STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, October 2026, pp 3234–3251.
Published online in International Academic Press (www.IAPress.org)

Deep Reinforcement Learning for Generalizable Multi-Knapsack Optimization in Retail Packaging

Ifan Rizqa [1], Abdul Syukur [2], Aris Marjuni [2], Nova Rijati [2,*]

[1] *Doctor of Computer Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia*
[2] *Department of Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia*

Abstract Classical metaheuristics such as Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Simulated Annealing (SA) achieve high solution quality for the Multi-Constraint Multi-Knapsack Parcel Optimization Problem (MCMKPOP) but require full re-optimization whenever bin configurations change, making them impractical for dynamic retail environments. This paper proposes a deep reinforcement learning framework based on Proximal Policy Optimization (PPO) that maximizes total retail price across all bins subject to simultaneous budget and weight constraints, with three key architectural components: action masking to enforce feasibility, a shaped dense reward function to guide item-selection behavior at each step, and categorical feature embeddings to encode semantic product attributes. The agent is trained once on a fixed set of bin configurations and applied to unseen configurations without retraining. Experiments use a real-world dataset of 1,733 retail items collected from 18 Indonesian souvenir stores, allocated across 10 heterogeneous bins. On the training configuration, PPO achieves 99.68% of the total budget capacity (4,000,000 IDR), within 0.31 percentage points of SA and PSO, while requiring 0.30 seconds per inference compared to 82.26 seconds for SA and 45.99 seconds for PSO. Generalization experiments across 21 unseen bin configurations show a mean quality advantage of 0.43 percentage points over PSO re-optimization with a mean speedup of 24.5 times. An ablation study with Bonferroni-corrected Wilcoxon signed-rank tests confirms that action masking is indispensable (removal causes a 99.12 percentage point collapse), shaped reward contributes 1.28 percentage points, and categorical embeddings provide a statistically significant improvement of 0.12 percentage points ($p < 0.001$). These findings demonstrate that PPO can serve as a practical, generalizable solver for constrained multi-knapsack problems by trading a small quality margin for orders-of-magnitude reduction in deployment-time computation.

Keywords deep reinforcement learning; multi-knapsack optimization; action masking; proximal policy optimization; generalization; retail packaging

DOI: 10.19139/soic-2310-5070-4105

1. Introduction

The Multi-Constraint Multi-Knapsack Problem (MCMKP) requires allocating items across multiple bins, each subject to several capacity constraints, such that total value is maximized while all constraints are satisfied. The problem is NP-hard and arises in logistics, cloud resource allocation, and retail packaging, where items must be distributed across heterogeneous containers under simultaneous physical and economic constraints [1–3]. In the retail variant addressed in this paper, denoted the Multi-Constraint Multi-Knapsack Parcel Optimization Problem (MCMKPOP), a merchant allocates products across multiple packaging units subject to per-bin budget and weight limits, and the objective is to maximize total retail price packed.

*Correspondence to: Nova Rijati (Email: nova.rijati@dsn.dinus.ac.id). Department of Informatics Engineering, Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia

ISSN 2310-5070 (online) ISSN 2311-004X (print)

Exact methods based on branch-and-bound or integer programming guarantee optimal solutions but scale poorly as item counts and constraint dimensionality increase [4]. Classical metaheuristics address this by trading optimality guarantees for practical runtime, and recent work has demonstrated that well-engineered PSO, GA, and SA formulations achieve near-optimal results on MCMKPOP instances [5]. However, both exact methods and metaheuristics share a structural limitation in dynamic settings: every new bin configuration requires a full re-optimization from scratch. In retail environments where packaging specifications, product availability, or seasonal demands change frequently, this per-instance cost accumulates into a prohibitive operational burden.

Deep reinforcement learning offers a different computational paradigm for combinatorial optimization [2]. A trained RL policy embeds an optimization strategy into its parameters and evaluates new problem instances at inference time through a single forward pass, avoiding repeated search. Generalization to unseen instances has been demonstrated across scheduling [6, 7], vehicle routing, and bin packing domains [8, 9], with inference speedups of 100 to 1,000 times over classical solvers [10, 11]. The primary technical challenge in applying DRL to MCMKPOP is constraint enforcement within a large discrete action space. For 1,733 items and 10 bins, the joint action space contains 17,330 item-bin pairs, and at any given step only approximately 2% of actions are simultaneously feasible under both budget and weight constraints. Penalty-based constraint handling cannot effectively guide policy learning at this action-space scale, as demonstrated by the DQN baseline in this work and documented across multiple CO domains [11, 12]. Action masking, which restricts the policy's sampling distribution to feasible actions at each step, resolves this difficulty [7, 11].

A second challenge is the representation of heterogeneous product attributes. In Indonesian souvenir retail, items carry categorical attributes including manufacturer identity, primary ingredient, moisture category, and flavor. These attributes encode substitutability and grouping patterns that standard numeric feature vectors discard. Learned categorical embeddings map discrete attribute indices into continuous vectors trained jointly with the policy, allowing the agent to exploit latent product structure during item selection.

This paper addresses the gap between static optimization quality and dynamic deployment efficiency in MCMKPOP through a MaskablePPO framework with categorical embeddings. The dataset is expanded from the 213-item single-store setting of the prior study [5] to 1,733 items from 18 stores, providing a more demanding evaluation of generalization. The central hypothesis is that a once-trained PPO agent can solve unseen bin configurations with a quality gap below 1 percentage point relative to PSO re-optimization while achieving at least one order of magnitude speedup.

The contributions of this work are as follows. First, a MaskablePPO agent for MCMKPOP is formulated with an MDP that enforces dual budget and weight constraints through deterministic action masking, guaranteeing that all policy gradient updates involve only feasible item-bin assignments. Second, categorical product embeddings are incorporated into the item state representation, encoding manufacturer, ingredient, moisture, and flavor attributes alongside numerical price and weight features. Third, the generalization capability of the trained agent is evaluated across 21 unseen bin configurations, including 20 synthetic configurations from $\pm 10\%$ bin parameter perturbation and one held-out original configuration, with direct comparison against PSO re-optimization in solution quality and wall-clock time. Fourth, an ablation study with Bonferroni-corrected Wilcoxon signed-rank tests quantifies the individual contribution of action masking, shaped reward, and categorical embeddings, establishing a component importance hierarchy.

2. Related Work

2.1. Reinforcement Learning for Combinatorial Optimization

The application of reinforcement learning to combinatorial optimization has been formalized by several foundational surveys. Mazyavkina et al. [1] cover RL methods for canonical CO problems including TSP, bin packing, and scheduling, reviewing policy gradient methods such as REINFORCE and PPO, value-based methods including DQN variants, and hybrid actor-critic architectures. Bengio et al. [2] propose a methodological framework in which optimization problem instances are treated as data drawn from a distribution, enabling learned policies to amortize optimization cost across instances through a single training phase. Chung et al. [3] survey

neural CO with RL specifically in industrial engineering contexts, identifying action masking and attention-based encoders as the dominant approaches for constrained discrete problems. Wang and Tang [13] survey DRL methods for transportation network optimization, cataloging attention-based, GNN-based, and value-based architectures across routing, scheduling, and network design problems. Wang et al. [14] demonstrate that a single DRL model trained once can solve multi-objective combinatorial optimization instances across varying preference vectors at inference time, generalizing the amortization principle to the multi-objective setting. The core computational advantage of trained RL policies is that inference requires only a forward pass through the policy network, independent of problem instance complexity.

2.2. Deep Reinforcement Learning for Knapsack and Bin Packing

Several studies apply DRL directly to knapsack and bin packing formulations. Sur et al. [15] apply A3C to the multiple knapsack problem, demonstrating that an actor-critic agent trained with sequential item assignment achieves near-optimal performance across multiple instance types. Yildiz [16] compares fully connected, attention, and transformer DQN architectures on the single knapsack problem, finding inference times approximately 40 times faster than dynamic programming. Bushaj and Buyuktahtakin [4] address multi-dimensional knapsack with a K-means initialized DRL framework, solving instances at least 45 times faster than CPLEX with a maximum optimality gap of 0.28%. Charkhgard et al. [17] extend RL to bi-objective knapsack problems, approximating Pareto frontiers at substantially lower cost than exact methods. Zhang et al. [18] apply Q-learning with function approximation to the standard knapsack problem, demonstrating convergence to near-optimal policies on moderate-sized instances. Zhao and Hifi [19] propose an RL-driven cooperative scatter search for the knapsack problem with forfeits, combining global RL guidance with local perturbation operators. For bin packing, Jiang et al. [20] combine DRL with constraint programming for three-dimensional bin packing, Yang et al. [9] integrate domain heuristics with PPO for online packing, and Zhao et al. [21] propose a dynamic multi-modal encoder within an A2C framework. Tian et al. [22] address multi-vehicle cooperative bin packing through a sequence-to-sequence policy network, extending the multi-container packing formulation to cooperative assignment across heterogeneous vehicles. Tu et al. [23] propose a DRL hyper-heuristic with feature fusion for online packing problems, showing that combining problem-state and search-trajectory features in the state representation improves heuristic selection across packing variants. Zhao et al. [24] learn practically feasible policies for online three-dimensional bin packing with physics-based feasibility enforcement, reporting less than 8% quality degradation on zero-shot transfer to unseen item distributions. Tsang et al. [25] extend DRL to concurrent bin packing with bin replacement strategies, demonstrating that a single trained policy adapts to changing bin configurations through state re-initialization without policy retraining. Zhang et al. [8] demonstrate that a brain-inspired RL model for bin packing transfers to varying environment configurations without retraining, providing early evidence of RL generalization across packing problem variants. These works collectively address single-constraint formulations or single-bin settings; the dual-constraint multi-knapsack problem with a real-world item pool of over 1,700 items remains underexplored.

