Maskable Proximal Policy Optimization for Safe Control of a Silo Level Balancing Process

Kerollan S. Ramos*†, Gustavo Pessin*‡, Thomás Pinto*‡,
*Instituto Tecnológico Vale, Ouro Preto – MG, Brazil
†Department of Mining Engineering, Universidade Federal de Ouro Preto, Ouro Preto – MG, Brazil
‡Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Instituto Tecnológico Vale and Universidade Federal de Ouro Preto, Ouro Preto – MG, Brazil

***Abstract*—Maintaining balanced material levels across a set of silos is a safety-critical control task in industrial mineral processing plants, commonly addressed through tripper car positioning under strict operational constraints. This work investigates the application of Maskable Proximal Policy Optimization (MaskablePPO) as a Reinforcement Learning (RL) framework for safety-aware decision making, using the silo level balancing problem as an industrial inspired test environment. Invalid action masking is employed as a hard constraint mechanism to prevent invalid actions, while a curriculum learning strategy is introduced to improve training stability. The agent is trained and evaluated in a simulated environment characterized by complex physical constraints, stochastic dynamics, and interlock logics. Experimental results show that the MaskablePPO outperforms a standard Proximal Policy Optimization (PPO) baseline in terms of training convergence, computational efficiency, and constraint satisfaction, while eliminating invalid actions by design. These results highlight the effectiveness of MaskablePPO for RL in constrained industrial settings.**

***Index Terms*—reinforcement learning, maskable proximal policy optimization, invalid action masking, silo level balancing, industry**

I. Introduction

Reinforcement Learning (RL) is a learning paradigm in which an agent learns how to behave through direct exposure to an environment for which it has no prior knowledge. The agent is incentivized to interact with the environment by taking action, and for every action and resulting state, it receives a reward (positive, negative, or zero) that reflects the immediate quality of the outcome. This cycle of trial-and-error interaction is repeated until the agent learns a policy that enables it to behave so as to maximize the expected cumulative reward [1].

This self-learning capability has allowed RL to achieve tremendous success in diverse fields such as board games [2], e-sports [3], robotics [4], among many others. In the industry, despite persistent concerns regarding the direct training of agents in real-world environments, which may be unsafe or economically too expensive [5], practical approaches have been adopted to enable agent training. In such cases, agents are typically trained either in simulated process environments [6] or using historical data through offline approaches [7].

Despite these advances, deploying trained RL agents in industrial settings still raises reliability concerns. In real-world applications, operating conditions are prone to changes, which may result in circumstances that differ considerably from those encountered during offline training, leading to out-of-distribution (OOD) scenarios. Under such conditions, the agent may exhibit degraded performance and generate invalid or infeasible actions that violate operational constraints, potentially triggering undesirable operational events [8].

These challenges motivated the formulation of Safe Reinforcement Learning (SafeRL) algorithms, focusing on the learning of optimal policies while respecting safety, operational, or functional constraints [9]. Constraint handling can be broadly categorized into two approaches. Soft constraint methods, such as Lagrangian relaxation [10], typically formulate the learning problem as a Constrained Markov Decision Process (CMDP). In this framework, constraint violations are discouraged through penalty terms in the expected cost. However, unsafe or infeasible actions remain in the policy's support and may still be selected during training or execution. Conversely, hard constraint approaches enforce safety by restricting the agent's action space, ensuring that invalid actions cannot be selected. In discrete action spaces, this is commonly achieved through invalid action masking, which removes invalid actions prior to policy sampling [11], [12]. By excluding infeasible actions, invalid action masking ensures constraint satisfaction at the action level by design and has been shown to correspond to a valid policy gradient update under a state-dependent, differentiable transformation of the policy [13].

Notably, by preventing invalid actions from being considered, invalid action masking not only enforces feasibility constraints but can also improve convergence during training of RL agents [14]. Several works have explored the use of invalid action masking in RL across a wide range of applications. In this context, Proximal Policy Optimization (PPO) has been frequently adopted as a base algorithm for action masked RL implementations. For instance, Guo et al. [15] have proposed two Maskable Proximal Policy Optimization (MaskablePPO) algorithms for autonomous vehicle decision-making at unsignalized intersections. The results have shown improved convergence and safety-related performance compared to unmasked PPO. Ali and Tirel [16] have proposed different action masked deep RL frameworks for controlling industrial assembly lines, addressing the problem under numerous operational

constraints. Among the tested frameworks, MaskablePPO has achieved faster and more stable convergence toward optimal solutions while ensuring constraint satisfaction.

