Curriculum RL meets Monte Carlo Planning: Optimization of a Real World Container Management Problem

Abhijeet Pendyala (✉) and Tobias Glasmachers

Ruhr-University Bochum, Bochum, Germany
firstname.lastname@ini.rub.de

Abstract. In this work, we augment reinforcement learning with an inference-time collision model to ensure safe and efficient container management in a waste-sorting facility with limited processing capacity. Each container has two optimal emptying volumes that trade off higher throughput against overflow risk. Conventional reinforcement learning (RL) approaches struggle under delayed rewards, sparse critical events, and high-dimensional uncertainty—failing to consistently balance higher-volume empties with the risk of safety-limit violations. To address these challenges, we propose a hybrid method comprising: (1) a *curriculum-learning* pipeline that incrementally trains a PPO agent to handle delayed rewards and class imbalance, and (2) an *offline pairwise collision model* used at inference time to proactively avert collisions with minimal online cost. Experimental results show that our *targeted inference-time collision checks* significantly improve collision avoidance, reduce safety-limit violations, maintain high throughput, and scale effectively across varying container-to-PU ratios. These findings offer actionable guidelines for designing safe and efficient container-management systems in real-world facilities.

Keywords: Reinforcement learning · Curriculum learning · Inference-time planning · Industrial control · Collision avoidance

1 Introduction

Waste-sorting facilities increasingly rely on data-driven methods to meet strict energy efficiency and sustainability goals, due in part to regulatory directives for responsible recycling of packaging waste. Modern plants must deal with fluctuating material types, unpredictable daily volumes, and stringent safety constraints. This work is inspired by the final stage of a waste-sorting facility where n containers (*bunkers*) accumulate various types of material at unique stochastic rates. These materials are subsequently transported to a processing unit (PU) for compaction into bales or products [1] during which PU is unavailable for further

[1] In this study the terms bunker/container and processing unit (PU)/press are used interchangeably.

processing. In addition, each container has a strict maximum capacity and overflowing beyond a threshold requires halting the facility for corrective measures, incurring steep penalties. In this context, *container management* emerges as a critical bottleneck: Emptying these containers too late risks overflow; emptying them too early or too frequently undermines throughput and raises energy costs.

Recent work has modeled container management as a reinforcement learning (RL) problem, notably through *ContainerGym* [9], providing a real-world benchmark where an RL agent decides *when* to empty each container. However, naive Proximal Policy Optimization (PPO) agents frequently fail in this domain, as *delayed rewards*, *sparse critical events*, and a *single shared PU* create complex scheduling dynamics. For instance, some containers have a "higher peak" volume (around 60–75% capacity) for optimal throughput and a "lower peak" (around 30–40%) as a fallback. Emptying containers at the peak volumes yields optimal products, hence emptying at other volumes is undesirable. The prime reason for missing a peak volume is unavailability of the PU caused by a *collision* of two or more containers reaching peak volume around the same time. Prior work showed that *curriculum learning* helps mitigate early overflows and improves PPO's learning curve [10]. Still, the problem of handling collisions remains unsolved, especially at higher container-to-PU ratios.

On the other hand, despite the installation of sophisticated sensors in these facilities and real-time monitoring, most waste sorting design layouts remain fixed once built, even if data points to major bottlenecks. This reveals a broader gap: leveraging RL not only to optimize day-to-day operations but also to guide *facility design decisions* such as how many containers can safely share a PU before collisions dominate. In other industries—like robotics or warehouse logistics—RL insights have influenced layout reconfiguration, helping systems adapt hardware choices to software-derived constraints. Yet in waste-sorting, this feedback loop remains largely unexplored.

To address these gaps, we present a *hybrid RL* method that integrates curriculum learning with a domain-specific *collision model* at inference time. Our approach (1) reduces collision-induced overflows and throughput losses, and (2) closes the loop between software-driven scheduling and hardware design insights. Specifically, we:

- Propose a three-phase *curriculum learning* strategy to tackle delayed rewards and dual-volume targets (higher vs. lower peak).
- Introduce an *offline-trained collision model* for on-the-fly inference checks, overriding risky "no-op" actions when multiple containers approach critical volumes.
- Systematically evaluate varying *container-to-PU ratios* (7:1 to 12:1), providing actionable guidelines on how many containers a single PU can realistically handle without causing excessive collisions.

Empirical results demonstrate that the proposed hybrid RL framework significantly cuts collisions, reduces safety limit violations, maintains higher throughput, and yields design insights for scaling container management.

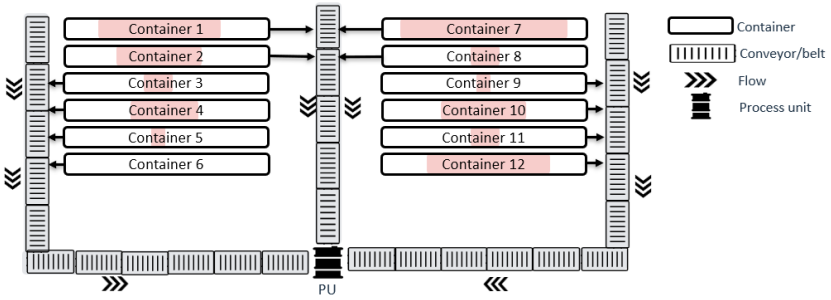


Fig. 1: Layout sketch of a facility with 12 containers and a PU, connected with conveyor belts. The containers are filled from above, with their current fill states indicated by the shaded areas.