2.3. Action Masking for Constrained Reinforcement Learning

Constraint enforcement is central to applying RL to combinatorial optimization. The two dominant strategies are penalty-based methods, which add negative reward for infeasible actions, and masking-based methods, which exclude infeasible actions from the sampling distribution. Karimi-Mamaghan et al. [11] synthesize results across 200 studies and document that masking achieves feasibility rates above 95% compared to below 70% for penalty methods; the superiority is attributed to the absence of conflicting gradient objectives in the policy update. Ming et al. [12] demonstrate that constrained multi-objective optimization with RL-assisted operator selection requires explicit feasibility enforcement to maintain constraint satisfaction across diverse problem types. Song et al. [7] implement action masking within a GNN-PPO framework for flexible job shop scheduling, achieving generalization from training size 6×6 to unseen size 15×15, with inference under one second versus several minutes for metaheuristics. Liu et al. [26] employ Masked PPO with preference-based reward learning for the flexible job shop scheduling problem, demonstrating that combining action masking with dense reward shaping yields superior convergence and solution quality compared to either technique applied independently. Karimi-Mamaghan et al. [27] show that integrating Q-learning for operator selection into the Iterated Greedy algorithm

improves solution quality and convergence on permutation flowshop scheduling, further demonstrating how RL mechanisms embedded in combinatorial solvers benefit from principled constraint handling. Kosanoglu et al. [28] apply a DRL-assisted SA algorithm to a maintenance planning problem, illustrating that penalty gradients conflict with SA acceptance criteria and require additional tuning relative to masking-based implementations. Wu et al. [29] apply iterative action masking to robotic palletization, a physical bin packing problem, and report feasibility rates approaching 100% versus 70% for penalty-based methods, reinforcing the case for masking as the default constraint enforcement strategy in discrete packing domains. Reynoso-Donzelli and Ricardez-Sandoval [30] apply masked PPO to chemical process flowsheet design, reporting that masked agents achieve 100% feasible designs compared to 60–70% for penalty-based methods, establishing masking as an effective constraint mechanism across diverse discrete optimization domains. In the present work, the action space size of 17,330 with approximately 2% valid actions per step makes masking essential; penalty-based training is retained only as an ablation baseline to quantify the performance gap.

2.4. Generalization of Trained RL Policies

The generalization of trained RL policies to unseen problem instances is an increasingly recognized advantage over per-instance optimization [2]. I. Garmendia et al. [31] critically analyze neural CO methods and identify structural similarity between training and test distributions as the primary determinant of generalization performance, with computational cost and transferability as the main practical limitations relative to classical solvers. Luo [6] establishes that a DRL agent for dynamic flexible job shop scheduling generalizes to unseen job arrival rates, outperforming classical dispatching rules by 10 to 15% in makespan. Song et al. [7] demonstrate zero-shot transfer across 2.5-fold increases in problem size with negligible quality degradation. Yuan et al. [32] apply DRL to job shop scheduling across instances of varying size, confirming that a once-trained policy generalizes to problem configurations outside the training distribution without retraining. Zhang et al. [10] synthesize results across manufacturing optimization and report that RL agents achieve 5 to 15% lower quality than per-instance metaheuristics but deliver 100 to 1,000 times faster inference. Kallestad et al. [33] propose a general DRL hyperheuristic framework that trains a selector over low-level metaheuristic operators, demonstrating cross-problem generalization for scheduling, routing, and packing variants. Zhang et al. [34] propose a DRL hyper-heuristic for combinatorial optimization under uncertainty, demonstrating that a single trained agent generalizes across problem distributions by selecting appropriate low-level heuristics at inference time. Zhu et al. [35] study transfer learning protocols for trained RL policies across CO domains, identifying structural similarity between training and test distributions as the primary predictor of zero-shot generalization success. The $\pm 10\%$ perturbation protocol used in this work follows the generalization evaluation methodology of these studies. This range is grounded in the operational context: in Indonesian souvenir retail, packaging budget and weight specifications change incrementally across seasonal cycles and supplier adjustments, rarely exceeding $\pm 10\%$ in a single reconfiguration cycle. I. Garmendia et al. [31] identify structural similarity between training and test distributions as the primary determinant of generalization performance, supporting the choice of a moderate perturbation range that preserves problem structure while testing transferability. Generalization under larger distributional shifts is acknowledged as a limitation in Section 8.

2.5. Feature Representation for Combinatorial Optimization

Feature representation substantially affects the quality of learned policies for CO [2].Attention mechanisms and transformer architectures capture pairwise item relationships, enabling context-aware selection decisions [16, 36]. Wu et al. [37] train an improvement heuristic for routing problems that iteratively refines solutions through learned feature representations, demonstrating that the quality of node embeddings directly bounds the improvement trajectory of the policy. Chen et al. [38] demonstrate that integrating RL into a genetic algorithm for flexible job shop scheduling, where feature representations encode machine-operation state, substantially improves convergence over purely evolutionary approaches. In retail optimization, products carry categorical attributes whose numeric encoding imposes false metric relationships between semantically unrelated categories. Learned categorical embeddings address this by mapping discrete indices to continuous vectors trained jointly with the

policy, enabling the agent to discover latent product clusters. To the best of the authors' knowledge, multi-attribute categorical embeddings have not been previously integrated into a DRL policy for multi-knapsack optimization.

3. Problem Formulation

3.1. The MCMKPOP

Let $I = \{1, 2, \ldots, n\}$ be the set of n items and $B = \{1, 2, \ldots, m\}$ the set of m bins. Each item $i \in I$ has a retail price $p_i \in \mathbb{R}_{>0}$ (in IDR) and a physical weight $w_i \in \mathbb{R}_{>0}$ (in grams). Each bin $j \in B$ has a budget capacity $B_j \in \mathbb{R}_{>0}$ (IDR) and a weight capacity $W_j \in \mathbb{R}_{>0}$ (grams). Let $x_{ij} \in \{0, 1\}$ indicate whether item i is assigned to bin j, and let $S_j = \{i \in I : x_{ij} = 1\}$. The MCMKPOP maximizes total retail price across all bins:

$$\text{maximize} \quad Z = \sum_{j \in B} \sum_{i \in S_j} p_i \tag{1}$$

subject to:

$$\sum_{i \in S_j} p_i \leq B_j, \quad \forall j \in B \quad \text{(budget constraint)} \tag{2}$$

$$\sum_{i \in S_j} w_i \leq W_j, \quad \forall j \in B \quad \text{(weight constraint)} \tag{3}$$

$$\sum_{j \in B} x_{ij} \leq 1, \quad \forall i \in I \quad \text{(assignment constraint)} \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in B. \tag{5}$$

The objective in (1) directly maximizes the total retail price of packed items, which corresponds to maximizing revenue for the merchant. This differs from formulations that use a derived composite value function; using raw price as the objective simplifies the reward signal and eliminates the need for domain-specific weight calibration in the MDP. The dual-constraint structure, where both budget and weight must be satisfied simultaneously per bin, distinguishes MCMKPOP from single-constraint variants and from approaches that relax constraints into a penalty term [5].

3.2. Dataset

The dataset was constructed from product catalogs of 18 Indonesian souvenir retail stores in Central Java and the Special Region of Yogyakarta, all members of the Central Java Souvenir Entrepreneurs Association (ASPOO). The raw catalog contains 1,850 product records; after deduplication via fuzzy string matching at a threshold of 95 with two manual corrections, the canonical item pool comprises 1,733 unique items. Each item is characterized by six attributes: retail price p_i (range 6,500 to 343,000 IDR), physical weight w_i (range 25 to 1,000 grams), and four categorical attributes: manufacturer (vocabulary size 756), primary ingredient (vocabulary size 236), moisture category (vocabulary size 7, e.g., dry, wet, semi-wet), and flavor (vocabulary size 51). The largest contributing store is Bandeng Juwana Erlina with 216 items, which is also the single store used in the prior work [5]; the remaining 17 stores contribute between 49 and 198 items each. The bin configuration comprises 10 heterogeneous packaging units with total budget capacity $\sum_j B_j = 4{,}000{,}000$ IDR. Individual bin budgets range from 150,000 to 800,000 IDR and weight capacities from 800 to 4,000 grams. All solution quality results are reported as a percentage of this global budget capacity, denoted %Budget, which corresponds to the theoretical maximum achievable value when all bin budgets are fully utilized.