In this work, we developed and evaluated the application of RL to a silo level balancing problem, commonly encountered in mineral processing plants and characterized by strict operational constraints, using a simulated environment. The RL agent is developed to control the tripper car by selecting its destination silo and feeding dwell time. The proposed framework is based on MaskablePPO, whose value and policy networks are implemented using a Multilayer Perceptron (MLP), and incorporates a curriculum learning strategy to improve training convergence. By explicitly preventing invalid actions, this approach promotes safer operation and improved operational continuity.

The source code for this work is publicly available at: https://github.com/RamosKS/IJCNN-MaskablePPO-Silo-Balancing.

II. MaskablePPO

The PPO algorithm has become one of the most widely adopted RL algorithms due to its favorable trade-off between performance, stability, and implementation simplicity. The PPO is derived from the Trust Region Policy Optimization (TRPO) algorithm, and follows an Actor–Critic architecture, commonly implemented using neural networks such as MLPs [17].

PPO belongs to the class of policy gradient methods and is specifically categorized as an on-policy algorithm. Consequently, it learns exclusively from data collected by the current policy, discarding experience samples after a few policy updates [17]. A fundamental feature of PPO is the use of a clipped objective function, which constrains the size of policy updates and improves training stability. Rather than enforcing a strict trust region, PPO limits the deviation between the new and previous policies during a single update step by clipping the probability ratio. The resulting objective function is given by

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

where $L^{\text{CLIP}}(\theta)$ denotes the PPO clipped surrogate objective under parameters θ, $\hat{\mathbb{E}}_t[\cdot]$ denotes the sample average over timesteps t, $\hat{A}_t$ denotes the advantage estimate at time step t, ϵ is the clipping constant, and $\rho_t(\theta)$ represents the probability ratio between the new and old policies under parameters θ.

The PPO agent interacts with the environment for a fixed number of steps, collecting trajectories of the form (s_t, a_t, r_t, s_{t+1}). These tuples, representing, respectively, the state, action, reward, and next state at time t, are temporarily stored in a rollout buffer. After the data collection phase, PPO leverages all collected samples to compute the advantage estimates and update the neural network parameters. Policy updates are typically performed using mini-batches to reduce gradient variance and improve generalization. After a given number of optimization epochs, the rollout buffer is cleared in order to mitigate overfitting to recently collected data.

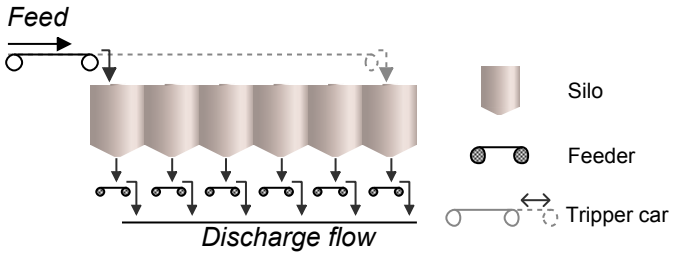

Fig. 1. Fluxogram of a set of silos.

The deployment of PPO agents in industrial environments requires strict adherence to safety and operational constraints. MaskablePPO is a variant of PPO designed for environments in which certain actions are illegal in specific states. In a standard PPO, the policy is defined over the entire action space for all states. When the agent selects an invalid action, the environment typically responds by either returning the same state or transitioning to a corrected state, while assigning a large negative reward. This strategy is inefficient, as it wastes interaction steps exploring actions that are known to be invalid.

