_{a} Q_{\theta}(s'_j, a)) \quad (2)$$

Where:

- y_j is the Q-target for transition j ,
- γ is the discount factor (0.95 in this work),
- Q_{θ} and $Q_{\theta'}$ are the main and target networks.

The loss function for training is Mean Squared Error (MSE) between y_j and the current prediction:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^N (y_j - Q_{\theta}(s_j, a_j))^2 \quad (3)$$

with N as the mini-batch size (32 samples). The target network is updated via soft weight copying:

$$\theta' \leftarrow \tau \theta + (1 - \tau) \theta' \quad (\tau = 0,01) \quad (4)$$

The ϵ -greedy exploration strategy is formalized as:

$$a_t = \begin{cases} \text{random action with probability } \epsilon_t \\ \text{argmax}_a Q_{\theta}(s_j, a), & \text{otherwise} \end{cases} \quad (5)$$

with ϵ_t decaying linearly from 0.9 to 0.1 during training.

The learning algorithm (Algorithm 1) combines Experience Replay and periodic target network updates. At each timestep t^* , the agent stores the transition (s_t, a_t, r_t, s_{t+1}) in a fixed-size buffer (10,000 samples) and samples random batches (mini-batches of 32 transitions) to train the

main network, reducing temporal correlations. The target network is updated every 100 steps to maintain stability.

Algorithm 1: Double DQN for Speed Control in Production Lines

- 1: Initialize networks Q_θ and $Q_{\theta'}$ with random weights.
- 2: Define experience replay buffer D with capacity N.
- 3: For each episode and each agent:
- 4: Observe initial state s .
- 5: For each timestep t :
- 6: Select action a via ϵ -greedy policy.
- 7: Execute a in the simulator, receive reward r and next state s' .
- 8: Store transition (s, a, r, s') in D.
- 9: Sample random mini-batch (s_j, a_j, r_j, s'_j) from D.
- 10: Compute target: $y_j = r_j + \gamma * Q_{\theta'}^*(s'_j, \operatorname{argmax}_{Q_\theta}(s'_j, a))$.
- 11: Update Q_θ via gradient descent on $(y_j - Q_\theta(s_j, a_j))^2$.
- 12: Every C steps, copy Q_θ to $Q_{\theta'}$.
- 13: set $s \leftarrow s'$.

The exploration strategy uses decaying ϵ -greedy, with initial $\epsilon = 0.9$ (high exploration) and linear decay to 0.1 over 10,000 episodes. This approach balances the discovery of new policies and the exploitation of acquired knowledge, adapting to the non-stationary nature of the multi-agent environment [Tan et al., 2025]. To prevent conflicts between agents, synchronization follows a hybrid model: agents make decentralized decisions but share a common experience replay buffer during training, enabling collaborative learning without centralization [Yu et al., 2024].

2.5. Multi-Agent Configuration

The adopted coordination strategy follows a decentralized implicit communication approach, where each agent makes autonomous decisions based solely on local information, reflecting typical constraints of industrial environments [Nikkerdar & ElMaraghy, 2025]. This design avoids computational bottlenecks of centralized solutions and enhances robustness to individual failures. Cooperation emerges indirectly through the shared reward function, which penalizes global undesirable states (e.g., cascade blockages), aligning individual objectives with collective performance [Karakostas & Vlahavas, 2024].