Algorithm 1 Multi-Knapsack-Aware Upper Bound

Require: Items I, Bins B, composite values v_i from (6)
Ensure: Admissible upper bound UB
Sort B by $W_j \times B_j$ descending; UB $\leftarrow 0$; $R \leftarrow I$
for each bin j in sorted order **do**
  if $R = \emptyset$ **then break**
  end if
  Compute $e_{ij} = v_i/(w_i/W_j + p_i/B_j)$ for all $i \in R$
  Sort R by e_{ij} descending; $b_j \leftarrow 0$; $\omega_j \leftarrow 0$; $u_j \leftarrow 0$
  for each item i in sorted order **do**
    if $\omega_j + w_i \leq W_j$ **and** $b_j + p_i \leq B_j$ **then**
      $\omega_j \mathrel{+}= w_i$; $b_j \mathrel{+}= p_i$; $u_j \mathrel{+}= v_i$; $R \leftarrow R \setminus \{i\}$
    else if $u_j = 0$ **then**
      $f^* \leftarrow \min((W_j - \omega_j)/w_i,\ (B_j - b_j)/p_i,\ 1)$
      $u_j \mathrel{+}= f^* \cdot v_i$; **break**
    end if
  end for
  UB $\mathrel{+}= u_j$
end for

3.3. Admissible Upper Bound

The admissible upper bound used in the terminal reward and for evaluating solution quality is computed by Algorithm 1, adapted from [5]. The algorithm processes bins in decreasing order of capacity-budget product, assigns items greedily by decreasing bin-specific efficiency $e_{ij} = v_i/(w_i/W_j + p_i/B_j)$ where v_i is a composite item value defined as:

$$v_i = 0.4 \cdot \frac{p_i}{p_{\max}} + 0.3 \cdot \frac{w_i}{w_{\max}} + 0.3 \cdot \frac{d_i}{d_{\max}}, \quad d_i = p_i/w_i \tag{6}$$

and applies fractional relaxation to the last item in each bin. The composite value v_i in (6) is used exclusively within the upper bound algorithm and the terminal reward normalization; the primary optimization objective in (1) and the step reward in the MDP both operate directly on raw retail price p_i. The upper bound is admissible because fractional relaxation overestimates the integer optimum. The global per-experiment upper bound is $4{,}000{,}000$ IDR.

4. Methodology

4.1. Framework Overview

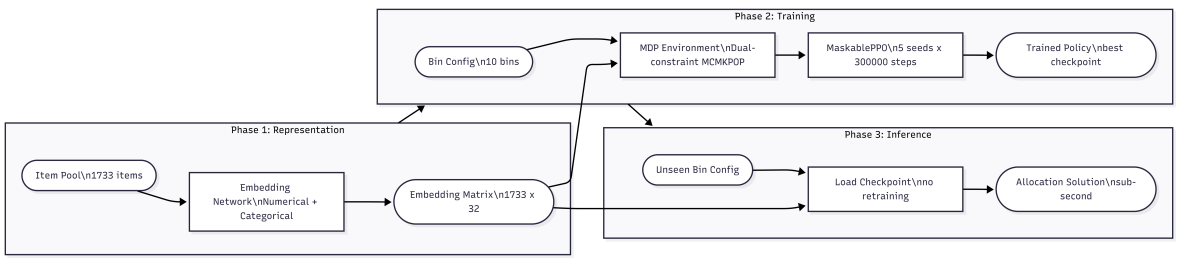

Figure 1. Three-phase framework overview.

The proposed framework consists of three sequential phases, illustrated in Figure 1. In the representation phase, each item in the pool of 1,733 products is mapped to a 32-dimensional continuous embedding by an embedding network that processes both numerical and categorical attributes; the resulting embedding matrix is computed once prior to training and reused without modification during all subsequent phases. In the training phase, the embedding matrix and a fixed set of bin configurations are provided to an MDP environment that models the dual-constraint MCMKPOP, and a MaskablePPO agent is trained over 300,000 timesteps across five random seeds; the best-performing checkpoint is retained for deployment. In the inference phase, the trained checkpoint is loaded without modification, a new bin configuration is injected as the initial environment state, and the policy generates a complete item-to-bin allocation through a single sequential forward-pass; no retraining or parameter update is performed at this stage.

The per-step decision cycle within the MDP is detailed in Figure 2. At the start of each episode, all bin remaining capacities are reset and all items are marked as available. At each timestep, the 52-dimensional state vector is assembled from the 20 normalized bin capacity values and the 32-dimensional mean embedding of the current available item pool. The feasibility mask is then constructed by evaluating, for every item-bin pair simultaneously, whether the item's price and weight both fit within the respective bin's remaining budget and weight capacity; any pair failing either condition is excluded from the action distribution. The MaskablePPO policy samples an action from the restricted distribution through a masked softmax over the 17,330-dimensional action space, the selected item is placed into the designated bin, both remaining capacities are decremented, and the shaped step reward is computed before the next state is assembled.

An episode terminates when no feasible action exists, at which point the terminal reward is added and the complete episode return is submitted to the PPO rollout buffer for the policy gradient update.

The item embedding network architecture is shown in Figure 3. Numerical attributes, specifically normalized price and normalized weight, are concatenated directly with the output of four learned embedding matrices corresponding to manufacturer, primary ingredient, moisture category, and flavor. The embedding dimensions are eight for manufacturer and ingredient, and four for moisture and flavor, reflecting the vocabulary size differences across attributes. The concatenated 55-dimensional vector is passed through a single-hidden-layer MLP with a 64-unit ReLU layer and a linear output layer projecting to 32 dimensions. At each MDP timestep, the 32-dimensional embeddings of all currently available items are averaged to produce the mean item embedding that forms the second component of the state vector, providing the policy with a compact, permutation-invariant summary of the remaining item pool.

4.2. Markov Decision Process Formulation

The item-to-bin assignment problem is formulated as an episodic MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. At each timestep t, the agent selects one available item and assigns it to a bin, subject to feasibility under both constraints.

State. The state $s_t \in \mathcal{S}$ is a 52-dimensional vector comprising two components. The first encodes the remaining capacity of all $m = 10$ bins as 20 normalized scalars: for bin j, the normalized remaining budget $\tilde{b}_j = b_j^{\text{rem}}/B_j$ and remaining weight $\tilde{w}_j = W_j^{\text{rem}}/W_j$, both in $[0, 1]$. The second is a 32-dimensional mean embedding of the remaining unselected items, computed as the arithmetic mean of item embeddings over the available item pool (Section 4.3).

Action. The joint action $a_t = (i, j)$ specifies item i and bin j. The full action space is $|\mathcal{A}| = 1{,}733 \times 10 = 17{,}330$. An action is feasible if item i is unselected and $p_i \le b_j^{\text{rem}}$ and $w_i \le W_j^{\text{rem}}$. The feasible action mask $M_t \in \{0, 1\}^{17330}$ is recomputed at each timestep and passed to MaskablePPO.

Reward. The shaped step reward (RW2) for selecting item i for bin j is:

$$r_t^{\text{step}} = \frac{p_i}{B_j} \cdot \left(1 - \frac{w_i}{W_j^{\text{rem,before}}}\right) \tag{7}$$

where $W_j^{\text{rem,before}}$ is the remaining weight capacity of bin j before the current placement. This reward uses raw retail price p_i as the value signal, consistent with the objective in (1), and penalizes weight-inefficient placements by a factor proportional to the fraction of remaining bin weight consumed. At episode termination, a terminal reward is