In contrast, MaskablePPO enforces constraints directly at the policy level by intervening on the action logits before probability computation via the softmax function. At each state s, the environment provides a binary action mask set M, where for every action i, $M_i = 1$ indicates a valid action and $M_i = 0$ denotes an invalid one. Invalid actions are handled by replacing the corresponding logits with a large negative value. Given the original logits, z_i, MaskablePPO produces masked logits, z_i^{masked}, according to

$$z_i^{masked} = z_i \cdot \mathbb{I}[M_i = 1] - \infty \cdot \mathbb{I}[M_i = 0],$$

where $\mathbb{I}[\cdot]$ is an indicator function that equals 1 if the condition is true; and 0 otherwise. After applying the softmax operation, this transformation assigns near-zero probability to invalid actions, effectively preventing their selection [18], [19].

The key distinction of MaskablePPO lies not only in preventing the execution of invalid actions but also in how this constraint affects the policy gradient. By assigning near-zero probability to invalid actions, the probability mass is renormalized exclusively over the valid action subset. Consequently, the policy gradient is computed solely within the subspace of valid actions. This results in faster and more stable learning, as gradient noise induced by invalid actions is removed from the optimization process [18], [19].

III. The material distribution problem

Silos are widely used in the mineral industry as storage structures that act as buffers between processing stages, absorbing upstream flow fluctuations and ensuring downstream material availability [20]. As illustrated in Fig. 1, they are typically arranged in parallel and connected to feeders (variable speed conveyor belts) that regulate discharge flow.

To ensure the effective use of these structures, a level-balancing control system is needed to evenly distribute material among the silos, enabling uninterrupted operation. This is

typically performed by keeping their levels within predefined operational bounds, allowing for normal fluctuations while preventing reaching minimum or maximum limits.

When these limits are exceeded, safety interlock mechanisms are activated to prevent silo overfilling or emptying, setting the affected silo to an inactive state until recovery conditions are met. In particular, if the level falls below the minimum limit, an interlock is triggered that shuts down the feeder located beneath the affected silo. Conversely, if the level exceeds the maximum limit, the upstream circuit is stopped, preventing additional material from entering the silo and potentially causing upstream congestion.

Among the control strategies employed, it usually involves controlling the behavior of a tripper car [21], [22]. The tripper car is a movable unit that moves along a conveyor belt installed above the silos and is used to allocate material among them. By moving along the conveyor, the tripper car delivers the material flow into a single silo at a time. The amount of material delivered to each silo is determined by the dwell time of the tripper car above that silo, together with the conveyor discharge rate.

IV. Experimental setup

A. Problem Definition

The simulation environment models an array of $N = 6$ adjacent silos, indexed by $j \in \{0, \dots, N-1\}$. Each silo is defined by a composite geometry consisting of an upper square prism and a lower truncated square pyramid (hopper). To introduce structural variability and improve the generalization capability of the control policy, the geometric parameters are stochastically generated at the beginning of each episode. The prism side length, L, is uniformly sampled from the interval $[4.0, 8.0]\,\mathrm{m}$ using a discretization step of $0.1\,\mathrm{m}$. The total height, H_{tot}, is similarly sampled from the interval $[6.0, 8.0]\,\mathrm{m}$. The pyramid height is fixed at a ratio of $H_{\mathrm{pyr}} = 0.3\,H_{\mathrm{tot}}$, with the remaining height constituting the prism section, H_{prism}. The outlet dimension of the hopper is defined proportionally as $l = 0.25\,L$. The maximum storage volume, $V_{\max}$, is the sum of the prism volume, $V_{\mathrm{prism}} = L^2 H_{\mathrm{prism}}$, and the pyramidal frustum volume, calculated as

$$V_{\mathrm{pyr}} = \frac{H_{\mathrm{pyr}}}{3}\left(L^2 + l^2 + L \cdot l\right).$$

The material level is represented by a normalized vertical coordinate, defined as the percentage of the total height. This normalized level, $\bar{h} \in [0, 1]$, is mapped to the physical height coordinate, $h = \bar{h} H_{\mathrm{tot}}$, with $h \in [0, H_{\mathrm{tot}}]$. Due to the pyramidal geometry of the hopper, the volume-to-height mapping, $h(V)$, and its inverse, $V(h)$, are non-linear within this region. Moreover, for a given physical height, h, the instantaneous cross-sectional side length, $L(h)$, corresponding to the lateral dimension of the square cross-section at that height, varies linearly along the hopper section, according to

$$L(h) = l + \left(\frac{L - l}{H_{\mathrm{pyr}}}\right) h, \quad \text{for } 0 \le h \le H_{\mathrm{pyr}}.$$

Consequently, the $V(h)$ is piecewise defined, requiring numerical inversion (e.g., bisection method) to compute the material level from the simulated volume.