2 Related Work

Reinforcement learning has proliferated in industrial and logistics applications, moving beyond canonical benchmarks to real-world applications. We categorize related literature under the following three key themes:

Curriculum Learning in Industrial RL Curriculum learning (CL) systematically organizes the training tasks, typically starting with simpler sub-problems before moving to more challenging ones [1,2]. In industrial domains, where critical actions are rare and reward signals can be delayed or sparse, CL helps agents gather meaningful experience without being overwhelmed by complex dynamics from the outset. Prior work in container management [10] showed that a curriculum-based PPO significantly reduces early overflows compared to naive baselines. We build on this by designing a multi-phase reward curriculum—enabling the agent to master *dual-peak* emptying targets and maintain stable performance under varying inflows.

Inference-Time Planning and Collision Avoidance While a trained RL policy provides baseline decisions at deployment, *inference-time planning* augments these decisions with lookahead logic or heuristic checks, often through Monte Carlo methods. Techniques vary in whether they update the agent's parameters or remain "stateless." Prominent examples include Monte Carlo simulations in game-playing AI [8,3], but similar ideas have surfaced in robotics [4,5], energy systems control [7] and autonomous driving [6]. In our context, we adopt a *collision model* that is trained offline to predict when multiple containers are poised to exceed safe volumes simultaneously. This model supplements a curriculum-trained policy by overriding risky no-operation actions—preventing multi-container collisions. By decoupling real-time safety checks from the offline-trained policy, our method maintains policy stability and adds minimal inference

overhead, which is essential for deployment in high-throughput industrial environments.

RL-Driven Design insights An emerging trend in industrial settings is the use of RL not merely to control a dynamic process but also to inform insights into *design* decisions. In large warehouse environments, multi-agent RL has optimized the assignment of "chutes" to destinations [12], reducing robot congestion and improving throughput. Similar approaches in factory automation focus on workstation placement [13], where RL-driven layout designs outperform handcrafted alternatives. Meanwhile, open-source platforms like "Storehouse" show that RL can outperform heuristic policies in dynamic warehouse slotting [14]. These successes underscore how AI insights can *redesign* processes for higher efficiency, paralleling our goal of using RL not just to operate container management but to shape decisions on how many containers a PU can handle effectively.

Context of our Contribution By integrating a *predefined curriculum* for delayed rewards with an *offline collision predictor*, our hybrid **PPO-CL-CM** approach targets both throughput optimization and collision avoidance in container management. Offline pairwise simulations yield a lightweight collision classifier, which can be queried at each time step to override the policy when collision risk is high. The result is a system that meets key challenges—dual-peak scheduling, sparse rewards, and collision hazards—while also generating real-world design insights. In the following sections, we present the formal environment setup, detail our curriculum learning phases, and then describe how we integrate collision checks at inference.

3 Environment and RL formulation

In this section, we provide details of the considered container management environment. Building on the scenario described in Section 1, we reiterate the central design optimization criterion; each container has two preferred or "ideal" volumes at which emptying yields the highest-quality output. These correspond to a *Higher peak* offering better overall throughput if the container can safely reach this point and a *Lower peak*, providing a reasonable fallback when waiting longer could risk overflow or collide with the PU's availability window. In practice, larger volumes generally improve efficiency, as the PU operates more effectively when processing bigger batches at once. However, strictly aiming for the higher peak can provoke collisions or safety limit breaches if the single PU is not available in time.

Markov Decision Process (MDP) Setup. We model the container-management scenario as an MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$:

- **State s_t:** Comprises volumes $\{v_{i,t}\}_{i=1}^{n}$ for each container, a PU-availability counter p_t (time until the processing unit becomes available), and auxiliary signals such as ideal volumes. This provides the agent with both physical constraints (capacity, current usage) and strategic cues (optimal targets).

- **Action** $a_t \in \{0, \ldots, n\}$**:** Either do nothing ($a_t = 0$) or attempt to empty container i. If $p_t > 0$ (the PU is busy), an emptying request fails, and container i continues to fill. This design highlights collision risks when multiple containers approach peak volumes simultaneously.
- **Volume Dynamics:** Each container i grows according to a *random walk with drift*:

$$v_{i,t+1} \;=\; \max\big(0,\; v_{i,t} + \alpha_i + \epsilon_{i,t}\big), \tag{1}$$

  where α_i is the average fill rate, and $\epsilon_{i,t} \sim \mathcal{N}(0, \sigma_i^2)$ captures stochastic fluctuations. Overflow occurs if $v_{i,t} > 40$, triggering a heavy penalty and episode termination.
- **PU Overhead:** Emptying container i with volume v imposes a busy time $g_i(v)$. The busy time consists of a material-dependent part that is proportional to the volume, and a constant offset for conveyor belt transport of the material from the container to the PU. During this period, p_t counts down to zero, after which the PU is free again. Requests made while $p_t > 0$ are effectively lost, emphasizing scheduling constraints.
- **Rewards:** The agent receives reward for emptying containers close to their peak volumes, since that behavior results in high quality output. Rewards are designed so that emptying at the higher peak is preferable. Container overflow is a terminal state with an episode reset. *Invalid* or wasteful empties (e.g., container already empty or PU busy) incur a negative penalty r_{pen}, while the *do-nothing* action yields zero.