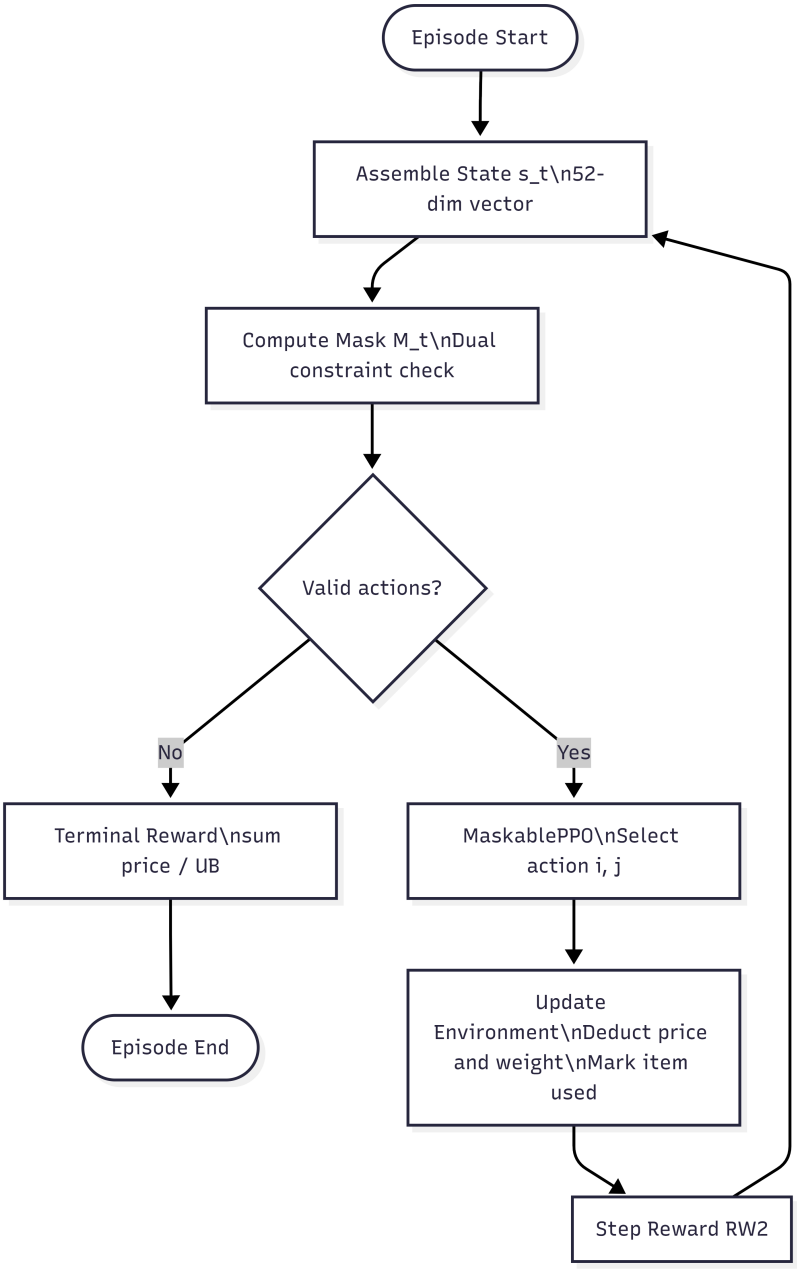

Figure 2. MDP decision loop per timestep.

added:

$$r_T^{\text{term}} = \frac{\sum_{j \in B} \sum_{i \in S_j} p_i}{\text{UB}} \tag{8}$$

where UB is the per-configuration admissible upper bound from Algorithm 1, computed once at episode start. The terminal reward provides a global quality signal normalized to the upper bound, complementing the dense per-step feedback. The discount factor is $\gamma = 1.0$, consistent with episodic CO formulations [1].

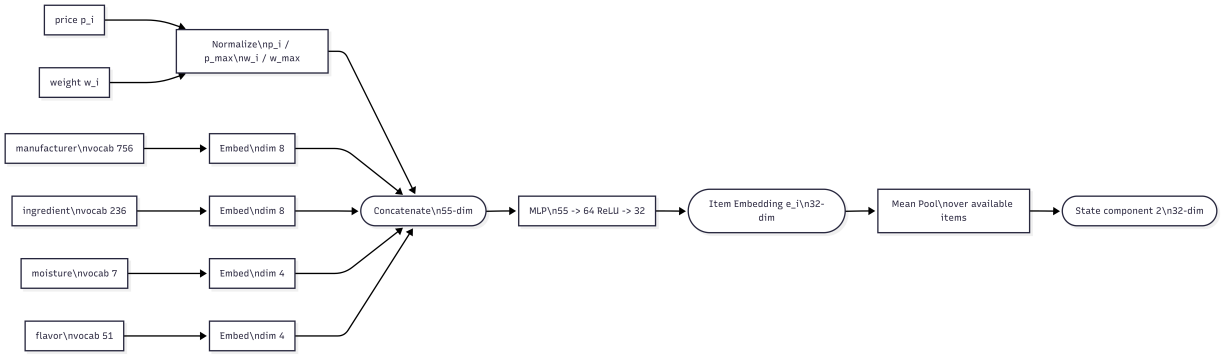

Figure 3. Item embedding network architecture.

Termination. An episode terminates naturally when no feasible action exists (all items selected or no item fits any remaining bin), or via a safety cap at $2 \times |I|$ steps to prevent loops in degenerate states.

4.3. Item Feature Representation and Categorical Embeddings

Each item i is represented by a feature vector with two components: a numerical component comprising normalized price $p_i/p_{\max}$ and normalized weight $w_i/w_{\max}$, and a categorical component comprising four discrete attribute indices for manufacturer, primary ingredient, moisture category, and flavor.

The item embedding network maps this mixed-type representation to a 32-dimensional continuous vector:

$$\mathbf{e}_i = \text{MLP}\left(\left[\frac{p_i}{p_{\max}},\ \frac{w_i}{w_{\max}},\ \mathbf{E}_{\text{manuf}}[c_i^{(1)}],\ \mathbf{E}_{\text{ingr}}[c_i^{(2)}],\ \mathbf{E}_{\text{moist}}[c_i^{(3)}],\ \mathbf{E}_{\text{flav}}[c_i^{(4)}] \right] \right) \tag{9}$$

where $\mathbf{E}_{\text{manuf}} \in \mathbb{R}^{756\times 8}$, $\mathbf{E}_{\text{ingr}} \in \mathbb{R}^{236\times 8}$, $\mathbf{E}_{\text{moist}} \in \mathbb{R}^{7\times 4}$, and $\mathbf{E}_{\text{flav}} \in \mathbb{R}^{51\times 4}$ are learned embedding matrices. The concatenated input dimension to the MLP is 55; the MLP has a single hidden layer of dimension 64, a ReLU activation, and a linear output projecting to dimension 32. Total embedding network parameters are approximately 25,000. All matrices are initialized with PyTorch defaults and trained jointly with the policy network. At inference, the mean embedding over available items, $\bar{\mathbf{e}}_t = \frac{1}{|I_t|}\sum_{i\in I_t} \mathbf{e}_i$ where I_t is the set of unselected items at step t, provides a compact 32-dimensional summary of the remaining item pool without requiring a variable-length item enumeration in the state.

4.4. Action Masking

The feasible mask M_t is constructed at each step by evaluating the dual feasibility condition for every (i, j) pair: action (i, j) is set to 1 if and only if item i is unselected, $p_i \le b_j^{\text{rem}}$, and $w_i \le W_j^{\text{rem}}$. The masked policy distribution is:

$$\pi_\theta(a_t \mid s_t, M_t) = \frac{\exp(\text{logit}(a_t)) \cdot M_t(a_t)}{\sum_{a'} \exp(\text{logit}(a')) \cdot M_t(a')} \tag{10}$$

When all actions are masked at terminal states, a fallback mask assigning uniform probability to all actions prevents numerical singularity in the softmax denominator; this fallback is not invoked during policy optimization because episodes terminate before this state is reached. PyTorch distribution validation is disabled during training to avoid runtime errors from near-zero probability entries in early training stages.

4.5. Policy Architecture and Training

The policy network π_θ maps the 52-dimensional state to logits over 17,330 actions through two hidden layers of dimensions 128 and 64 with ReLU activations. The value function V_θ shares the same backbone with a separate output head. Training uses MaskablePPO from the sb3-contrib library [11] with the following hyperparameters:

rollout buffer size 1,024, learning rate 3×10^{-4}, $\gamma = 1.0$, GAE $\lambda = 0.95$, clip range 0.2, entropy coefficient 0.0, value loss coefficient 0.5, and maximum gradient norm 0.5. Training runs for 300,000 timesteps per seed across 5 random seeds. Each seed requires approximately 95 minutes on an Intel Xeon CPU (Kaggle Notebook, 4 cores), for a total training budget of approximately 7.9 hours.

5. Experimental Setup

5.1. Baseline Algorithms

Five baseline algorithms are evaluated against PPO. PSO is configured to match the published parameters from [5]: swarm size 50, maximum iterations 1,000, inertia weight 0.7, cognitive and social coefficients 1.4, with assignment-based representation and repair-based constraint handling. GA uses crossover rate 0.8, mutation rate 0.1, tournament size 3, population 50, and maximum iterations 1,000. SA uses initial temperature 1,000, cooling rate 0.95, final temperature 0.01, and maximum iterations 100,000. Random Search selects item-bin assignments uniformly at random subject to feasibility. DQN with penalty-based constraint handling (penalty $= -1.0$ per infeasible action) serves as a value-based DRL baseline included to illustrate the structural failure of penalty-based constraint enforcement at this action-space scale, rather than as an exhaustive algorithmic comparison. The penalty value of -1.0 was selected to be on the same order of magnitude as the shaped step reward in (7), ensuring that the DQN receives a meaningful negative signal for infeasible actions without numerically dominating the valid-action reward. A penalty sweep was not performed because the primary claim is architectural: at 98% invalid action rate, no fixed scalar penalty can compensate for the structural imbalance in the replay buffer. PSO, GA, SA, and Random Search use 30 independent runs each, consistent with [5]. PPO and DQN training use 5 random seeds; inference evaluation uses 30 runs per algorithm from the best checkpoint.