1) Initialization: At the start of each episode, up to two silos are randomly selected to be blocked (out of service). The initial fill levels of the active silos are sampled uniformly from the interval $[40, 80]\%$.

2) Safety Interlocks and Hysteresis: The control process implements interlock logic:

- Emptying Protection: If a silo level drops below 5%, discharge is stopped. To prevent oscillation, drainage only resumes once the level recovers to at least 10%.
- Overfilling Protection: If a silo level exceeds 90%, feeding into that silo is strictly prohibited. The interlock follows a hysteresis curve, lifting the feeding ban only when the level drops below 80%.
- Blocked Silo Protection: Silos flagged as blocked cannot receive or discharge material, and attempting to feed them is a policy violation.

3) Operational Dynamics and Constraints: The process is fed by a tripper car moving at a constant speed s_{trip}, sampled at the beginning of each episode from the interval $[s_{\min}, 0.35]\,\mathrm{m/s}$. The lower bound $s_{\min}$ is determined via a heuristic simulation focused on the system's critical constraint: maintaining equilibrium between the two extreme silos. By factoring in the specific discharge rates and a fixed moderate dwell time of $30\,\mathrm{s}$, the algorithm identifies the minimum velocity required to offset drainage in these extreme silos during the round-trip travel.

4) Flow Rates and Topological Compensation: The process feed rate, $F_{j,t}$, and the discharge rates, $D_{j,t}$, are modeled as Discrete-Time Markov Chains (DTMCs) to represent stochastic variability, characterized by nominal mean rates of $\mu_F \approx 0.48\,\mathrm{m^3/s}$ and $\mu_D \approx 0.08\,\mathrm{m^3/s}$, respectively. While the system enforces mass balance by scaling active discharge rates by N/N_{active}, this capacity is further adjusted to account for the operational dead-time spent traversing gaps, defined as inactive silos situated between the first and last active units. This topological efficiency discount is applied to the mean discharge rate, μ'_D, according to

$$\mu'_D = \mu_D \cdot \frac{N}{N_{\mathrm{active}}} \cdot \left(1 - n_{\mathrm{gaps}} \cdot \phi(L, s_{trip})\right),$$

where n_{gaps} is the number of identified gaps and $\phi(L, s_{trip})$ is a penalty factor per gap. This factor is calculated via a min-max normalization of the traversal cost $L/\sqrt{s_{trip}}$, mapping it to a range between 0.1% and 4%. This ensures that fragmented configurations, which impose higher traversal latency, result in a proportional reduction of the effective drainage capacity compared to contiguous layouts.

B. Reinforcement Learning Formulation

The environment is formulated as a Partially Observable Markov Decision Process (POMDP), with a state representation that provides sufficient statistics for the defined control objective.

1) State Space: The observation vector o_t concatenates four feature sets:

$$o_t = [\Delta\vec{\ell}_t, \vec{\ell}_t, \vec{Q}_t, J_t],$$

where $\Delta\ell_{j,t} \in [-1, 1]$ denotes the rate of change of the fill levels, $\ell_{j,t} \in [0, 1]$ represents the normalized silo fill levels, $Q_{j,t} \in \{0, 1\}$ is the binary mask of active silos, and $J_t \in [0, 1]$ encodes the normalized position of the tripper car,

2) Action Space: The agent controls the tripper car via a MultiDiscrete action space $\mathcal{A} = \{0, \ldots, 5\} \times \{0, \ldots, 6\}$. The first component selects the target silo index, J_{dest}, and the second selects the dwell time index $k \in \{0, 1, \ldots, 6\}$, corresponding to dwell times $\{0, 10, \ldots, 60\}$ s. Constraints are enforced to prevent the agent from selecting the current silo or any inactive or blocked silo as a destination. Actions violating these constraints are classified as invalid. To ensure simulation continuity in algorithms that do not support explicit invalid action masking, the environment implements a deterministic fallback mechanism. If an invalid or blocked silo is selected as the destination, J_{inv}, the environment overrides the action and redirects the tripper car to the nearest valid silo index, J^*, according to

$$J^* = \arg\min_{J \in \mathcal{J}_{\text{valid}}} |J - J_{\text{inv}}|.$$

The original decision is nevertheless flagged as invalid and penalized during reward computation.