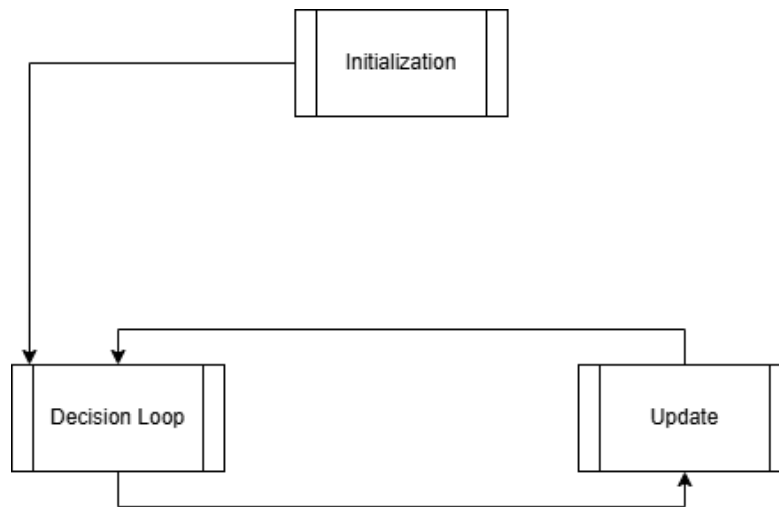
Agent communication is limited to observing adjacent machine states, with no explicit message exchange. This restriction simulates real-world conditions where factory communication systems often lack support for high-frequency data transfers between all machines. However, the use of a shared experience replay buffer during training allows agents to learn indirectly from others' interactions—a technique known as Centralized Training with Decentralized Execution (CTDE) [Park & Oh, 2025].

Training is conducted in 8-hour simulated episodes (equivalent to one production shift), totaling 10,000 episodes. Each episode resets agents to randomized states to ensure experiential diversity, including failure scenarios and variable demand. This approach, termed self-play with environment randomization, was adapted from Bi et al. (2025) to improve model generalization.

2.6 Integration with Simulated Environment

The interaction between learning and simulation follows a cyclic pipeline (Diagram 1):

Diagram 1: Python and Plant Simulation Integration



Source: Author

1. Initialization: The Siemens Plant Simulation software is launched with operational parameters based on historical factory data.
2. Decision Loop: Every 5 simulated minutes (equivalent to the minimum time required for safe speed adjustments in the real production line), agents receive the current simulator state via the Python library (PlantSim).
3. Update: Agent actions (speed settings) are applied in the simulator, which then advances time.

The 5-minute decision frequency was determined based on:

- Stabilization time: Interval needed for speed changes to produce measurable effects in adjacent buffers.
- Operational constraints: Physical limits of machines to prevent premature wear from frequent adjustments.

This integration enables model validation in extreme scenarios (e.g., simultaneous failures in multiple machines) before physical deployment, reducing operational risks [Gao & Song, 2023].

3. Future Work

This study represents an initial step toward implementing an intelligent, decentralized control system for industrial production lines using multi-agent deep reinforcement learning techniques. As a continuation, we propose migrating the simulated environment to a real-time control architecture grounded in Digital Twin principles.

The goal is to replace Plant Simulation interactions with direct connections to PLCs (Programmable Logic Controllers) of real production line machines. Each studied machine has a PLC that monitors its state (speed, failures, stoppages, etc.) and executes operational commands. The plant already has a continuous industrial data collection system in place, enabling this integration.

The proposed transition will allow:

- DDRL model agents to receive real-time machine data via PLC database integration.
- Dynamic speed decisions to be calculated and transmitted back to PLCs for adaptive line control.
- Autonomous system responses to typical factory events like unplanned stoppages, operational bottlenecks, and intermittent failures.
- Continuous effectiveness monitoring with real-time feedback for policy refinement and agent retraining.

This evolution will create an intelligence layer over the manufacturing process, enabling a highly responsive and resilient production line capable of adapting to adverse situations beyond conventional empirical rules. It also paves the way for a generalizable platform where multiple lines can be controlled by continuous learning systems driven by shop-floor data.

4. Conclusion

This work presented a decentralized deep reinforcement learning methodology for speed control in industrial production lines. The proposal integrates Double Deep Q-Learning algorithms into a multi-agent architecture coupled with a validated Siemens Plant Simulation model, widely used by our industry partner. Formulating the problem as a Markov Decision Process (MDP) enabled structured representation of the production line's dynamic behavior, accounting for operational variabilities like failures and blockages.

Using Siemens Plant Simulation as a training environment provides a robust foundation for validating agent behavior in representative stochastic scenarios. The MDP formulation, combined with Double Deep Q-Networks (DDQN), allows agents to learn efficient policies, improving throughput while reducing idle and blockage times.

