HUNGARIAN JOURNAL OF
INDUSTRY AND CHEMISTRY
Vol. 54(SI) pp. 87–94 (2026)
hjic.mk.uni-pannon.hu
DOI: 10.33927/hjic-2026-21

HUNGARIAN JOURNAL OF INDUSTRY AND CHEMISTRY
HJIC

PRODUCTION CELL OPERATION OPTIMIZATION BY REINFORCEMENT LEARNING

BENCE PÁLVÖLGYI[1]*, ZSOLT JÁNOS VIHAROS[1,2], JENŐ CSANAKI[3], KRISZTIÁN MESKÓ[3] AND ZSOLT NAGY[3]

1 Research Laboratory on Engineering and Management Intelligence of the HUN-REN Institute for Computer Science and Control, Center of Excellence of the Hungarian Academy of Sciences (MTA), Budapest, 1111, HUNGARY
2 Faculty of Economics and Business of the John von Neumann University, Kecskemét, 6000, HUNGARY
3 Opel Szentgotthárd Autóipari Kft., Szentgotthárd, 9970, HUNGARY

Machine learning, particularly reinforcement learning, plays an increasing role in optimizing complex industrial processes. One such challenge arises in production systems, where products must be processed, often involving nontrivial scheduling and routing problems. The paper presents a reinforcement learning (RL)-based method to optimize a specific production cell, where two material-moving units and several machining units must cooperate to manufacture items that require both processing and occasional cleaning. The proposed methodology models the environment as a Markov Decision Process and employs RL algorithms to maximize throughput. Several popular RL algorithms were compared, and it was found that Maskable Proximal Policy Optimization (Maskable PPO) delivers the best performance, as agent-specific, valid and differentiated behavior is ensured for both material handling and machining units through action masking. Among the various masking strategies tested, a distinct masking approach proved to be the most effective.

Keywords: reinforcement learning, manufacturing operation optimization, action masking, production cell control

1. Introduction

Direct experimentation in real-world systems to optimize processes is often impractical due to the associated risks, costs, and potential disruptions. For this reason, appropriate simulation models are widely used, as they enable safe and accelerated testing of new strategies without interfering with ongoing operations. Simulation therefore provides a solid foundation for advanced optimization methods, including machine learning and reinforcement learning.

The potential of reinforcement learning (RL) across industrial domains has been widely recognized. Kegyes [1] emphasizes that RL offers promising solutions for a broad spectrum of fields, such as robotics, logistics, quality control, maintenance, and energy management. For example, recent work has shown that tabular Q-learning can be successfully applied to vehicle motion control, where an agent learns to stabilize steady-state drift maneuvers in a MATLAB/Simulink simulation environment [2].

The breadth of reinforcement learning applications in manufacturing has been systematically reviewed by Panzer and Bender [3]. Their study highlights that deep reinforcement learning (DRL) has been applied across a wide range of production disciplines, including scheduling and dispatching, process control, predictive maintenance, logistics, assembly, and energy management. The review also shows that policy-based and actor–critic algorithms, such as Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), Advantage Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) are among the most frequently and successfully applied methods, often outperforming heuristics and mathematical programming in simulated environments. At the same time, the authors identify significant gaps, such as the lack of standardized frameworks, the limited adoption of advanced techniques such as action masking, and the still-rare validation in real factory environments.

These observations are further reinforced by more recent review work, which emphasizes that real-world deployment of DRL in Industry 4.0 remains challenging

Received: 10 Nov 2025; Revised: 18 Dec 2025;
Accepted: 18 Dec 2025
*Correspondence: palvolgyi.bence.tamas@sztaki.hun-ren.hu

due to high-dimensional state and action spaces, multi-objective and poorly defined reward functions, stringent real-life constraints, and the persistent difficulty of transferring policies from simulation to physical systems [4]. The review also highlights the increasing reliance on digital twins, the importance of explainability and safety, and the need for robust benchmarking practices, underscoring that industrial DRL applications require both methodological rigor and domain-specific engineering considerations.

In addition to survey work, recent applied studies further demonstrate the usefulness of simulation-based reinforcement learning in industrial settings. Wei et al. [5] developed a deep reinforcement learning framework for optimizing complex production workflows, where tasks and their dependencies were represented as a directed acyclic graph (DAG). By integrating a Double Deep Q-Network with a simulation environment, their approach dynamically allocated resources to minimize makespan, reduce response time, and lower machine idle rates. The results showed consistent improvements over heuristic baselines such as First-Fit, Tabu Search, and Bayesian Optimization, particularly in large-scale or highly dynamic scenarios. This work illustrates how carefully designed simulation models and appropriately shaped rewards can effectively address intricate scheduling and resource-allocation challenges in manufacturing systems.