3) Reward Function: The control objective is to maintain continuous operation while minimizing safety violations. The reward r_t is defined as a negative cost function

$$r_t = -(\alpha \cdot C_{low} + \omega \cdot C_{high} + \sigma \cdot C_{inactive} + \kappa \cdot \mathbb{I}_{invalid}),$$

where C_{low} and C_{high} are the amounts of low-level and high-level interlocks, respectively. The term $C_{inactive}$ represents the inactive-travel penalty, counting the number of times the tripper car traverses a silo that is currently inactive or blocked. The term $\mathbb{I}_{invalid} \in \{0, 1\}$ is the invalid action indicator, triggered when the agent attempts to select an inadmissible destination, regardless of the subsequent fallback correction. Based on the calibrated environment, the weights are set to $\alpha = 0.8$, $\omega = 0.9$, $\sigma = 0.2$, and $\kappa = 1.0$, heuristically chosen to approximate the relative practical impact of each type of violation.

C. Training procedure

The policy was trained using the MaskablePPO algorithm to explicitly handle validity constraints within the action space. The value and policy networks share a MLP architecture with two hidden layers of 64 neurons each, all activated using the hyperbolic tangent (tanh) function.

1) Hardware and Parallelization: Training was conducted on a workstation equipped with an AMD Ryzen 9 9950X3D processor. We utilized `SubprocVecEnv` to parallelize data collection across $n_{\text{envs}} = 28$ independent environment instances. This configuration yields a rollout buffer of $57,344$ transitions per update ($n_{\text{steps}} = 2,048 \times n_{\text{envs}}$).

2) Curriculum Learning: A progressive curriculum strategy was employed to improve convergence. The maximum episode length, $T_{\max}$, expressed in timesteps, was incremented through stages, taking values in the set $\{100, 250, 500, 1000, 2000\}$. Transitions between stages utilized dynamic reconfiguration to update $T_{\max}$ without process re-instantiation (hot-swapping). The training budget was allocated incrementally: 2M timesteps for the initial stages ($T_{\max} \in \{100, 250\}$), 4M for intermediate stages ($T_{\max} \in \{500, 1000\}$), and 8M for the final stage ($T_{\max} = 2000$), totaling 20M timesteps. All hyperparameters were kept at their default values, except for $\gamma = 0.995$, favoring long-term reward optimization, a learning rate of 3×10^{-4}, regulating the size of policy updates, and an entropy coefficient of 0.01, encouraging exploration of alternative strategies, and a batch size of 1024, defining the number of samples used per update.

3) Implementation and Evaluation: Policy evaluation was conducted periodically using a deterministic evaluation environment running in a single process (`DummyVecEnv`) to ensure reproducibility, with performance assessed over 1,000 episodes, using an evaluation frequency set to 10,000 training steps. In the constrained learning configuration, invalid actions were explicitly masked during the forward pass using the MaskablePPO algorithm from the `sb3_contrib` (version 2.3.2) library. Conversely, the standard PPO baseline was implemented using `stable_baselines3` (version 2.3.2), relying on the reward penalty and the fallback mechanism to learn feasibility constraints. The environment was integrated under the `gymnasium` (version 1.1.1) framework. Both models shared the same training configuration and random seed.

V. Results

This section evaluates the proposed MaskablePPO agent against the standard PPO baseline. We analyze training stability, computational efficiency, and the impact of invalid action masking on performance and policy robustness under safety constraints.