Collision state. A *collision* arises when one or more containers simultaneously approach or exceed their higher-peak volumes, but the PU remains busy processing another container. This can rapidly push volumes over physical capacity if not addressed promptly. Although we first train the agent without a collision-specific mechanism, Section 5 discusses how we incorporate a collision model at inference time.

Objective. The agent must *schedule empties* near either the higher or lower volume peak to maximize efficiency, while preventing overflows and avoiding collisions on the shared PU. The tension between delaying emptying for higher-volume payoffs versus frequent empties for safety and reduced collisions creates a challenging RL problem under delayed rewards and a scarce PU bottleneck.

Key Challenges

- **Stochastic inflow & sensor noise:** Each container's fill rate depends on material type, density, and unpredictable external factors (e.g., time of day, seasonal fluctuations). Sensor noise further obscures volume estimates, making it difficult to predict precisely when a container will reach a target volume.
- **Delayed rewards & class imbalance:** Some containers fill slowly, requiring hours of in-simulation time to reach the target volume. Consequently, episodes contain many "do nothing" steps, during which the state changes

steadily but rewards from emptying remain sparse. As emptying actions are rare, reward-bearing moves are infrequent, creating skewed action distributions that challenge many reinforcement learning algorithms.

– **Dual-peak design & Collisions:** Having two ideal emptying peaks per container creates a nuanced trade-off: waiting for the higher peak boosts efficiency but risks collisions and overflow if multiple containers converge while the PU is busy. Alternatively, settling too often for the lower peak increases emptying frequency, raising energy costs and volume deviations. Balancing these opposing factors—alongside scheduling constraints to avoid collisions and safety violations—poses a central challenge.

Algorithm 1: Three-Phase Reward Computation

Input : Phase $ph \in \{1, 2, 3\}$, current volume v_t, action a_t, penalty r_{pen}, peaks $(v_{\text{low}}, v_{\text{high}})$, Gaussian params (h, w) or (h_1, h_2, w_1, w_2)

Output: Immediate reward r_t

if $a_t = 0$ **then** `// No-op`
  $r_t \leftarrow 0$;
else
  if *(*invalid conditions*)* **then** `// Invalid empty`
    $r_t \leftarrow r_{\text{pen}}$;
  else
    if $ph = 1$ **then** `// Phase 1`
      $r_t \leftarrow (h - r_{\text{pen}}) \exp\left(-\frac{(v_t - v_{\text{high}})^2}{2\,w^2}\right) + r_{\text{pen}}$;
    else if $ph = 2$ **then** `// Phase 2`
      $r_t \leftarrow r_{\text{pen}} + \sum_{i \in \{\text{low,high}\}} \left[h_i - r_{\text{pen}}\right] \exp\left(-\frac{(v_t - v_i)^2}{2\,w_i^2}\right)$;
    else `// Phase 3`
      if $|v_t - v_{low}| \leq 1 \ \vee \ |v_t - v_{high}| \leq 1$ **then**
        $r_t \leftarrow 1.0$
      else
        $r_t \leftarrow 0$

4 Methodology

In this section, we detail our complete pipeline for training PPO-based agents to manage containers. We begin with a *naive PPO* baseline, highlighting its struggles with sparse, multimodal rewards and the tendency toward premature empties. To address these issues, we then propose a *curriculum learning* scheme (PPO-CL) that incrementally shapes the agent's reward landscape. This two-stage progression—naive PPO followed by PPO-CL—lays the groundwork for an inference-time *collision model*, which further mitigates overflow risks when multiple containers compete for the PU.

4.1 Naive PPO Baseline and Its Shortcomings

A straightforward approach is to train a PPO agent directly on the final (multimodal) reward that encourages emptying containers at either the higher or lower peak. However, as outlined in Section 3, this environment poses multiple challenges: rewards are delayed and infrequent, containers fill at varying rates, and collisions can arise when the PU services multiple containers simultaneously. Without additional structure or foresight, empirical results (see table 2) show that a naive PPO agent tends to:

- **Ignore long-term returns.** Driven by sparse rewards, the agent often performs multiple partial empties rather than waiting for the larger peak. This short-sighted strategy blocks the PU more frequently, increasing energy usage and leaving less capacity for containers that are about to overflow.
- **Struggle with skewed action distributions.** With many "do-nothing" steps before a container reaches its target volume, the agent fails to learn precise timing to consistently hit higher-volume empties.
- **Overlook future collisions.** Having no explicit mechanism to anticipate bottlenecks on the PU, the agent may wait too long and face simultaneous arrivals at near-peak volumes, risking overflow or forced early empties.

4.2 Curriculum Learning with PPO

These shortcomings motivate a more structured approach to handle delayed rewards, skewed actions, and collision risks. We therefore present a *curriculum learning* strategy that gradually refines the agent's timing and decision-making. We train a PPO agent with a *three-phase reward curriculum*, gradually introducing complexity over successive training segments. In each phase, the agent follows the same *state* and *action* definitions, but the reward function evolves to guide the agent toward better timing of empties:

- **Phase 1 (Unimodal Reward):** We place a single Gaussian peak at the higher peak volume and no reward at the lower peak, helping the agent learn to avoid excessively early empties.
- **Phase 2 (Multimodal Reward):** Two Gaussian peaks (higher and lower), letting the agent discover a fallback if waiting for the higher peak is unsafe or if the PU is unavailable.
- **Phase 3 (Step Reward):** A strict scheme awarding positive reward only when the emptied volume is within a narrow window (± 1) around *either* peak, refining precision once the agent learned to handle both targets.