Furthermore, this study advanced the Digital Twin concept by establishing a link between simulated systems and real shop-floor systems. The developed model may eventually operate in real-time, connected directly to machine PLCs, enabling intelligent automated responses to production environment variations and adversities.

As a natural continuation, we propose integrating real PLC data via IIoT (Industrial Internet of Things) infrastructure, allowing the machine learning model to directly control operational machine parameters in real-time. This advancement will represent a significant step toward intelligent automation and effective implementation of digital twins on the factory floor.

6. Acknowledgements

The authors gratefully acknowledge the financial support from the Human Resources Program of the Brazilian National Agency of Petroleum, Natural Gas and Biofuels (PRH-ANP) and the Graduate Program in Production Engineering (PPGEP), which were fundamental for the development and execution of this work.

Reference

- Bahrpeyma, F., & Reichelt, T. (2022). A survey on multi-agent deep reinforcement learning: From a practical perspective. *Frontiers in Robotics and AI*, 9, 1027340.
- Bi, X., Chen, Q., Zhou, Y. (2025). A digital twin-driven deep reinforcement learning approach for production line reconfiguration in smart factories. *Journal of Manufacturing Systems*, 67, 14–28.
- Bi, X. et al. (2025). Digital twin-assisted deep reinforcement learning for dynamic production line reconfiguration. *Journal of Manufacturing Systems*, 64, 123–135.
- Bollinga, S., Kubler, S. (2024). Rule-based multi-agent reinforcement learning for allocating technical staff in production systems. *Journal of Manufacturing Systems*, 70, 536–548.
- Busoniu, L., Babuska, R., & De Schutter, B. (2008). "A Comprehensive Survey of Multiagent Reinforcement Learning". *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 38(2), 156-172.
- Gao, J., Song, B. (2023). Simulation-based multi-agent reinforcement learning for autonomous robotic systems in manufacturing. *Journal of Manufacturing Systems*, 67, 315–330.

Karakostas, A., Vlahavas, I. (2024). Collaborative MARL for bottleneck mitigation in assembly lines. *Engineering Applications of AI*, 118, 106201.

Karakostas, A., Vlahavas, I. (2024). Multi-agent reinforcement learning for bottleneck avoidance in robotic assembly lines. *Journal of Manufacturing Systems*, 70, 85–99.

Le, T., Zhang, D. (2024). Recent advances in deep reinforcement learning for industrial applications: A review. *Journal of Manufacturing Systems*, 70, 629–647.

Nikkerdar, A., Elmaraghy, H. (2025). A multi-agent reinforcement learning approach for balancing intelligent production lines with predictive maintenance. *Journal of Manufacturing Systems*, 69, 283–295

Park, Y., Oh, S. (2025). Multi-agent deep reinforcement learning for coordinated industrial robot tasks. *Journal of Manufacturing Systems*, 68, 228–240.

Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Qiu, Y., Wang, H., Li, Y. (2024). Data-driven digital twin and multi-agent reinforcement learning for reconfigurable manufacturing. *Journal of Manufacturing Systems*, 68, 375–389.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Tan, Z., Liao, J., Hu, P. (2025). Multi-agent deep reinforcement learning-based dynamic maintenance strategy for discrete manufacturing systems. *Journal of Manufacturing Systems*, 69, 57–70.

Van Hasselt, H. et al. (2016). Deep reinforcement learning with double Q-learning. *AAAI*, 30(1).

Xu, X., Wang, R., Chen, J. (2023). Federated multi-agent reinforcement learning for collaborative scheduling in manufacturing. *Journal of Manufacturing Systems*, 67, 50–64.

Yu, P., Zhang, X., Li, Y. (2024). Multi-agent reinforcement learning-based real-time scheduling in a smart job shop environment. *Journal of Manufacturing Systems*, 68, 1–14.

Zhao, X., Wang, H., Zhang, L. (2023). Multi-agent reinforcement learning for real-time scheduling in flexible job-shop with dynamic arrivals. *Journal of Manufacturing Systems*, 67, 301–315.