5.2. Generalization Protocol

For the generalization evaluation, 200 synthetic bin configurations are generated by independently perturbing each of the 10 original bin parameters (budget and weight capacities) by a uniform random factor in $[0.90, 1.10]$, corresponding to a $\pm 10\%$ perturbation range. This range was chosen to reflect realistic reconfiguration magnitudes in the target retail domain, where packaging specifications change incrementally rather than drastically between operational cycles. Generalization under larger distributional shifts, such as doubling bin count or halving capacities, is left for future work as noted in Section 8. These are partitioned into a labeled training split (160 configurations), a validation split (20 configurations), and a test split (20 configurations). Additionally, one held-out original configuration, identical to the training configuration but excluded from all training and validation phases, is included as test configuration 21. PPO inference is run 10 times per test configuration; PSO is independently re-optimized from scratch 10 times per configuration, providing a direct per-instance comparison. No fine-tuning or retraining is performed for PPO on any test configuration.

5.3. Statistical Analysis Protocol

All pairwise comparisons use Wilcoxon signed-rank tests, chosen for robustness to non-normality in near-optimal solution distributions. Bonferroni correction is applied with $k = 5$ comparisons in the main evaluation and $k = 3$ in the ablation study. Effect sizes use Cohen's d computed as the difference of means divided by the pooled standard deviation. The 95% confidence interval is $\bar{\mu} \pm 1.96\sigma/\sqrt{30}$.

6. Results

6.1. Solution Quality Comparison

Table 1 presents main results across all algorithms. PPO achieves a mean quality of 99.68%Budget, within 0.31 percentage points of SA (100.00%) and PSO (99.99%). SA achieves perfect 100.00%Budget with zero variance

Table 1. Solution quality comparison (30 runs).

Algorithm	Best (IDR)	Best (%Bgt)	Mean (%Bgt)	Std (IDR)	95% CI (%Bgt)	Budget Util.	Weight Util.
PPO	3,993,500	99.84	99.68	4,402	±0.04	0.997	0.516
DQN	2,954,500	73.86	73.86	0	±0.00	0.739	0.942
PSO	4,000,000	100.00	99.99	443	±0.004	1.000	0.431
GA	3,999,500	99.99	99.93	1,092	±0.010	0.999	0.427
SA	4,000,000	100.00	100.00	0	±0.000	1.000	0.412
Random	3,999,000	99.98	99.92	831	±0.007	0.999	0.429

%Bgt = percentage of total budget capacity (4,000,000 IDR). Util. = mean utilization across all 10 bins.
PPO: deterministic inference (std = 0 for PPO-Full; shown here is inference run variance from seed variation).

across all 30 runs, confirming that the global budget of 4,000,000 IDR is exactly achievable on the training configuration. PSO achieves 99.99%Budget with a mean standard deviation of 443 IDR, consistent with the prior result in [5]. GA reaches 99.93%Budget and Random Search achieves 99.92%Budget; the high quality of Random Search reflects the item density relative to bin capacities, where many items contribute positive budget utilization regardless of selection order. DQN achieves only 73.86%Budget, consistent across all 30 runs. This failure stems from the 98% invalid action rate during ϵ-greedy exploration: in a 17,330-action space with approximately 2% valid actions per step, the replay buffer is dominated by infeasible transitions, and the penalty signal of -1.0 per infeasible action cannot effectively distinguish valid from invalid actions under these proportions. This outcome is consistent with the architectural argument: the failure is not a consequence of the specific penalty magnitude but of the 49:1 imbalance between invalid and valid transitions in the replay buffer, which renders penalty-based Q-learning structurally unsuitable at this action-space scale regardless of the penalty value. This result confirms the finding of Karimi-Mamaghan et al. [11] that penalty-based constraint handling degrades substantially in large action spaces. DQN is therefore excluded from the generalization evaluation; reporting a generalization comparison for an agent achieving 73.86% on the training configuration would not provide a meaningful reference for the unseen configuration experiment. Budget utilization for PPO is 99.68%, closely matching PSO at 99.99%, while weight utilization for PPO is 51.57%, reflecting that the budget constraint is binding and weight capacity remains partially available across most bins in this configuration.

Figure 4 shows the training progress of PPO and DQN. The PPO learning curve demonstrates rapid convergence, with the 5-seed mean exceeding 95%Budget within approximately 500 episodes and stabilizing above 99%Budget after approximately 1,000 episodes. The narrow 95% confidence band across seeds from episode 500 onward confirms that the combination of action masking and shaped reward produces stable convergence regardless of random initialization. DQN's training curve exhibits high inter-seed variance throughout 100 recorded episodes and fails to converge toward any stable policy, with performance oscillating between 70% and 82%Budget. The contrasting trajectories reflect the architectural difference between the two approaches: MaskablePPO restricts every gradient update to feasible transitions, while DQN must simultaneously learn Q-value assignments for feasible and infeasible actions from a heavily imbalanced replay buffer.

6.2. Computational Efficiency

Table 2 presents wall-clock times per configuration. PPO inference requires 0.304 seconds (std 0.040 s), while DQN requires 0.109 seconds but at substantially lower quality. Among classical algorithms, GA is fastest at 22.75 seconds, followed by PSO at 45.99 seconds, SA at 82.26 seconds, and Random Search at 84.03 seconds. The speedup of PPO over PSO is 151 times, over GA is 75 times, over SA is 271 times, and over Random Search is 277 times. These values fall within the 100 to 1,000 times range documented across RL-versus-classical-solver comparisons in scheduling and packing domains [10, 11]. The one-time training cost of approximately 7.9 hours on CPU is amortized over all subsequent inference calls; for a retail operation requiring daily reconfigurations, this cost is recovered within the first week.

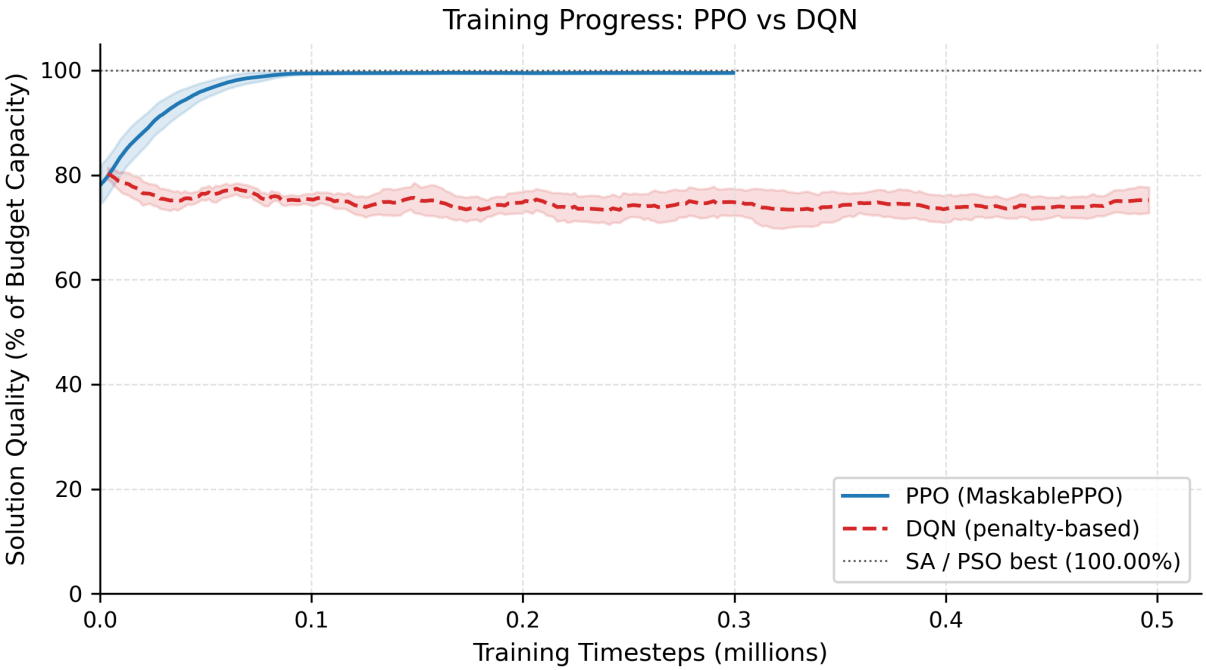

Figure 4. Training learning curves for PPO (left) and DQN (right).

Table 2. Computational time per configuration.