A. Training Analysis and Computational Cost Performance

The training progression over 20×10^6 steps reveals a clear contrast between the two agents in terms of learning dynamics and optimization stability. As shown in Fig. 2, MaskablePPO outperformed the standard PPO for most of the training duration across all curriculum stages. The differences in learning behavior are also reflected in the training loss shown in Fig. 3. Both models exhibited a sharp loss reduction during the early stages of training, reaching fractional error values around 1M steps. After this point, MaskablePPO showed lower-size fluctuations and a more consistent downward trend in the training loss, whereas the standard PPO exhibited substantially higher loss oscillations and slower progression toward comparable error levels, which were only reached after approximately 13M steps. This behavior suggests that the availability of invalid actions in standard PPO leads to noisier gradient updates, increasing loss variance throughout training.

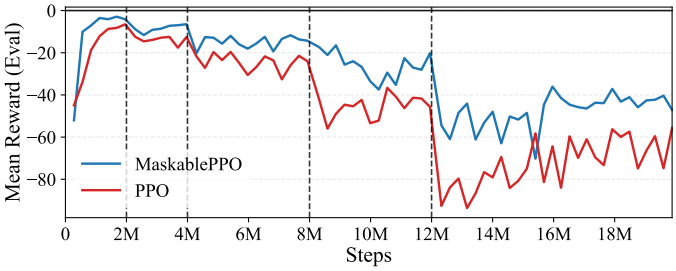

Fig. 2. Evaluation mean reward over steps for MaskablePPO and PPO. Dashed vertical lines indicate curriculum learning stage transitions.

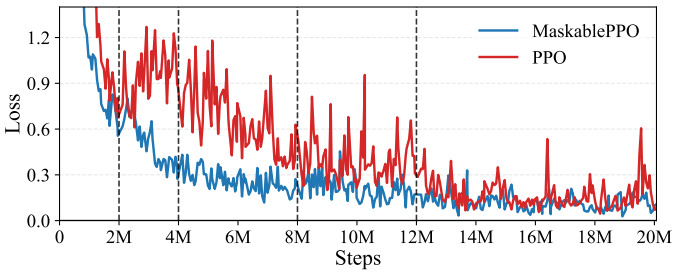

Fig. 3. Training loss over steps for MaskablePPO and PPO. Dashed vertical lines indicate curriculum learning stage transitions.

Beyond improved convergence properties, the enhanced training stability of MaskablePPO translates directly into superior computational efficiency. MaskablePPO required 5.705 h of training time, corresponding to a 9% reduction in total wall-clock time compared to the 6.27 h required by standard PPO.

B. Statistical Performance and Infraction Analysis

To validate the robustness of the learned policies, both agents were evaluated over 1,000 randomized scenarios (10,000 steps each) using a deterministic policy and shared seeds. The statistical summary is presented in Table I.

The results reveal distinct constraint-handling behaviors between the two approaches. MaskablePPO never selected actions classified as invalid across all evaluated scenarios, directly confirming the effectiveness of invalid action masking in enforcing hard constraints. In contrast, the standard PPO was able to learn to avoid invalid actions in most cases, achieving a third quartile as low as one invalid action per episode. However, the presence of a maximum value of 521 indicates that, under specific conditions, the agent may still incur a large number of constraint violations, highlighting a lack of robustness in rare but critical scenarios.

While the standard PPO exhibited fewer low-level infractions on average, MaskablePPO consistently outperformed it in reducing high-level interlocks and inactive or blocked silo traversals. This led to a more stable and consistently lower total number of infractions, which is reflected in a substantially improved accumulated reward. Specifically, MaskablePPO accumulated reward indicates superior operational performance under constraint-sensitive conditions.

TABLE I
STATISTICAL SUMMARY OF AGENT PERFORMANCE (1,000 EPISODES PER MODEL)