As in [10,11], we further stabilize training by *freezing* the policy network in parts of Phase 2, updating only the value estimator to account for changes in reward structure. During Phase 3, we *unfreeze* the policy network but apply a stricter KL-divergence constraint, ensuring the agent does not deviate too aggressively from the policy learned in earlier phases. Algorithm 1 presents the logic for all three phases. By stepping through these phases with carefully tuned budgets, the

agent (*PPO-CL*) acquires more robust scheduling behaviors than a naive single-stage approach. In particular, it learns to occasionally pick the lower peak to avert overflow. However, PPO-CL alone remains largely myopic about *collisions* across containers, motivating our inference-time mitigation strategy in the next section.

5 Collision Model and Inference Pipeline

Although PPO-CL improves emptying behavior, it still shows no signs of successfully learning about collision risks. We mitigate the problem by designing a mechanism to handle situations where multiple containers simultaneously approach their peak volumes. We address this by introducing a *collision model (CM)* trained offline on pairwise container data, then integrating it with the PPO-CL agent at inference time to form **PPO-CL-CM**. This approach balances high-volume empties with timely overrides to avert collisions, all at minimal run-time overhead.

Monte Carlo Rollouts for Pairwise Collisions: For efficient inference-time planning, we generate a large offline dataset of pairwise collision scenarios. By simulating isolated or small groups of containers, we capture diverse collision states with minimal run-time cost. We simulate each container pair (i, j) under stochastic filling (1). Two million repetitions across a container configuration yield a comprehensive offline data set of near-capacity, overflow, and collision events, minimizing deployment computation. For each pair, we perform:

1. Random initialization: Sample means and standard deviations $\mu_i, \mu_j, \sigma_i, \sigma_j$ for filling rates.
2. Stochastic evolution: Evolve volumes $v_i(\tau), v_j(\tau)$ over timesteps τ.
3. Collision bookkeeping: Record collisions when both containers near peaks while the PU is busy.

Feature Extraction and Training At each simulation timestep, we extract collision-predictive features:

- Volumes: Container states $\big(v_i, v_j\big)$
- Proximity to peaks: $\Delta v_i = p_i - v_i, \Delta v_j = p_j - v_j$
- Filling parameters: $(\mu_i, \sigma_i, \mu_j, \sigma_j)$
- Time-lag context: Optional short-volume histories $\{v_i(\tau - 1), v_j(\tau - 1)\}$

Each timestep receives a *collision label* $\{0, 1\}$, creating a supervised dataset $\{\text{features}, \text{collision label}\}$ for training. From this dataset, we train an XGBoost classifier to estimate collision probabilities: $P\big(C_{i,j} \mid s_t, a_t\big) = f_{\text{col}}\big(\text{features}_{i,j}\big)$. XGBoost's gradient-boosted trees efficiently model complex relationships between volumes, fill rates, and near-capacity states. Tuned XGBoost offers fast, accurate pairwise collision risk predictions.

Inference-Time Integration with PPO-CL: Once trained, the collision model f_{col} is invoked at each decision step to assess pairwise collision probabilities. The *baseline* PPO-CL agent first proposes an action a_t. If it decides to empty a specific container ($a_t \neq 0$), we accept that choice directly. However, if PPO-CL proposes *no operation* ($a_t = 0$), the system queries f_{col} for all container pairs to estimate near-future collision risks. If any container at high volume is flagged with a collision probability above a threshold θ, we *override* the no-op by forcing an empty action on the most at-risk container. Algorithm 2 details the procedure.

Action Override Rationale. By limiting overrides to the no-op action, we minimally disturb PPO-CL's learned preference for waiting until containers reach higher volumes. Only when the model detects a high probability of overflow or severe collisions do we *force* an empty on a likely-to-clash container. This blend of *offline collision modeling* and *selective inference-time planning* yields a more collision-aware agent.

Algorithm 2: Integrated Inference-Time Decision with Pairwise Collision Prediction

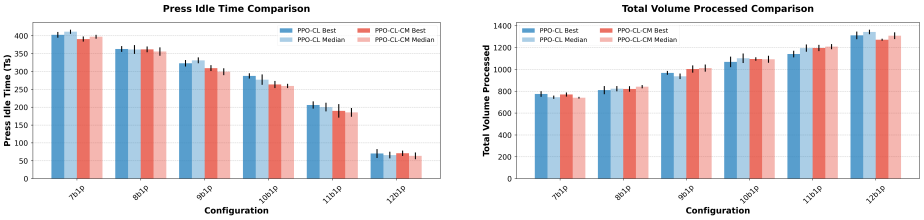
Input : State s_t, PPO-CL policy π_{CL}, collision model f_{col}, threshold θ, peak volumes $\{p_i\}$
Output: Final action $a_{\text{final}} \in \{0, 1, \ldots, n\}$

1. PPO-CL Action: $a_t \sim \pi_{\text{CL}}(a_t \mid s_t)$;
if $a_t \neq 0$ **then**
  return $a_{\text{final}} \leftarrow a_t$; `// Accept non-zero action`