A recent study by Stappert et al. [6] provides a comprehensive analysis of action masking as a mechanism not only for preventing invalid actions but also for integrating human domain knowledge into reinforcement learning policies. Their work evaluates three distinct operations research problems—paint shop scheduling, peak-load energy management, and stochastic inventory control—and demonstrates that embedding heuristic or provably optimal actions into the action mask can significantly enhance both learning efficiency and final policy performance. The paper further shows that action masking is often essential in environments with constrained action spaces, where unguided agents fail to learn effective strategies. At the same time, their results caution against overly restrictive masks, as enforcing suboptimal heuristics can limit exploration and degrade performance. Overall, this study highlights how action masking can simultaneously improve trust, safety, and convergence speed while serving as a practical interface for integrating expert knowledge into RL systems deployed in real-world operational settings.

Laflamme et al. [7] investigated explainable reinforcement learning for automotive powertrain control. Their approach combined the Gymnasium RL and Stable-Baselines3 environments to train PPO agents and then distilled the resulting neural network policies into decision trees and ultimately lookup tables. This transformation largely preserved the performance of RL policies while providing the transparency and interpretability required for practical deployment in embedded automotive systems.

A related study demonstrated the effectiveness of DRL for optimizing industrial assembly lines by formulating the system as a Markov Decision Process and training agents in a simulated environment [8]. The authors showed that action masking greatly accelerated learning by eliminating infeasible choices and that a multi-agent architecture enabled scalability to large state and action spaces. Their results confirmed that DRL can achieve faster convergence and higher-quality schedules than traditional model-based approaches.

A similar combination of simulation and reinforcement learning was explored in recent work integrating a policy-based RL method into a digital twin of a semi-automated chaku-chaku production system [9]. By analyzing the effects of reward shaping and hyperparameter configuration, the authors derived an improved production control logic that increased productivity while maintaining stable worker task assignments. Their approach, validated on the digital twin of a real automotive assembly line, demonstrates how RL can complement digitalization efforts to optimize production control strategies in practice.

RL has also been explored for matrix-structured production systems, where conventional control strategies struggle with the dynamics of routing and dispatching. A combined RL approach tested in a simulated car-body construction environment achieved heuristic-level logistic performance while significantly reducing deadlocks in autonomous guided vehicle fleets [10]. This highlights the potential of RL for highly dynamic production concepts.

Multi-agent reinforcement learning has also been explored for system-level optimization in complex manufacturing environments. A recent study introduced an integrated MARL control framework combining system-process modeling with advanced actor–attention–critic architectures to improve overall system yields [11]. Their methods demonstrated fast convergence and robustness to environmental variations, highlighting the potential of MARL for coordinating multiple interacting components in large-scale manufacturing systems.

Reinforcement learning has also been applied to dynamic flexible job shop scheduling. A recent study developed a multi-agent RL framework for a robot assembly cell, where each robot learned decentralized scheduling decisions using a Double DQN-based approach [12]. Through centralized training and locally observed execution, the system outperformed rule-based heuristics in makespan optimization and demonstrated how agent-level observation design affects scheduling performance. This highlights the suitability of RL and MARL for highly dynamic and decentralized manufacturing environments.

A related line of work has demonstrated how reinforcement learning can support adaptive control in multi-layered and modular production systems. A hyper-heuristic control model was proposed in which distributed manufacturing and logistics agents used deep reinforcement learning to select situation-specific low-level heuristics within a semi-heterarchical production

system [13]. The approach, validated in simulation and transferred to a hybrid real-world environment, improved key multi-objective metrics such as throughput time, tardiness, and priority-order handling. This underscores the potential of RL to enhance adaptability and robustness in highly dynamic manufacturing contexts.

Reinforcement learning has also been applied to dynamic flexible job-shop scheduling, where traditional heuristics struggle with variability. A recent study formulated the scheduling problem as a heterogeneous multi-agent partially observable MDP and used a dueling double deep Q-network with reward shaping to coordinate job and machine agents [14]. The method outperformed state-of-the-art baselines in minimizing weighted tardiness and demonstrated strong adaptability in dynamic scenarios, underscoring the potential of RL for complex and variable scheduling environments.