Metric	Statistic	MaskablePPO	Standard PPO
Invalid Actions	Mean ± SD	**0.00 ± 0.00**	6.78 ± 32.46
	Median	**0.00**	**0.00**
	3rd Quartile	**0.00**	1.00
	Maximum	**0.00**	521.00
Low-Level Infraction	Mean ± SD	83.93 ± 284.90	**60.97 ± 263.24**
	Median	**0.00**	**0.00**
	3rd Quartile	2.00	**0.00**
	Maximum	**2169.00**	2880.00
High-Level Infraction	Mean ± SD	**8.83 ± 27.07**	13.30 ± 39.56
	Median	**0.00**	**0.00**
	3rd Quartile	**4.00**	**4.00**
	Maximum	**320.00**	508.00
Inactive Path Travel	Mean ± SD	**838.85 ± 1149.48**	1245.64 ± 1707.91
	Median	**27.00**	42.00
	3rd Quartile	**1806.25**	2829.25
	Maximum	**5295.00**	7982.00
Total Infractions	Mean ± SD	**931.61 ± 1294.43**	1326.68 ± 1795.72
	Median	**84.50**	90.00
	3rd Quartile	**1919.25**	2908.25
	Maximum	**7401.00**	7996.00
Accumulated Reward	Mean ± SD	**−242.86 ± 387.59**	−316.64 ± 450.79
	Median	**−50.40**	−65.10
	3rd Quartile	**0.00**	−1.00
	Minimum	**−2781.60**	−2979.20

Note: Bold values indicate superior performance in the respective metric.

TABLE II
INITIAL SIMULATION CONDITIONS FOR BOTH SCENARIOS

Parameter	Scenario 1	Scenario 2
Seed	412	317
Active Silos Mask	[1, 1, 1, 0, 1, 1]	[1, 1, 1, 1, 1, 1]
Initial Tripper Car Position	Silo 0	Silo 5
Silo Side Length	7.50 m	4.80 m
Silo Height	7.40 m	7.90 m
Tripper Car Speed	0.34 m/s	0.28 m/s
Initial Fill Levels (%)	[41.19, 70.51, 50.49, 69.94, 67.04, 51.19]	[52.22, 75.52, 70.26, 63.14, 42.92, 48.46]

C. Policy Behavioral Analysis

Having established that the MaskablePPO exhibits superior safety-aware performance, we then analyzed whether the learned policy is beneficial for production under two distinct scenarios, as detailed in Table II. Scenario 1 represents a constrained environment where silo 3 is blocked, while Scenario 2 features a fully operational system.

In Scenario 1, the effect of invalid action masking is evident in Fig. 4, where the agent strictly avoids the inactive silo, silo 3, crossing this region in fewer than 4% of all movements. Instead, the policy focuses on the remaining active silos, prioritizing more isolated locations such as silos 4 and 5, while largely avoiding silo 1 as an explicit destination. This behavior arises because the agent can replenish silo 1 during transit when servicing adjacent silos, exploiting the continuous feeding that occurs along the tripper car trajectory. A notable characteristic of this policy is the high frequency of short dwell times, particularly at 0s and 20s. This reflects the environment dynamics, as the presence of a blocked silo increases the discharge rates of the active ones, accelerating level depletion. Moreover, since the silos have a considerable width, 7.5 m,

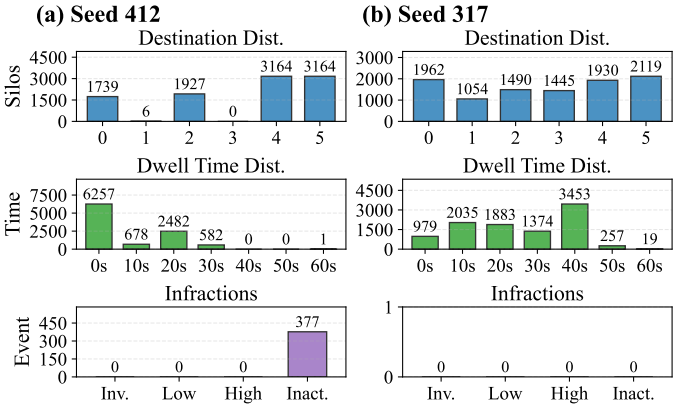

Fig. 4. Results following the trained policy. The y-axis represents counts.

longer travel times are required to reach distant locations, allowing the agent to partially replenish silos during transit and thereby reduce the need for prolonged dwell times. As the agent can replenish only one silo at a time while all active silos continue to drain, it adopts a faster, reactive control cycle to prevent depletion before returning to a given silo.