2. Collision Assessment:
foreach *pair* (i, j) **do**
  $P(C_{i,j}) \leftarrow f_{\text{col}}(\text{features}_{i,j})$;
Assemble matrix $P_t[i, j] = P(C_{i,j})$;

3. Check Potential Collision Overrides:
$\mathcal{C} \leftarrow \{\, i \mid v_i(t) \geq p_i - \delta \}$;
if $\mathcal{C} \neq \emptyset$ **then**
  foreach $i \in \mathcal{C}$ **do**
    $\text{CollisionRisk}(i) \leftarrow \textsc{RiskScore}(i, P_t)$
  $i^\star \leftarrow \arg\max_{i \in \mathcal{C}} [\text{CollisionRisk}(i)]$;
  if *CollisionRisk*$(i^\star) \geq \theta$ **then**
    $a_{\text{final}} \leftarrow i^\star$; `// Override with container` $i^\star$
  else
    $a_{\text{final}} \leftarrow 0$; `// Retain no-op`
  return a_{final}

return $a_{\text{final}} \leftarrow 0$; `// No containers at risk, do nothing`



(a) Press idle time comparison. (b) Total volume processed comparison.

Fig. 2: Performance metrics comparison between PPO-CL and PPO-CL-CM methods across different container configurations (7b1p to 12b1p). The bars show mean values and error bars indicate standard deviation. Left: Press idle time shows the duration the press remains inactive. Right: Total volume processed indicates the amount of material handled during one inference episode of 600 timesteps.

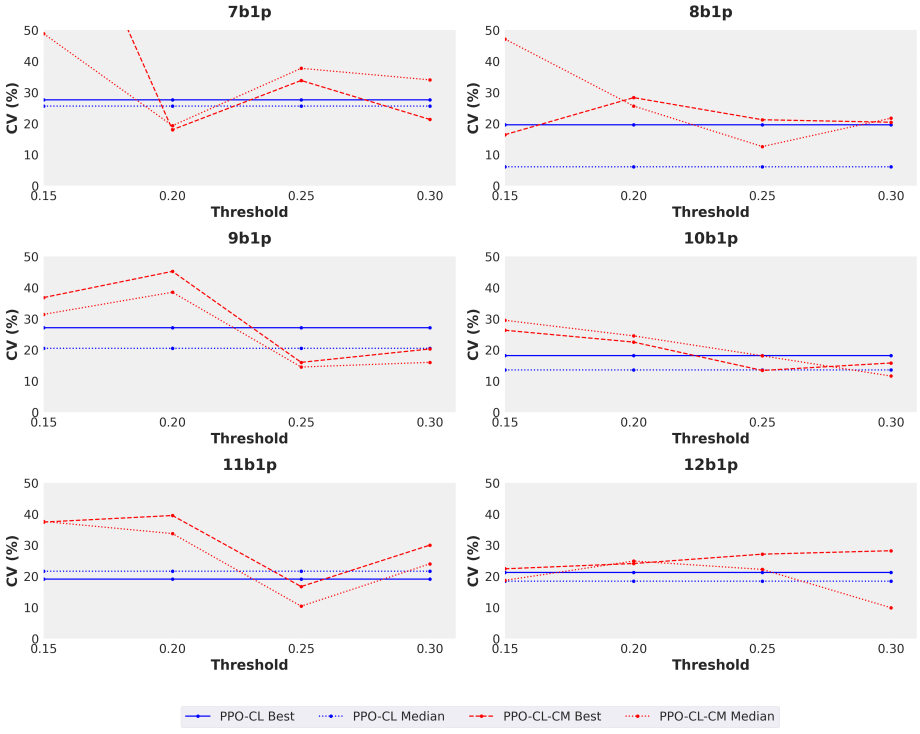


Fig. 3: Comparison of Coefficient of Variation (CV%) across different collision probability thresholds for all container configurations. Each subplot shows the performance of PPO-CL and PPO-CL-CM methods for a specific configuration. Lower CV% indicates more consistent performance.

6 Experimental Evaluation

In this section, we present a quantitative evaluation of the three agents naive PPO, PPO-CL, and PPO-CL-CM across multiple container-to-PU configurations. Our analysis addresses the following research questions (**RQs**):

1. **RQ1:** Is the inference-time collision model (*PPO-CL-CM*) effective compared to PPO-CL and naive PPO in terms of achieved reward?
2. **RQ2:** How effectively does the collision model reduce safety-limit violations (i.e., empties above the critical volume limit)?
3. **RQ3:** How do these findings inform real-world design choices, such as the ideal ratio of containers to processing units?

In the spirit of open and reproducible research, we make our source code available via an ***anonymous repository***.[2] The repository contains a script for reproducing all results presented in this section.

6.1 Experimental Setup

We evaluate our methods on environments with *7 to 12* containers and a single PU (labeled "7b1p" through "12b1p"). Following [9] we use an episode length of 600 timesteps, with a 60-second granularity. For each agent, we conduct *15* independent training runs using distinct random seeds. During inference, we run each seed-based policy in *5* rollouts and collect statistics. We present results for both the *best* and *median* performing seeds of each method, ensuring a fair and comprehensive comparison. Specifically, the *best seed* is selected based on the smallest number of *collision timesteps*, while the *median seed* is chosen from the remaining seeds in terms of the same metric.