Reinforcement learning has also been applied to statistical process control (SPC). Viharos and Jakab [15] demonstrated that reinforcement learning can be applied to maintain processes within control limits while minimizing costs. The method was validated in simulation, and it was also shown to be useful in industrial settings, demonstrating that RL can adaptively learn control policies that balance quality assurance with operational efficiency.

Reinforcement learning has also been applied to complex scheduling problems. Du and Li [16] proposed a deep reinforcement learning (DRL)-based evolutionary algorithm for distributed precast concrete production scheduling, a highly constrained and multi-objective industrial task. Their framework integrated multiple deep Q-networks to balance objectives such as minimizing tardiness, reducing earliness, and lowering electricity costs under time-of-use pricing. The study demonstrated that DRL could effectively handle distributed scheduling scenarios that are difficult to solve with traditional optimization techniques.

Clemente et al. [17] introduced an efficient framework for parallel deep RL, where multiple environment instances are executed synchronously on a single machine. Their Parallel Advantage Actor-Critic (PAAC) algorithm achieved state-of-the-art Atari performance within hours, demonstrating that parallelization can reduce training time by an order of magnitude while maintaining stability. This highlights the importance of parallelized experimentation for practical deployments, where training efficiency is often as important as final performance.

Ali and Tirel [18] applied action-masked reinforcement learning to industrial assembly line balancing, which is a classical NP-hard scheduling problem. By dynamically filtering infeasible actions at each step, their agents converged faster and achieved stable performance compared to unmasked baselines. The study illustrates that action masking not only improves sample efficiency but also ensures realism by embedding domain constraints directly into the training process.

Despite these advances, transferring RL from simulation to industrial practice remains challenging. Dulac-Arnold et al. [19] identified several obstacles for real-world RL, including high-dimensional state and action spaces, multi-objective and poorly defined reward functions, and the need to satisfy strict constraint requirements. These difficulties underline the necessity of designing environments and frameworks that can handle stochasticity and complexity while still being computationally efficient.

Having summarized RL applications in various industrial fields, the next section describes the specific industrial case and the related RL optimization framework. After detailing the benchmark task, Chapter 3 present comparisons and proposals concerning the applied RL algorithms, the optimal masking solution, the advantages of RL parallelization, and the benefits of reward shaping.

2. Experimental methodology

The industrial partner operates a factory organized into multiple production cells, each responsible for specific processing and assembly tasks. The overall objective is to improve production efficiency through advanced optimization methods. One production cell was selected as a case study. This cell comprises two material movement units and several production units responsible for both machining and periodic cleaning operations.

To build an accurate simulation environment for optimization purposes, extensive discussions were held with operators and engineers directly involved in daily operations. Real-world data on cycle times, machine usage, cleaning requirements, and workflow constraints were collected and analyzed. This collaborative process ensured that the developed simulation model closely reflects the actual manufacturing processes and challenges faced in the factory.

The simulation execution environment was developed in Python due to its extensive ecosystem for scientific computing, data analysis, and machine learning. Python's compatibility with reinforcement learning toolkits, in particular the Farama Foundation's Gymnasium environment creation tool, as well as Stable-Baselines3's compatibility with Gymnasium and its wide palette of reinforcement learning algorithms, enabled efficient development, experimentation, and benchmarking of different approaches.

2.1. RL environment creation

Gymnasium, maintained by the Farama Foundation, originally developed by OpenAI, provides a standardized framework for reinforcement learning research. It is widely used due to its large collection of benchmark environments (such as CartPole, MountainCar, and Atari, etc.), which serve as testbeds for algorithm development and comparison. In addition to these benchmarks, Gymnasium allows researchers to design and integrate custom environments tailored to specific application domains, such as the production cell investigated in this paper.

The Gymnasium library offers a rich set of tools to facilitate environment creation. A variety of observation/state and action space components are available, ranging from discrete and continuous spaces to more complex composite structures (e.g., dictionaries, tuples). This flexibility makes it possible to accurately represent the state, and decision spaces of industrial processes. The reported use case uses 45 different values for the observation space, both discrete and continuous, wrapped with dictionaries for better readability. These values describe the state of the product-placement units, the machining and testing stations, and the products themselves. The material-handling units are represented through their actual enumerated positions, the state of the product they currently carry, the most recent action they executed, and the time marking the completion of that action. Similarly, each machining or test station is described by its product state and the continuous time at which its current operation will finish. All position-related information follows an enumerated index system, product states are encoded using serial discrete values, and all timing information is expressed in continuous seconds.