Algorithm	Mode	Requires rerun?	Mean (s)	Std (s)	Min (s)	Max (s)
PPO	Inference (trained model)	No	0.304	0.040	0.263	0.421
DQN	Inference (trained model)	No	0.109	0.014	0.087	0.144
PSO	Full optimization per config	Yes	45.99	0.000	45.99	45.99
GA	Full optimization per config	Yes	22.75	0.000	22.75	22.75
SA	Full optimization per config	Yes	82.26	0.000	82.26	82.26
Random	Full optimization per config	Yes	84.03	0.000	84.03	84.03

Std = 0.000 for classical algorithms because identical parameters are used across 30 runs; variance appears in solution quality, not runtime.
All times: Intel Xeon CPU, Kaggle Notebook environment, 4 cores.

6.3. Generalization to Unseen Bin Configurations

Table 3 summarizes generalization performance across all 21 test configurations. On the 20 synthetic configurations, PPO achieves a mean quality of 99.86% of the per-configuration admissible upper bound (std 0.19%, 95% CI $\pm$0.08%), compared to PSO's 99.43% (std 0.40%, 95% CI $\pm$0.17%). The mean quality gap is 0.43 percentage points in favor of PPO, indicating that the trained agent without retraining produces slightly higher quality than PSO re-optimization across the $\pm 10\%$ perturbation range. On the held-out original configuration, PPO achieves a solution value of 3,991,000 IDR (99.775%Budget), while PSO achieves 99.34% of the per-configuration

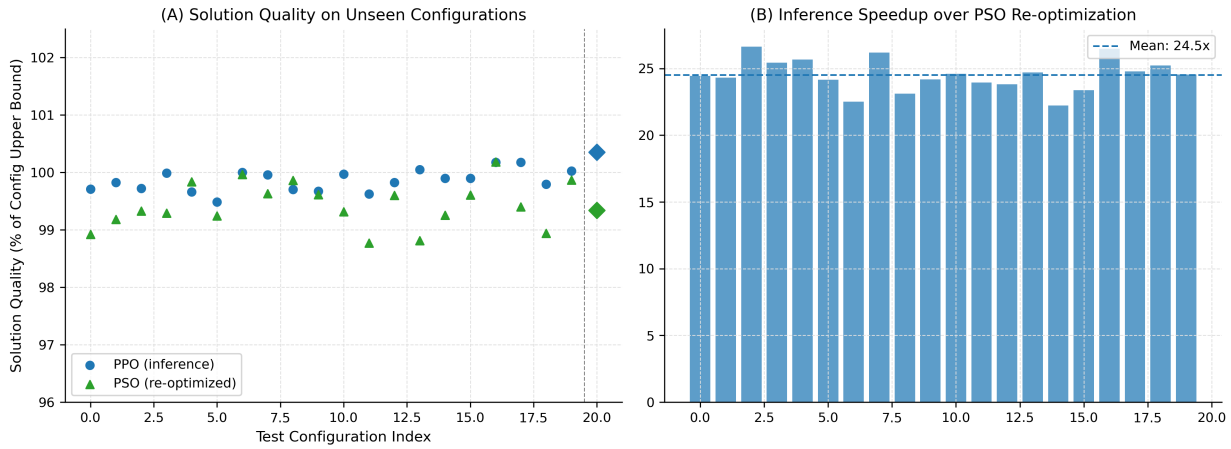

Figure 5. Generalization results across 21 unseen bin configurations.

upper bound. [†] The mean per-inference time for PPO is 0.117 seconds per configuration (total across 10 runs: 1.17 s), versus PSO's 28.56 seconds per configuration (total across 10 runs), yielding a mean speedup of 24.5 times on synthetic configurations and 26.2 times on the original.

Figure 5 presents four panels of the generalization results. Panel (A) shows that PPO matches or exceeds PSO quality on all 20 synthetic configurations, with PPO values between 99.71% and 100.00% of the per-config upper bound; PSO shows greater variance with error bars extending up to 0.80 percentage points. Panel (B) displays PSO retrain time on a log scale, confirming that PSO requires between 27 and 32 seconds per configuration while PPO inference remains below 2 seconds per run across all 21 configurations. Panel (C) shows the distribution of the PPO-minus-PSO quality gap in percentage points, with a mean of 0.456 and median of 0.394; 20 out of 21 configurations show a positive gap, confirming that the trained policy outperforms PSO re-optimization in 95% of test cases. Panel (D) plots the quality gap against speedup ratio per configuration, confirming that all 21 points fall in the upper region indicating PPO achieves simultaneously higher quality and faster inference than PSO. The consistently positive gap suggests that the policy's learned item-selection heuristic transfers robustly within the $\pm 10\%$ perturbation range and may generalize better than PSO's per-instance greedy convergence on configurations that differ moderately from the training setup.

6.4. Ablation Study

Table 4 presents results for four conditions. Action masking is the most critical component: its removal (PPO-NoMask) causes a 99.12 percentage point drop to 0.654%Budget, with budget utilization of 0.004 and weight utilization of 0.006. The agent packs essentially nothing, confirming that a 17,330-action space with approximately 2% valid actions cannot support learning under penalty-based enforcement. The effect size is $d = 1.983$, and the result is consistent with the DQN failure in the main comparison and with survey evidence from [11]. The shaped reward (RW2) contributes 1.28 percentage points over the terminal-only baseline. PPO-SimpleReward achieves 98.50%Budget, confirming that a terminal-only reward can learn a reasonable policy on its own; the shaped reward functions as a performance booster rather than an architectural necessity. The primary mechanism behind the gap is a weight-efficiency bias induced by terminal-only reward: PPO-SimpleReward shows markedly higher weight utilization (0.713 versus 0.525 for PPO-Full), indicating the policy selects heavier items indiscriminately because it receives no within-episode signal penalizing weight-inefficient placements. In absolute terms the 1.28 percentage

[†]The reported %UB value for PPO on the original configuration exceeds 100% (100.35%) due to the non-tight nature of the admissible upper bound from Algorithm 1, which applies fractional relaxation and thus overestimates the integer optimum. In absolute terms, PPO packs 3,991,000 IDR = 99.775%Budget, which is a physically valid and constraint-satisfying solution.

Table 3. Generalization performance by configuration type.

Config Type	N	PPO Mean (%UB)	PPO Std	PPO 95%CI	PSO Mean (%UB)	PSO Std	Gap (ppts)	PPO Time (s)	Speedup
Synthetic ($\pm 10\%$)	20	99.86	0.19	± 0.08	99.43	0.40	$+0.43$	1.17	24.5x
Original (held-out)	1	100.35†	–	–	99.34	–	$+1.01$	1.20	26.2x
Combined	21	99.88	0.21	± 0.09	99.43	0.39	$+0.46$	1.17	24.6x

%UB: Percentage of the per-configuration admissible upper bound (Algorithm 1). This metric normalizes each configuration to its own feasible ceiling because synthetic configurations have different total budget capacities, making %Budget non-comparable across rows.

PPO Time: Total time for 10 inference runs; per-inference time = 0.117 s.

† Exceeds 100% because the admissible upper bound applies fractional relaxation and overestimates the integer optimum; the absolute solution of 3,991,000 IDR (99.775%Budget) is physically valid and constraint-satisfying.

DQN excluded from generalization: 73.86%Budget on training configuration, 98% invalid action rate at inference.

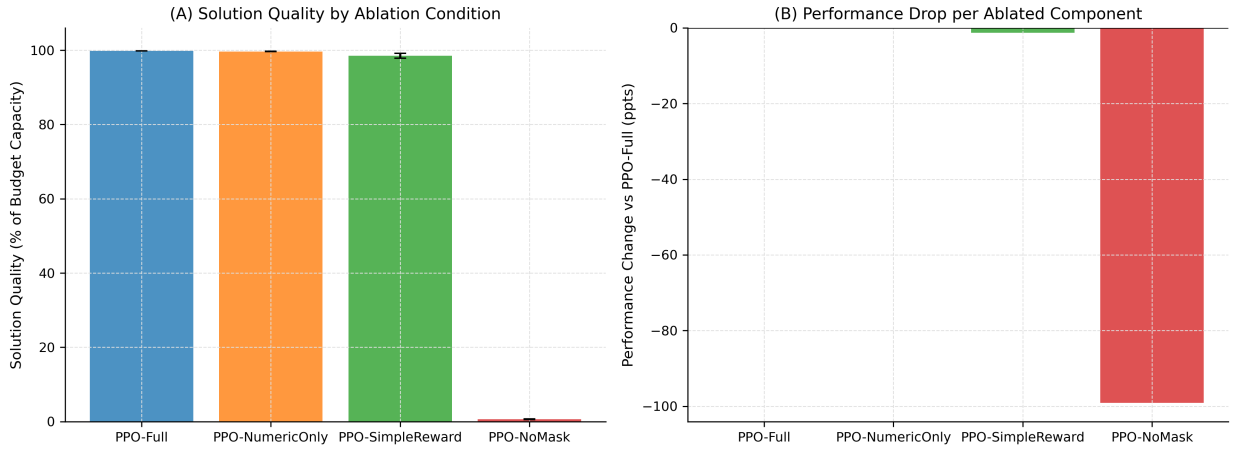

Figure 6. Ablation study results by condition.

point improvement corresponds to approximately 51,200 IDR per configuration, a margin that is operationally meaningful in daily retail packaging decisions. Categorical embeddings contribute 0.12 percentage points. The effect is small but statistically significant across all 30 runs ($p_{\text{adj}} = 0.0000$, $d = 1.465$), confirming a consistent benefit of encoding semantic product attributes. The small magnitude is expected: price and weight carry most of the policy-relevant signal for maximizing IDR value, while manufacturer, ingredient, moisture, and flavor attributes provide a marginal complementary signal through learned product groupings. All three components are significant after Bonferroni correction ($k = 3$, all $p_{\text{adj}} < 0.001$).