Metrics: The reward signal aggregates many different subgoals. To obtain a fine-grained picture of algorithm performance, we monitor the following performance metrics:

- *Press Idle Time* (Fig. 2a): total timesteps during which the PU is not processing any container.
- *Total Volume Processed* (Fig. 2b): material throughput in one 600-step inference episode.
- *Collisions in time-steps*: the total number of timesteps in which a *collision state* occurs; that is, at least two containers simultaneously approach or exceed their ideal volumes (especially the higher peak) while the PU is busy and thus unable to service them.
- *Coefficient of Variation (CV%) (Fig. 3)*: a normalized measure of variability in performance under different collision thresholds.
- *Total Volume Deviation*: the average (over all containers and timesteps) of the absolute difference between a container's actual volume and its nearest ideal peak, gauging how well empties align with target volumes.

[2] https://gitlab.com/anonymousppocl_cm1/anonymous_collisions_paper

- *Actions per Container, Reward per Action, and Peak-Usage Ratios*: drawn from Tables 1 and 2.
- *Safety-Limit Violation Percentage* (Fig. 4): the fraction of emptying actions during inference where a container volume exceeds a fixed ***critical volume limit***, set 5 units above that container's higher ideal emptying peak.

All agents are tested under the same environmental conditions to isolate the impact of curriculum training and collision modeling. Moreover, the results shown in Figures 2 and 4 and in Tables 1 and 2 are reported using the threshold values that yield the lowest CV% in Figure 3, reflecting the most collision-stable configurations in our analysis.

6.2 Impact of Collision Model (RQ1)

From Table 2, we note that naive PPO often fails to empty containers at the *higher* ideal peak altogether (*e.g.*, ratio close to zero), indicating it resorts to early or sub-optimal empties. This behavior leads to more frequent episodes ending prematurely due to overflow (especially in the larger "12b1p" setup) or consistently high volume deviations. While PPO-CL capitalizes on the multi-phase reward shaping to handle delayed feedback and class imbalance, collisions can still occur if multiple containers simultaneously converge on their higher peaks. This is where incorporating *inference-time collision checks* to yield PPO-CL-CM has an edge.

As shown in Fig. 3, PPO-CL-CM achieves lower variability (CV%) across different collision probability thresholds, meaning it more consistently avoids hazardous states. From Fig. 2a, we see PPO-CL-CM generally reduces press idle time compared to PPO-CL, indicating fewer deadlocks where containers are left unemptied until near-overflow conditions. Meanwhile, Fig. 2b shows that *total volume processed* remains at least on par with (and often surpasses) PPO-CL, demonstrating that collision avoidance does not compromise overall throughput in an inference episode.

Table 1 reveals that *collision timesteps* drop systematically for *PPO-CL-CM*, and its *volume deviation* is also slightly lower on average. The *higher/lower peak ratio* in Table 2 confirms that PPO-CL-CM empties containers earlier (lower peak) only when the collision model flags imminent risk, thereby balancing high-throughput empties with safety. Hence, **RQ1** is answered: adding an inference-time collision model mitigates bottlenecks and collisions beyond what curriculum-based RL can achieve alone.

6.3 Safety-Limit Violations (RQ2)

Figure 4 summarizes the frequency of empties that exceed the safety critical volume limit, thus posing a higher risk of overflow and breach of physical limit. We observe that **PPO-CL-CM** consistently maintains a smaller fraction of risky empties overall than PPO-CL for all bunker-to-PU setups from *7b1p* through *12b1p*. This indicates that the collision model not only reduces direct collisions

but also prompts timely empties before containers venture into risky volume ranges. Hence, we conclude **RQ2** by confirming that inference-time collision checks can effectively mitigate dangerous critical empties, supporting safer operation without sacrificing throughput.

6.4 Real-World Implications and Design Guidance (RQ3)

Fig. 2a illustrates that as we move from "7b1p" up to "12b1p," *press idle time* diminishes significantly, reflecting how the PU becomes fully utilized with rising container counts. On the other hand, in extremely large configurations (e.g., 12 containers to 1 PU), collisions inevitably remain because a single resource cannot realistically handle multiple near-peak arrivals at once. While PPO-CL-CM can curb severe collisions, it cannot eliminate them entirely when the resource ratio is unfavorable. Thus for *Moderate Ratios:* Up to around 7–11 containers per PU, collision avoidance measures like PPO-CL-CM yield strong improvements without saturating the system. But for *High Ratios:*, beyond a certain limit (e.g., 12b1p), press idle time becomes negligible, and collision events dominate. An additional PU may be necessary for higher container configurations. Addressing **RQ3**, in practice, operators can apply these findings when deciding how many containers can be feasibly connected to a single processing unit, and whether advanced collision checks are cost-effective. Importantly, applying a collision model allows to safely operate more containers with the same number of expensive PUs.