The action space contains 21 different actions. These actions fall into movement, replacement, and waiting categories, each with its own structure. Movement actions relocate a unit between enumerated positions, require reachability and agent availability as preconditions, and have a deterministic duration proportional to path length, just as with real production used PLC controllers. Replacement actions cover picking, placing, and transferring products. Their preconditions depend on the content of both the agent and the station (or the two agents), and they are strictly enforced through action masking. Waiting actions require no preconditions and simply consume a defined duration. Three masking strategies are implemented: one that encodes existing production logic for benchmarking, another that applies only physical and environmental constraints, and a hybrid masking scheme that combines both, offering a balance between autonomy and optimization. Episodes terminate after a predefined number of learning steps, optionally allowing manual early stopping in line with common reinforcement learning practice. Despite actions having defined durations, the entire environment operates in continuous time, advancing the global clock to the next event completion, which enables a realistic simulation of asynchronous, event-driven manufacturing processes.

Moreover, Gymnasium includes wrappers, which are modular tools that can modify observations/states, rewards, or actions without changing the underlying environment. For instance, the observation wrapper enables state discretization or unpacking of state values, while other wrappers can normalize rewards or log additional training statistics. This feature is useful for some learning algorithms that cannot process layered data or continuous values as input.

In addition, Gymnasium makes it straightforward to define reward functions that depend on multiple factors, such as throughput, waiting times, or energy consumption. This allows the creation of hierarchical reward systems, where partial rewards are assigned to intermediate steps, and full rewards are given only when multiple correct actions occur in sequence. Such reward shaping is particularly suited for complex tasks such as production scheduling, where success requires a long chain of valid and coordinated decisions. The reward structure consists of five components that correspond to the essential steps required to manufacture and forward a finished product while avoiding unnecessary delays or counterproductive actions. These components include a high positive reward for produced products, reflecting the primary objective of maximizing throughput; a utilization-based reward that captures the effective use of machining units; a penalty for inefficient trajectories of the material-handling agents, discouraging unnecessary movements; a reward for successful product replacements, which represent critical interactions between handling units and workstations; and a deadlock penalty that signals situations where progress becomes impossible due to conflicting decisions. There is also one additional reward in the training system, namely a penalty for an unfeasible action choice; this is not used with the masked algorithms, as the basic environment mask forbids these decisions. Each of these reward components is computed based on its real-world contribution to efficient manufacturing operations. To balance their influence, the reward weights were determined through a manual grid-search procedure performed as part of the hyperparameter tuning, ensuring that the shaped reward meaningfully guides the learning process toward productive, stable, and interpretable behavior.

These reward components are calculated with the real-world benefit of manipulating or working on a product, incorporating factors analogous to those used in economic analysis, such as the contribution of each action to finished-product throughput, the implied energy usage associated with machine operation and agent movement, and the overall progression toward completing value-adding steps in the manufacturing workflow.

An important feature for production-related simulations is the possibility of action masking. This approach enables certain actions to be made unavailable to the agent under specific conditions; consequently, it narrows the actual search/learning space. Action masking improves both learning efficiency and realism by ensuring that agents do not have to deal with invalid or infeasible choices. These features collectively make Gymnasium an ideal framework for modeling and simulating the analyzed production cell.

2.2. RL learning toolkit – Stable Baselines3 (SB3)

SB3 is one of the most widely used reinforcement learning toolkits in the research community, known for its balance between ease of use, reliability, and performance. It provides high-quality, well-maintained implementations of many state-of-the-art (SOTA)

algorithms, making it suitable both for academic research and for applied industrial studies.

One of the main advantages of SB3 is its compatibility with Gymnasium environments, a synergy rooted in the fact that both frameworks were originally developed under the OpenAI ecosystem before their maintenance was handed over to independent developer communities. This shared heritage ensures consistent design principles and smooth interoperability, which allows agents to be trained seamlessly in both standard benchmark tasks and in user-defined environments such as the simulated production cell. The standardized software (Application Programming Interface, API) makes it straightforward to integrate agents with minimal overhead, reducing the time and effort needed for experimentation.