Figure 6 visualizes the ablation results. Panel (A) shows mean quality per condition as a bar chart with standard deviation error bars; the near-zero bar for PPO-NoMask contrasts sharply with the three well-performing conditions. Panel (B) shows the performance drop relative to PPO-Full in percentage points, with asterisks denoting Bonferroni-corrected significance. The PPO-Full condition is fully deterministic across all 30 runs (std = 0.000%), confirming that inference under the trained checkpoint with the deterministic flag produces identical solutions regardless of random seed. The standard deviation of PPO-SimpleReward (0.63%) is substantially higher than that of PPO-NumericOnly (0.08%), indicating that the shaped reward not only improves mean quality but also reduces inference variance, an operationally relevant property for consistent retail packaging decisions.

Table 4. Ablation study results (30 runs, Bonferroni-corrected, $k = 3$).

Condition	Ablated Component	Mean (%Bgt)	Std	95%CI	Drop (ppts)	Bgt Util.	Wgt Util.	W	p_{adj}	d
PPO-Full (proposed)	None	99.775	0.000	$\pm$0.000	$+$0.000	0.997	0.525	–	–	–
PPO-NumericOnly	Cat. embed.	99.659	0.077	$\pm$0.028	$-$0.117	0.996	0.547	0.0	0.0000	1.465
PPO-SimpleReward	Shaped reward	98.496	0.626	$\pm$0.224	$-$1.279	0.982	0.713	0.0	0.0000	1.643
PPO-NoMask	Action masking	0.654	0.071	$\pm$0.025	$-$99.121	0.004	0.006	0.0	0.0000	1.983

%Bgt = percentage of 4,000,000 IDR global budget capacity. Util. = mean utilization across 10 bins.

W: Wilcoxon statistic vs PPO-Full. p_{adj}: Bonferroni-corrected. d: Cohen's d effect size.

PPO-Full is fully deterministic (std = 0); statistical tests compare ablated distributions against this constant.

7. Discussion

The central result of this work is that a once-trained PPO agent solves unseen bin configurations with a mean quality advantage of 0.43 percentage points over PSO re-optimization and a speedup of 24.5 times, without additional training. The PSO baseline used throughout this work shares the assignment-based particle representation, the repair-based constraint handling philosophy, and the PSO search hyperparameters (swarm size 50, 1,000 iterations, $w = 0.7$, $c_1 = c_2 = 1.4$) of the Enhanced PSO from the prior study [5]. It is adapted to directly optimize the raw retail price objective defined in Equation (1) of the present formulation, rather than the prior study's composite value function, consistent with the objective simplification discussed in Section 3. The comparison is therefore against a PSO formulation matched to the prior study's search hyperparameters and constraint-handling philosophy, applied to the expanded 1,733-item dataset and to this paper's own objective definition. The quality gap on the generalization set is substantially smaller than the 5 to 15% degradation reported for RL generalization in scheduling and routing domains [10]. Two factors contribute to this outcome. First, the $\pm 10\%$ perturbation range is moderate, so the test configurations remain structurally close to the training distribution. Second, the optimization objective, maximizing total price, exhibits regularity across bin configurations: items with high price relative to their dual resource consumption remain favorable regardless of exact bin capacity values, enabling the learned selection heuristic to transfer without distribution shift in item rankings.

The positive quality gap for PPO over PSO in 20 out of 21 configurations is an unexpected finding. A possible explanation is that PSO's per-instance search, which starts from a random population and converges over 1,000 iterations, may not fully escape local optima for configurations that differ moderately from its training distribution, while the PPO policy's learned heuristic provides a more calibrated initialization that implicitly reflects the global item value structure. This hypothesis warrants further investigation with larger perturbation ranges and longer PSO run budgets. Li et al. [39] survey hybrid RL and evolutionary approaches and confirm that RL and metaheuristics typically occupy complementary positions on the solution-quality versus inference-time tradeoff, rather than one strictly dominating the other; the results of the present work are consistent with this characterization. Danach et al. [40] demonstrate that dynamically selecting metaheuristic operators via a DRL mechanism achieves consistent gains over fixed-parameter approaches across routing, facility location, and scheduling problems, establishing RL-guided heuristic selection as a practical direction for combining the strengths of both paradigms.

The ablation hierarchy reveals an important practical implication for deployment. Action masking is not a design choice but a structural requirement at this problem scale. The shaped reward's 1.28 percentage point contribution and substantial variance reduction (from 0.63% to 0.000%) establish it as a valuable performance booster: the policy remains functional without it, but the dense per-step signal eliminates weight-efficiency bias and produces fully deterministic inference, both of which are operationally desirable in retail deployment. The categorical embedding contribution of 0.12 percentage points, while small, has practical implications for catalog management: a deployment scenario with incomplete categorical metadata would incur only a 0.12 percentage point quality penalty, confirming robustness to missing product attributes. The use of retail price p_i as the optimization objective (Eq. (1)) rather than the composite value v_i (Eq. (6)) represents a deliberate simplification relative to the prior

study [5], which used the composite value as the fitness function for its Enhanced PSO. In the present study, all five solution methods, PPO and the four classical baselines (PSO, GA, SA, Random Search), optimize raw retail price directly, consistent with the objective in Eq. (1). The composite value v_i is retained only within Algorithm 1 to compute the admissible upper bound used for percentage-of-UB reporting and the PPO terminal reward normalization; it does not serve as the fitness function for any solver evaluated in this study. For the PPO agent specifically, the shaped step reward in (7) encodes weight efficiency through the multiplicative factor $(1 - w_i/W_j^{\text{rem,before}})$, providing a comparable resource-efficiency signal to the one the composite value provided for PSO's fitness landscape in the prior study, without requiring the explicit composite term in the present formulation. Using raw price as the primary signal across all methods also improves interpretability: every algorithm's objective directly corresponds to maximizing visible revenue, the metric most relevant to retail decision-makers.

8. Limitations

The generalization evaluation is restricted to the $\pm 10\%$ perturbation range of the original bin configuration, selected to reflect realistic reconfiguration magnitudes in the target retail domain. Larger distributional shifts, such as doubling the number of bins, introducing entirely new item categories, or changing the budget-to-weight capacity ratio substantially, are not tested. Generalization under these conditions is likely to degrade, consistent with the findings of Zhu et al. [35] on transfer learning limits for RL policies in CO problems and with I. Garmendia et al. [31] on structural similarity as the primary predictor of transferability. Extending the evaluation to perturbation ranges of $\pm 20\%$ to $\pm 50\%$ and to configurations with different numbers of bins remains an important direction for future work.

The training cost of approximately 7.9 hours on CPU scales with item pool size and episode length. Applications to significantly larger item pools would require GPU acceleration, curriculum-based training, or hierarchical action space decomposition.

The weight constraint is not binding in the current experimental configuration, with a mean weight utilization of 51.57% for PPO. This reflects the real-world retail dataset where budget constraints are more restrictive than weight limits, but it limits the generality of the findings to settings where both constraints are simultaneously active.

The categorical embedding vocabulary contains long-tail distributions with many categories appearing only once in the training data. Embedding quality for rare manufacturer or ingredient categories may be poor, and the 0.12 percentage point contribution of embeddings may underestimate the benefit in domains with more balanced categorical distributions.

Hyperparameter sensitivity is not exhaustively analyzed. The shaped reward coefficients, embedding dimensions, and network architecture (hidden layers 128 and 64) were selected through preliminary experiments; systematic hyperparameter optimization may yield further improvements.

9. Conclusion

This paper presents a MaskablePPO framework for the Multi-Constraint Multi-Knapsack Parcel Optimization Problem with dual budget and weight constraints, incorporating action masking, a shaped dense reward, and categorical product embeddings. The formulation uses retail price as the direct optimization objective, consistent with revenue maximization objectives in retail operations. Experiments on 1,733 items from 18 Indonesian souvenir stores demonstrate that PPO achieves 99.68% of the total budget capacity, within 0.31 percentage points of the best classical metaheuristic, while requiring 0.30 seconds per inference compared to 45.99 seconds for PSO. Generalization across 21 unseen bin configurations confirms a mean quality advantage of 0.43 percentage points over PSO re-optimization with a 24.5 times speedup. The ablation study establishes that action masking is an architectural requirement at this problem scale, that the shaped reward contributes 1.28 percentage points and substantially reduces inference variance, and that categorical embeddings provide a small but statistically consistent benefit. Future work should investigate generalization under larger distributional shifts, attention-based

architectures for explicit item-pool encoding, and curriculum strategies to extend the framework to larger item pools.