In Scenario 2, where all silos are active and the silo width is reduced to 4.80 m, the agent exhibits a more balanced destination selection pattern, with a clear preference for the extremities and, to a lesser extent, the central silos. This behavior reflects an optimized in-transit replenishment strategy, where selecting anchor silos at the ends allows intermediate silos to be naturally serviced during transit. The reduced silo width shortens travel times between destinations, enabling longer dwell times, predominantly around 30s and 40s, without compromising level stability. As a result, the agent maintains adequate levels across all silos without incurring infractions, demonstrating a more stable and less reactive control regime compared to scenarios with blocked silos.

Furthermore, the infraction analysis in Fig. 4 reveals an absence of "Low", "High", or "Invalid" level events. This demonstrates that the MaskablePPO agent successfully internalized the physical constraints and discharge dynamics, developing a robust and context-aware control policy.

VI. Conclusion

In this work, we developed and evaluated a RL agent based on the MaskablePPO algorithm for level balancing control in silos through tripper car positioning. The results demonstrate advantages over standard PPO, including the complete elimination of invalid actions, a reduction of approximately 34% in high-level infractions, and a 32% decrease in inactive path travel, indicating safer and more efficient control behavior. These improvements led to an overall reduction of nearly 30% in total infractions, alongside a 23% improvement in accumulated reward. Additionally, MaskablePPO achieved a 9% reduction in training time in the evaluated simulation scenario, indicating improved computational efficiency under the conditions considered. Overall, the proposed approach provides a robust and efficient solution, reinforcing its potential for real-world industrial deployment in safety-critical environments.

References

[1] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual review of fluid mechanics*, 2020.

[2] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does alphago go: From church-turing thesis to alphago thesis and beyond," *IEEE/CAA Journal of Automatica Sinica*, 2016.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[4] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *Annual Review of Control, Robotics, and Autonomous Systems*, 2025.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[6] B. Osiński, A. Jakubowski, P. Ziecina, P. Miłoś, C. Galias, S. Homoceanu, and H. Michalewski, "Simulation-based reinforcement learning for real-world autonomous driving," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 6411–6418.

[7] J. Deng, S. Sierla, J. Sun, and V. Vyatkin, "Offline reinforcement learning for industrial process control: A case study from steel industry," *Information Sciences*, vol. 632, pp. 221–231, 2023.

[8] T. Haider, K. Roscher, F. Schmoeller da Roza, and S. Günnemann, "Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023.

[9] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, 2015.

[10] J. Dai, X. Pan, R. Sun, J. Ji, X. Xu, M. Liu, Y. Wang, and Y. Yang, "Safe rlhf: Safe reinforcement learning from human feedback," 2023.

[11] E. Delavari, F. K. Khanzada, and J. Kwon, "Action space reduction strategies for reinforcement learning in autonomous driving," 2025.

[12] A. Pastor, P. Escofet, S. Ben Rached, E. Alarcón, P. Barlet-Ros, and S. Abadal, "Circuit partitioning for multi-core quantum architectures with deep reinforcement learning," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024.

[13] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *The International FLAIRS Conference Proceedings*, 2022.

[14] R. Stolz, H. Krasowski, J. Thumm, M. Eichelbeck, P. Gassert, and M. Althoff, "Excluding the irrelevant: Focusing reinforcement learning through continuous action masking," *Advances in Neural Information Processing Systems*, vol. 37, pp. 95 067–95 094, 2024.

[15] D. Guo, S. He, and S. Ji, "Intersection decision making for autonomous vehicles based on improved ppo algorithm," *IET Intelligent Transport Systems*, 2024.

[16] A. M. Ali and L. Tirel, "Action masked deep reinforcement learning for controlling industrial assembly lines," in *2023 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2023, pp. 0797–0803.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

[19] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, pp. 1–8, 2021.

[20] P. Itävuo, E. Hulthén, M. Yahyaei, and M. Vilkko, "Mass balance control of crushing circuits," *Minerals Engineering*, vol. 135, pp. 37–47, 2019.

[21] F. N. Caldas and A. X. Martins, "Proposed solutions to the tripper car positioning problem." in *ICEIS (1)*, 2018, pp. 344–352.

[22] K. Albuquerque, H. Silva, E. Teodoro, A. Fonseca, G. Garioli, Ê. Lopes, L. Cota, and T. Euzébio, "Averaging level control of bulk solid material using a tripper car," *IFAC-PapersOnLine*, pp. 147–152, 2019.