The SB3 library includes implementations of widely adopted algorithms such as Proximal Policy Optimization (PPO), Deep Q-Networks (DQN), Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), and A2C, among others. These algorithms represent the current state of the art in reinforcement learning and provide researchers with a diverse toolkit to explore different approaches to optimization problems. The availability of multiple algorithms within a unified framework makes benchmarking more efficient, as identical environments and evaluation procedures can be used across all tested methods.

In addition to its broad algorithm support, SB3 provides a variety of utilities and callbacks that simplify tasks such as monitoring training progress, logging performance metrics, saving checkpoints, and evaluating agents at regular intervals. These features support both reproducibility and scalability of experiments, which are essential in scientific and practical settings.

Beyond the main SB3 package, the developers also maintain Stable-Baselines3 Contrib, a supplementary repository that contains experimental or newly implemented algorithms not yet fully integrated into the main library. Examples include algorithms with novel features, research-focused extensions, or alternative implementations under active development. While these contributions may not always offer the same level of stability or support as the core algorithms, they provide valuable opportunities for early experimentation with emerging reinforcement learning methods.

In this study, SB3 Contrib is relevant as it offers access to an action-masked variant of PPO, which directly supports the handling of invalid actions within the production cell environment.

2.3. Action masking

In the production cell under study, two product movement and manipulation units are responsible for executing actions that directly influence the throughput and efficiency of the system. A particular challenge arises from the fact that these units operate independently of one another and must select actions at changing time intervals, as some tasks require longer durations to complete than others. This dynamic timing introduces additional complexity into the control problem, since at any given moment only a subset of actions may be valid for each unit. Moreover, to achieve an optimal solution, they must coordinate their actions.

The overall action space of the environment is defined as the union of the possible actions of both manipulation units; consequently, the production cell logistics and scheduling must be solved. This design allows for a flexible representation of the system but also means that, without restrictions, an agent could attempt to select actions that are not feasible under the current conditions (e.g., a unit attempting to place a product while it is still busy with a previous operation).

In the initial version of the environment, the validity of an action was checked only after it was selected. If the chosen action was infeasible, the environment responded by returning to the same state along with a highly negative reward. While this approach discouraged invalid choices, it also forced the agent to spend significant training time exploring invalid decisions, which slowed, if not derailed, the convergence and reduced efficiency.

To address this limitation, action masking was introduced. Action masking dynamically restricts the available choices to only those actions that are feasible (and rational) for the product movement units at a given time step of the production cell. This ensures that the agent does not waste training effort on invalid decisions and accelerates convergence toward useful policies.

The masking mechanism can be implemented using a custom wrapper, following the modular design principles of Gymnasium. Wrappers allow modifications to the environment's interface without altering its internal logic, ensuring that the core simulation remains unchanged and reusable. In this case, the wrapper dynamically generates a mask of feasible actions at each time step, based on the availability and status of the two product movement units. This mask is then passed to the reinforcement learning agent, restricting its choices to only the valid subset of the full action space. By embedding action masking into a wrapper, the approach remains highly transparent, easily extendable, and consistent with the Gymnasium framework's (and the RL community's) best practices.

In addition to the basic feasibility mask, additional masking strategies were introduced to accelerate training according to the complexity of the environment and the task.

The first additional mask is based on the production plant's internally used fixed logic system. This rule-based decision system reflects how actions are typically selected in practice (before the introduction of RL). By embedding this logic into a mask, the simulated environment could be tested against real-world operational data, ensuring that the model produces results consistent with factory behavior. The outcomes of this mask were also validated in discussions with operators, providing an important check for the real-world validity of the simulated environment.

The second extension is an expanded logic mask, designed specifically for training purposes. This mask

combines elements of rule-based control with reinforcement learning flexibility. At each decision point, the mask allows one action derived from the operator-selected (rule-based) choice, if available, together with two additional actions permitted by the environment's feasibility constraints. This hybrid approach ensures that the agent explores beyond the fixed logic while still retaining guidance from expert knowledge. Importantly, the setup accounts for configurations where the rule-based system might not propose any valid action, in which case only the environment-based feasible actions remain available.

2.4. Benchmark for production cell optimization

To evaluate the effectiveness of reinforcement learning in the given production cell optimization problem, a benchmarking framework was established. Benchmarking in this context refers to the systematic process of running, logging, measuring, and comparing multiple algorithms under controlled conditions. Each algorithm was trained and tested in the same environment setup, using identical reward functions, action masking configurations, and hyperparameter settings.