Config	Agent	Collisions (Ts)	Tot. vol. dev (%)	Reward/action
7b1p	PPO-CL	72.4±20.0 / 81.6±20.9	7.3±2.1 / 6.9±1.2	0.4 / 0.4
	PPO-CL-CM	**22.0±3.9** / **34.4±6.7**	**5.1±0.7** / **6.1±0.6**	**0.7** / **0.8**
8b1p	PPO-CL	77.2±15.1 / 98.6±6.1	7.4±0.8 / 7.3±0.9	0.3 / 0.4
	PPO-CL-CM	**66.2±14.0** / **80.6±10.2**	**6.7±1.2** / **7.0±0.7**	**0.6** / **0.5**
9b1p	PPO-CL	117.0±31.7 / 153.4±31.4	6.5±1.4 / 7.5±1.1	0.4 / 0.3
	PPO-CL-CM	**89.8±14.4** / **117.0±17.0**	**6.3±1.4** / 7.6±0.7	**0.6** / **0.4**
10b1p	PPO-CL	154.6±28.1 / 189.8±25.8	6.8±1.2 / 8.7±0.9	0.4 / 0.2
	PPO-CL-CM	**138.2±18.5** / **145.4±26.3**	**6.4±0.4** / **6.8±0.9**	**0.5** / **0.5**
11b1p	PPO-CL	130.8±25.0 / 156.4±33.8	8.0±1.3 / 6.9±0.5	0.3 / 0.3
	PPO-CL-CM	**83.2±13.9** / **148.4±15.4**	**7.6±1.2** / **6.8±0.7**	**0.5** / **0.5**
12b1p	PPO-CL	126.4±26.8 / 195.8±36.0	11.8±2.1 / 10.2±1.3	0.3 / 0.3
	PPO-CL-CM	**117.4±31.9** / **175.6±38.9**	13.0±2.2 / 11.2±1.6	**0.5** / **0.4**

Table 1: Main performance metrics (Best/Median) for each bunker configuration, all rounded to one decimal place. Each cell shows *Best* ± std. / *Median* ± std. in one line. We boldface the **better** result in PPO-CL-CM whenever it outperforms PPO-CL (lower collisions/dev or higher reward).

Config	Type	Higher/Lower Peak % Ratio			Actions/Container		
		Naive PPO	**PPO-CL**	**PPO-CL-CM**	**Naive PPO**	**PPO-CL**	**PPO-CL-CM**
7b1p	Best	0.0	45.7	1.0	32.7±8.9	20.0±3.4	25.6±2.6
	Median	0.0	18.6	0.8	29.6±12.3	19.6±2.8	24.4±3.1
8b1p	Best	0.0	8.4	2.2	31.0±10.0	18.9±3.8	20.8±3.9
	Median	0.0	14.0	2.9	31.2±8.6	18.8±3.1	20.9±4.7
9b1p	Best	0.0	16.5	4.5	33.3±9.0	19.4±3.4	21.2±2.7
	Median	0.0	19.6	3.3	34.8±9.9	18.3±4.1	21.4±3.1
10b1p	Best	0.0	30.5	3.3	33.2±10.3	18.9±2.5	20.8±4.2
	Median	0.0	11.7	3.1	31.6±8.0	19.1±2.8	20.7±5.0
11b1p	Best	0.0	3.1	1.8	34.1±6.3	20.6±4.6	22.5±4.3
	Median	0.0	6.7	2.9	32.9±7.6	20.3±2.7	21.7±3.5
12b1p	Best	N/A	1.9	1.6	N/A	23.1±7.0	23.2±8.1
	Median	N/A	2.6	2.4	N/A	22.1±4.8	22.1±6.1

Table 2: Comparison of Higher/Lower Peak Percentage Ratio and Mean Actions per container (single decimal precision). For each configuration, "Best" / "Median" rows show mean ± standard deviation where applicable.

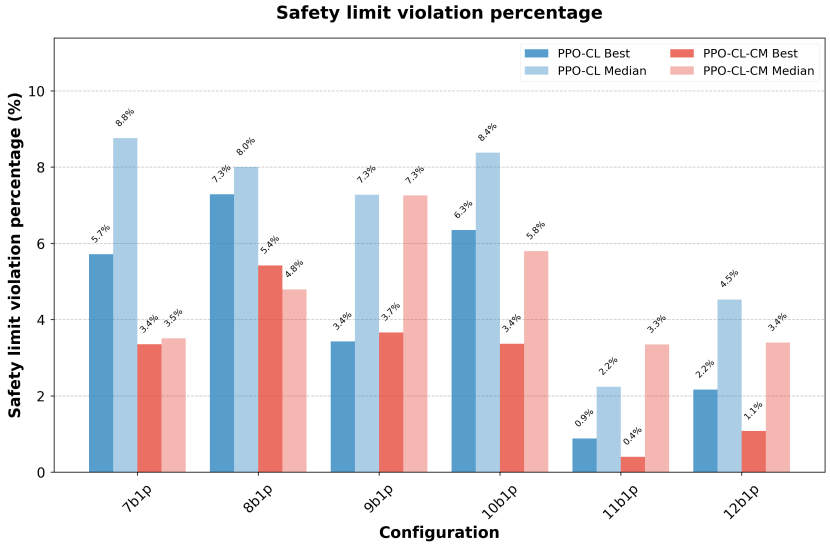


Fig. 4: Comparison of safety limit violation percentages across different bunker configurations. Bars show the percentage of emptying actions that exceeded the safety limit for each bunker configuration using PPO-CL and PPO-CL-CM methods.