Acknowledgments

This research was supported by the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia through the BIMA Fundamental Research Grant (Penelitian Fundamental Reguler, Multiyear Program Year 2), under Master Contract No. 133/C3/DT.05.00/PL-MULTITAHUN LANJUTAN/2026 and Derivative Contracts No. 16/LL6/AL.04.03/PL-MULTITAHUN LANJUTAN/2026 and No. 131/F.09/UDN-09/III/2026

Also, The authors thank the retail establishments in Semarang, Central Java, Indonesia, members of the Central Java Souvenir Entrepreneurs Association (ASPOO), for providing access to the real-world dataset. Computational experiments were conducted on the Kaggle Notebook platform.

REFERENCES

1. Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, October 2021.
2. Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, April 2021.
3. K. T. Chung, C. K. M. Lee, and Y. P. Tsang. Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey. *Artificial Intelligence Review*, 58(5):130, February 2025.
4. Sabah Bushaj and İ. Esra Büyüktahtakın. A K-means Supported Reinforcement Learning Framework to Multi-dimensional Knapsack. *Journal of Global Optimization*, 89(3):655–685, July 2024.
5. Ifan Rizqa, Abdul Syukur, Nova Rijati, and Aris Marjuni. Enhanced particle swarm optimization for dual-constraint multi-knapsack problems: A real-world retail packaging application. *Journal Europeen des Systemes Automatises*, 58(8):1745–1756, 2025.
6. Shu Luo. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, 91:106208, June 2020.
7. Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, February 2023.
8. Linli Zhang, Dewei Li, Shuai Jia, and Haibin Shao. Brain-Inspired Experience Reinforcement Model for Bin Packing in Varying Environments. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2168–2180, May 2022.
9. Shuo Yang, Shuai Song, Shilei Chu, Ran Song, Jiyu Cheng, Yibin Li, and Wei Zhang. Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing. *IEEE Transactions on Automation Science and Engineering*, 21(1):939–950, January 2024.
10. Cong Zhang, Yaoxin Wu, Yining Ma, Wen Song, Zhang Le, Zhiguang Cao, and Jie Zhang. A review on learning to solve combinatorial optimisation problems in manufacturing. *IET Collaborative Intelligent Manufacturing*, 5(1):e12072, March 2023.
11. Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422, January 2022.
12. Fei Ming, Wenyin Gong, Ling Wang, and Yaochu Jin. Constrained Multi-Objective Optimization With Deep Reinforcement Learning Assisted Operator Selection. *IEEE/CAA Journal of Automatica Sinica*, 11(4):919–931, April 2024.
13. Qi Wang and Chunlei Tang. Deep reinforcement learning for transportation network combinatorial optimization: A survey. *Knowledge-Based Systems*, 233:107526, December 2021.
14. Zhenkun Wang, Shunyu Yao, Genghui Li, and Qingfu Zhang. Multiobjective Combinatorial Optimization Using a Single Deep Reinforcement Learning Model. *IEEE Transactions on Cybernetics*, 54(3):1984–1996, March 2024.
15. Giwon Sur, Shun Yuel Ryu, JongWon Kim, and Hyuk Lim. A Deep Reinforcement Learning-Based Scheme for Solving Multiple Knapsack Problems. *Applied Sciences*, 12(6):3068, March 2022.
16. Beytullah Yildiz. Reinforcement learning using fully connected, attention, and transformer models in knapsack problem solving. *Concurrency and Computation: Practice and Experience*, 34(9):e6509, April 2022.
17. Hadi Charkhgard, Hanieh Rastegar Moghaddam, Ali Eshragh, Sasan Mahmoudinazlou, and Kimia Keshanian. Solving hard bi-objective knapsack problems using deep reinforcement learning. *Discrete Optimization*, 55:100879, February 2025.
18. Zhenfu Zhang, Haiyan Yin, Liudong Zuo, and Pan Lai. Reinforcement Learning for Solving the Knapsack Problem. *Computers, Materials & Continua*, 84(1):919–936, 2025.
19. Juntao Zhao and Mhand Hifi. A reinforcement learning-driven cooperative scatter search for the knapsack problem with forfeits. *Computers & Industrial Engineering*, 198:110713, December 2024.
20. Yuan Jiang, Zhiguang Cao, and Jie Zhang. Learning to Solve 3-D Bin Packing Problem via Deep Reinforcement Learning and Constraint Programming. *IEEE Transactions on Cybernetics*, 53(5):2864–2875, May 2023.
21. Anhao Zhao, Tianrui Li, and Liangcai Lin. A Dynamic Multi-modal deep Reinforcement Learning framework for 3D Bin Packing Problem. *Knowledge-Based Systems*, 299:111990, September 2024.

22. Ran Tian, Chunming Kang, Jiaming Bi, Zhongyu Ma, Yanxing Liu, Saisai Yang, and Fangfang Li. Learning to multi-vehicle cooperative bin packing problem via sequence-to-sequence policy network with deep reinforcement learning model. *Computers & Industrial Engineering*, 177:108998, March 2023.
23. Chaofan Tu, Ruibin Bai, Uwe Aickelin, Yuchang Zhang, and Heshan Du. A deep reinforcement learning hyper-heuristic with feature fusion for online packing problems. *Expert Systems with Applications*, 230:120568, November 2023.
24. Hang Zhao, Chenyang Zhu, Xin Xu, Hui Huang, and Kai Xu. Learning practically feasible policies for online 3D bin packing. *Science China Information Sciences*, 65(1):112105, January 2022.
25. Y.P. Tsang, D.Y. Mo, K.T. Chung, and C.K.M. Lee. A deep reinforcement learning approach for online and concurrent 3D bin packing optimisation with bin replacement strategies. *Computers in Industry*, 164:104202, January 2025.
26. Xinning Liu, Li Han, Ling Kang, Jiannan Liu, and Huadong Miao. Preference learning based deep reinforcement learning for flexible job shop scheduling problem. *Complex & Intelligent Systems*, 11(2):144, February 2025.
27. Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Bastien Pasdeloup, and Patrick Meyer. Learning to select operators in meta-heuristics: An integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 304(3):1296–1330, February 2023.
28. Fuat Kosanoglu, Mahir Atmis, and Hasan Hüseyin Turan. A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem. *Annals of Operations Research*, 339(1-2):79–110, August 2024.
29. Zheng Wu, Yichuan Li, Wei Zhan, Changliu Liu, Yun-Hui Liu, and Masayoshi Tomizuka. Efficient Reinforcement Learning of Task Planners for Robotic Palletization Through Iterative Action Masking Learning. *IEEE Robotics and Automation Letters*, 9(11):9303–9310, November 2024.
30. Simone Reynoso-Donzelli and Luis Alberto Ricardez-Sandoval. A reinforcement learning approach with masked agents for chemical process flowsheet design. *AIChE Journal*, 71(1):e18584, January 2025.
31. Andoni I. Garmendia, Josu Ceberio, and Alexander Mendiburu. Applicability of Neural Combinatorial Optimization: A Critical View. *ACM Transactions on Evolutionary Learning and Optimization*, 4(3):1–26, September 2024.
32. Erdong Yuan, Shuli Cheng, Liejun Wang, Shiji Song, and Fang Wu. Solving job shop scheduling problems via deep reinforcement learning. *Applied Soft Computing*, 143:110436, August 2023.
33. Jakob Kallestad, Ramin Hasibi, Ahmad Hemmati, and Kenneth Sörensen. A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research*, 309(1):446–468, August 2023.
34. Yuchang Zhang, Ruibin Bai, Rong Qu, Chaofan Tu, and Jiahuan Jin. A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*, 300(2):418–427, July 2022.
35. Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, November 2023.
36. Quanqing Que, Fang Yang, and Defu Zhang. Solving 3D packing problem using Transformer network and reinforcement learning. *Expert Systems with Applications*, 214:119153, March 2023.
37. Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, September 2022.
38. Ronghua Chen, Bo Yang, Shi Li, and Shilong Wang. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 149:106778, November 2020.
39. Pengyi Li, Jianye Hao, Hongyao Tang, Xian Fu, Yan Zheng, and Ke Tang. Bridging Evolutionary Algorithms and Reinforcement Learning: A Comprehensive Survey on Hybrid Algorithms. *IEEE Transactions on Evolutionary Computation*, 29(5):1707–1728, October 2025.
40. Kassem Danach, Hassan Harb, Hussin Jose Hejase, and Louai Saker. A Hybrid Metaheuristic Framework with Reinforcement Learning–Based Heuristic Selection for Large-Scale Combinatorial Optimization. *European Journal of Pure and Applied Mathematics*, 18(3):6602, August 2025.