Performance was tracked through a combination of quantitative metrics and training logs. Quantitative measures included cumulative reward, throughput, convergence speed, and stability across runs, while logging tools provided insight into training dynamics such as episode lengths, reward distributions, and action frequencies. These metrics formed the basis for assessing both learning efficiency and practical applicability in the production domain.

An important aspect of benchmarking in reinforcement learning is handling the stochastic behavior inherent in both the algorithms and the environment. Since outcomes can vary significantly across different training seeds and trajectories, relying on a single run may lead to misleading conclusions. To mitigate this, both training and evaluation were performed with parallelization. During training, the agent interacted simultaneously with multiple copies of the environment, enabling faster learning and greater robustness. This approach not only accelerates convergence by providing more diverse experiences per time step but also prevents stagnation in cases where one environment instance becomes stuck in a local pattern – since the others continue to progress – the agent still receives meaningful updates. Similarly, evaluation was carried out on multiple parallel environments, producing more robust performance estimates and reducing the variance in reported results. By averaging across multiple trajectories, the benchmarking process delivered a more accurate assessment of an agent's true capabilities. This methodology provided a fair and reliable comparison of algorithms, ensuring that the conclusions drawn from the experiments reflect genuine performance differences rather than random fluctuations.

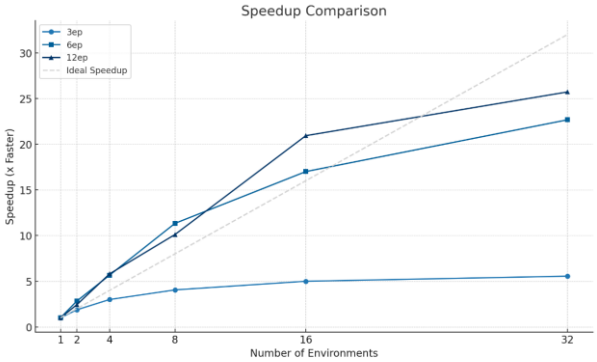

Figure 1: Speed increase of the training according to the number of parallel environments

3. Results and analysis

The results are structured to provide both quantitative comparisons and qualitative insights into the agent's behavior. Given the stochastic nature of reinforcement learning and the environment itself, each experiment was repeated multiple times using different random seeds; moreover, both training and evaluation were conducted in parallelized settings. This ensured that the reported results are statistically robust and not artifacts of individual runs. Performance was assessed with respect to several key indicators: cumulative reward, convergence speed, learning stability, and production throughput efficiency. The results are further compared to baseline rule-based strategies in the production plant, providing a practical point of reference. Parallelization not only increases the reliability of the results by averaging over multiple environments, but it also substantially reduces the required runtime compared to serial execution. *Figure 1* illustrates the achieved speedup across different numbers of environments measured at different training lengths showing that higher workloads scale more efficiently and approach ideal parallel performance.

3.1. Production performance comparisons

The first set of results focuses on evaluating the performance of four promising reinforcement learning algorithms: Advantage Actor-Critic (A2C), Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), and Maskable PPO. Each algorithm was trained in the same production cell environment using identical hyperparameter configurations (where applicable), with the simplest environmental action masking applied only in the case of Maskable PPO.

Since finishing even a single product is a difficult task, the performance was measured using average cumulative reward per episode (*Table 1*). This has to be calculated over time and not per step, since different policies can lead to different action taking sequences, and The lengths of these actions may vary. Additionally, throughput and product flow could be used efficiently as

Table 1: Agent comparisons by average cumulative reward per episode

	A2C	TRPO	PPO	MPPO
Min	53.0	-37.6	51.6	98.0
Max	119.6	57.9	118.9	390.1
Average	58.1	52.7	58.2	269.7

Table 2: Reward settings comparison by average cumulative reward per episode

	Min	Max	Average
Setting 1	28	35	31.5
Setting 2	26	37	31.5
Setting 3	35	48	41.5
Setting 4	51	67	59

practical measures of performance within the production cell context.

Ideally, the algorithms would have been compared based on produced products per hour, which is the most meaningful real-world performance indicator in a manufacturing context. However, during evaluation it became evident that, except Maskable PPO, none of the baseline algorithms were able to complete even a single finished product within the allotted benchmark action steps. As a consequence, a throughput-based comparison was not feasible, and performance had to be evaluated instead through the cumulative reward each agent achieved.