7 Conclusion

We have presented a hybrid strategy for container management in a waste-sorting facility in a constrained resource scenario, where each container has two preferred "ideal" emptying volumes. The first component, *PPO-CL*, addresses the challenges of sparse rewards and dual-volume targets through a curriculum-learning approach that teaches the agent to empty containers near either a lower or higher peak. The second component, a *collision model* integrated at inference time (*PPO-CL-CM*), provides targeted overrides when containers risk colliding at peak volumes. This combination effectively balances the key design criteria: prioritizing the higher peak for optimal throughput while resorting to the lower peak only when collisions or safety violations become imminent.

Empirical results across various container-to-PU ratios (7:1 to 12:1) demonstrate that *PPO-CL-CM* reduces both collision episodes and empties above a critical safety threshold, *without sacrificing total processed volume*. By explicitly modeling future collision risk, it prevents premature empties that might otherwise occur if the agent tried to avoid collisions by abandoning the higher peak too soon. From an operational standpoint, these findings suggest that facility managers can confidently scale up container counts to a point, relying on our framework to avoid overflows and to ensure safe, high-volume empties. Beyond that point, collisions become unavoidable, but our method still lessens their severity.

A notable advantage of this framework is that the collision model is trained entirely offline via Monte Carlo simulations, adding only minimal overhead during inference. This approach stands in contrast to computationally intensive online planners like Monte Carlo Tree Search, making it better suited for large, stochastic industrial environments with real-time decision needs. Looking ahead, we envision several avenues to extend this work: integrating multiple PUs (or more complex resource constraints) within the same collision-avoidance framework, dynamically tuning collision thresholds based on time-of-day inflows or real-time capacity data, and more tightly fusing offline collision insights with online RL updates to further refine scheduling decisions. Our findings highlight that a domain-aware collision model, combined with carefully shaped RL curricula, can yield safer, more efficient container management, establishing a robust framework for broader industrial adoption and more complex resource-allocation tasks.

Acknowledgements: This work was funded by the German federal ministry of economic affairs and climate action through the "ecoKI" grant.

References

1. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. Journal of Machine Learning Research 21, 1–50 (2020).

2. Xin Wang, Yudong Chen, and Wenwu Zhu: A Survey on Curriculum Learning, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022.
3. Browne, C. B. and Powley, E. and Whitehouse, D. and Lucas, S. M. and Cowling, P. I. and Rohlfshagen, P. and Tavener, S. and Perez, D. and Samothrakis, S. and Colton, S.: A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games 4(1), 1–43 (2012).
4. Romero, A., Zhang, Y., Gibon, K., Goy, M., Boulet, M., Pineau, J., Labbé, M.: Actor-Critic Model Predictive Control: Differentiable Optimization Meets Reinforcement Learning. arXiv preprint arXiv:2306.09852 (2023).
5. Wang, Z., Wei, W., Xie, A., et al. (2022). Hybrid bipedal locomotion based on reinforcement learning and heuristics. Micromachines, 13(10):1688.
6. Hoel, C.-J., Driggs-Campbell, K., Wolff, K., Laine, L., Kochenderfer, M.J.: Combining Planning and Deep Reinforcement Learning in Tactical Decision Making for Autonomous Driving. IEEE Transactions on Intelligent Vehicles **5**(2), 294–305 (2020).
7. Eseye, A. T., Zhang, X., Knueven, B., et al. (2022). A hybrid reinforcement learning–MPC approach for distribution system critical load restoration. In Proc. IEEE Power and Energy Society General Meeting (PESGM).
8. Silver, D. and Huang, A. and Maddison, C. J. and Guez, A. and Sifre, L. and van den Driessche, G. and Schrittwieser, J. and Antonoglou, I. and Panneershelvam, V. and Lanctot, M. and Dieleman, S. and Grewe, D. and Nham, J. and Kalchbrenner, N. and Sutskever, I. and Lillicrap, T. and Leach, M. and Kavukcuoglu, K. and Graepel, T. and Hassabis, D.: Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529(7587), 484–489 (2016).
9. Pendyala, A., Dettmer, J., Glasmachers, T., Atamna, A.: ContainerGym: A Real-World Reinforcement Learning Benchmark for Resource Allocation. In: Machine Learning, Optimization, and Data Science, pp. 78–92. Springer Nature Switzerland (2024).
10. Pendyala, A., Atamna, A., Glasmachers, T.: Solving a Real-World Optimization Problem Using Proximal Policy Optimization with Curriculum Learning and Reward Engineering. In: Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track, pp. 150–165. Springer Nature Switzerland (2024).
11. Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, Haitao Long, and Quan Yuan, "SCC: an efficient deep reinforcement learning agent mastering the game of StarCraft II," arXiv preprint arXiv:2012.13169, Dec. 2020.
12. Yi Shen, Benjamin McClosky, Joseph W. Durham, and Michael M. Zavlanos, Multi-Agent Reinforcement Learning for Resource Allocation in Large-Scale Robotic Warehouse Sortation Centers, in *2023 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 7137-7143.
13. H. Ikeda, H. Nakagawa, and T. Tsuchiya, Towards Automatic Facility Layout Design Using Reinforcement Learning, in *Communication Papers of the 17th Conference on Computer Science and Intelligence Systems*, vol. 32, Annals of Computer Science and Information Systems, 2022, pp. 11–20
14. J. Cestero, M. Quartulli, A. M. Metelli, and M. Restelli, Storehouse: a Reinforcement Learning Environment for Optimizing Warehouse Management, in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–9