The results still reveal clear performance differences across the algorithms. TRPO struggled to achieve consistent results, with some runs producing negative rewards and only limited improvement in the best cases. A2C and PPO both reached moderate levels of performance, with comparable averages and similar ranges of outcomes. In contrast, Maskable PPO achieved substantially higher scores, both in terms of average and maximum reward, while also maintaining a higher minimum performance compared to the other algorithms. This demonstrates the effectiveness of incorporating action masking, which prevents agents from exploring infeasible actions and accelerates convergence toward productive strategies. (The negative reward given for the invalid action is artificial and only affects the three non-maskable algorithms, since Maskable PPO is unable to return such action; this may skew the results.)

These findings confirm that Maskable PPO is particularly well-suited for the production cell optimization problem, combining the stability of PPO with additional robustness and efficiency derived from masking.

3.2. Other improvements

In this work, "parameter tuning" refers not to algorithmic hyperparameters, but to the relative weights assigned to different components of the reward function in the environment model.

By adjusting these weights, different emphases could be placed on specific aspects of the optimization problem. For example, increasing the penalty weight on idle time encouraged the agent to prioritize resource utilization, while raising the throughput reward weight emphasized overall production speed. Systematic variation of these reward weights with grid search allowed the identification of balanced configurations that promoted long-term efficiency.

The experiments confirmed that reward shaping plays a decisive role in guiding the agent toward realistic and productive strategies. In particular, properly tuned reward components reduced variance in training and produced agent behaviors that more closely aligned with the expectations of factory operators, while other settings resulted in a still-standing behavior to evade high negative rewards. The final reward shape was obtained through a manual grid-search procedure, iteratively adjusting the component weights until further deviations produced no meaningful differences in achieved throughput.

As shown in *Table 2*, in addition to the final reward tuning result (Setting 1), the effect of different masking strategies was analyzed. As discussed in the methodology, three main masking configurations were available: the basic feasibility mask, the rule-based mask reflecting the plant's fixed internal logic, and the expanded logic mask combining rule-based guidance with additional feasible actions (Settings 2-4). Each of these masking approaches influences the agent's exploration strategy in different ways.

The benchmarking results showed that the basic feasibility mask was sufficient for ensuring valid actions but provided less guidance, resulting in slower convergence, while the expanded logic mask produced the best training outcomes. This configuration balanced the reliability of rule-based decisions with the flexibility of reinforcement learning exploration, allowing the agent to escape the limitations of purely rule-based control while still avoiding unproductive actions. This setup achieved 82% efficiency with respect to total product-throughput compared to the purely rule-based approach used until now; however, it also provided sufficient insight to suggest changes to the current factory-implemented rule-set.

Taken together, these experiments highlight that, beyond algorithm choice, reward tuning and carefully designed masking strategies are crucial for achieving efficient and robust training in complex industrial environments.

4. Conclusion

The reported research demonstrated the successful application of reinforcement learning to the optimization of a real-world production cell, delivering both scientific insights and engineering contributions. As the resulting simulation provides a valid digital counterpart of the production process, this work therefore bridges the gap

between abstract reinforcement learning research and practical industrial deployment.

On the scientific side, the paper established that incorporating action masking is essential in complex production environments, as it substantially improves convergence speed, stability, and robustness compared to traditional penalty-based methods. The investigation of different masking strategies showed that hybrid approaches, combining feasibility constraints with already tested rule-based logic, can significantly enhance training efficiency. This is a very important contribution for practical implementation of reinforcement learning based industrial solutions.

Acknowledgements

The research was supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory and by the European Commission through the DiGreeS project under grant No. 101178079.

On behalf of project "Comprehensive testing of machine-learning algorithms" we thank for the use of HUN-REN Cloud (https://science-cloud.hu/) that significantly helped us achieve the results published in this paper.

REFERENCES

[1] Kegyes, T.; Süle, Z., Abonyi, J.: The applicability of reinforcement learning methods in the development of Industry 4.0 applications, *Complexity,* 2021, **2021**(1), 7179374, DOI: 10.1155/2021/7179374

[2] Tóth, Sz.H.; Viharos, Zs.J.; Bárdos, Á.; Szalay, Zs.: Sim-to-real application of reinforcement learning agents for autonomous, real vehicle drifting, *Vehicles*, 2024, **6**(2). 781–798, DOI: 10.3390/vehicles6020037

[3] Panzer, M.; Bender, B.: Deep reinforcement learning in production systems: a systematic literature review, *Int. J. Prod. Res*., 2022, **60**(13), 4316–4341, DOI: 10.1080/00207543.2021.1973138

[4] Khdoudi, A.; Masrour, T.; El Hassani, I.; El Mazgualdi, C.: A deep-reinforcement-learning-based digital twin for manufacturing process optimization, *Systems* 2024, **12**(2), 38, DOI 10.3390/systems12020038

[5] Wei, Z.H., Yan, L., Yan, X.: Optimizing production with deep reinforcement learning. *Int. J. Simul. Model*., 2024, **23**(4), 692–703, DOI: 10.2507/IJSIMM23-4-CO17

[6] Stappert, M.; Lutz, B.; Goby, N.; Neumann, D.: Integrating human knowledge through action masking in reinforcement learning for operations research, *arXiv*, 2025, arXiv:2504.02662, DOI: 10.48550/arXiv.2504.02662

[7] Laflamme, C.; Doppler, J.; Pálvölgyi, B.; Dominka, S.; Viharos, Zs.J.; Haeussler, S.: Explainable reinforcement learning for powertrain control engineering, *Eng. Appl. Artif. Intell*., 2025, **146**, 110135, DOI: 10.1016/j.engappai.2025.110135

[8] Ali A.M.; Tirel, L.; Hashim H.A.: Novel multi-agent action masked deep reinforcement learning for general industrial assembly lines balancing problems, *J. Autom. Intell.*, 2025, **4**(4), 299–311, DOI: 10.1016/j.jai.2025.07.001

[9] Overbeck, L.; Hugues, A.; May, M.C.; Kuhnle, A.; Lanza, G.: Reinforcement learning based production control of semi-automated manufacturing systems, *Procedia CIRP*, 2021, **103**, 170–175, DOI: 10.1016/j.procir.2021.10.027

[10] Steinbacher, L.M; Wegmann, T.; Freitag, M.: Production control with Reinforcement Learning for a matrix-structured production system, *Int. J. Prod. Res.*, 2025, **63**(11), 4114–4136, DOI: 10.1080/00207543.2024.2436126

[11] Li, C.; Chang, Q.; Fan, H.-T.: Multi-agent reinforcement learning for integrated manufacturing system-process control, *J. Manuf. Syst.*, 2024, **76**, 585–598, DOI: 10.1016/j.jmsy.2024.08.021

[12] Johnson, D.; Chen, G.; Lu, Y.: Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell, *IEEE Robot. Autom. Lett.*, 2022, **7**(3), 7684–7691, DOI: 10.1109/LRA.2022.3184795

[13] Panzer, M.; Bender, B.; Gronau, N.: A deep reinforcement learning based hyper-heuristic for modular production control, *Int. J. Prod. Res.*, 2023, **62**(8), 2747–2768, DOI: 10.1080/00207543.2023.2233641

[14] Zhang, L.; Yan, Y.; Yang, C.; Hu, Y.: Dynamic flexible job-shop scheduling by multi-gent reinforcement learning with reward-shaping, *Adv. Eng. Inform.*, 2024, **62**, 102872, DOI: 10.1016/j.aei.2024.102872

[15] Viharos, Zs.J.; Jakab, R.: Reinforcement learning for statistical process control in manufacturing, *Measurement,* 2021, **182**, 109616, DOI: 10.1016/j.measurement.2021.109616

[16] Du, Y.; Li, J.-Q.: A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling, *Int. J. Prod. Econom*., 2024, **268**, 109102, DOI: 10.1016/j.ijpe.2023.109102

[17] Clemente, A.V.; Castejón, H.N.; Chandra, A.: Efficient parallel methods for deep reinforcement learning, *arXiv*, 2017, arXiv:1705.04862, DOI: 10.48550/arXiv.1705.04862

[18] Ali, A.M.; Tirel, L.: Action masked deep reinforcement learning for controlling industrial assembly lines, *Proc. IEEE World AI IoT Congress (AIIoT 2023), Seattle, WA, USA*, 2023, 0797–0803, DOI: 10.1109/AIIoT58121.2023.10174426

[19] Dulac-Arnold, G., Levine, N., Mankowitz, D.J.; Li, J.; Paduraru, C.; Gowal, S.; Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis, *Mach. Learn.*, 2021, **110**(9), 2419–2468, DOI: 10.1007/s10994-021-05961-4