QUANTITATIVE METHODS IN ECONOMICS
Volume XXVII, No. 1, 2026, pp. 27 – 40

Received: 10.01.2026
Accepted: 17.02.2026

PROXIMAL POLICY OPTIMISATION VERSUS ANT COLONY OPTIMISATION FOR THE THREE-DIMENSIONAL BIN PACKING PROBLEM: A COMPARATIVE STUDY

Michał Sawicki
Łazarski University, Warsaw, Poland
Tomasz Woźniakowski https://orcid.org/0000-0002-0779-4769
Department of Econometrics and Statistics
Warsaw University of Life Sciences – SGGW, Poland
e-mail: tomasz_wozniakowski@sggw.edu.pl

Abstract: This paper compares a Proximal Policy Optimisation (PPO) deep reinforcement-learning agent with an Ant Colony Optimisation (ACO) solver on the offline, heterogeneous-bin three-dimensional bin packing problem (3D-BPP). Both algorithms were evaluated on fifty synthetic instances using a unified composite scoring function covering placement ratio, volume utilisation, bin-count penalty and mean per-bin waste. PPO achieves a higher mean composite score (0.346 vs. 0.283), wins on 38 of 50 instances with an average winning margin of 0.101, and resolves each instance in under 60 seconds on a commodity CPU. ACO exhibits greater score variance and resolves instances in up to 1,706 seconds, but its training-free character makes it relevant when the instance distribution changes too rapidly for policy retraining. The PPO training cost of approximately 5.5 hours is recovered after 58 instances compared with ACO at mean inference times. A paired Wilcoxon signed-rank test is identified as the appropriate significance test once per-instance data are made available.

Keywords: 3D bin packing, Proximal Policy Optimisation, Ant Colony Optimisation, deep reinforcement learning, swarm intelligence, logistics optimisation

JEL classification: C61, C63, M11

https://doi.org/10.22630/MIBE.2026.27.1.2

INTRODUCTION

The three-dimensional bin packing problem (3D-BPP) requires placing a set of rectangular boxes into a finite collection of bins such that boxes do not overlap, do not exceed bin boundaries, and the number of bins used is minimised. The problem arises directly in vehicle loading, pallet building and warehouse slot allocation. In high-volume logistics settings, a one-percent improvement in average bin fill translates into fewer truck movements and lower fuel cost per unit shipped [Wäscher et al. 2007; Gzara et al. 2020].

Because 3D-BPP is strongly NP-hard [Garey, Johnson 1979; Christensen et al. 2016], optimal solvers are tractable only for small instances. Two algorithmic families have attracted sustained research attention. Ant Colony Optimisation (ACO) solves each instance from scratch: simulated ants build candidate packings guided by pheromone trails that reinforce good solutions and decay over time [Dorigo, Gambardella 1997; Dorigo, Stützle 2018]. Proximal Policy Optimisation (PPO), a deep reinforcement-learning (DRL) method, invests computation in an off-line training phase and then resolves new instances through fast policy inference [Schulman et al. 2017].

The two approaches imply different cost structures. ACO requires no training but spends computation at inference time on every instance. PPO requires substantial training but resolves instances in seconds once trained. Whether the PPO training overhead is economically justified depends on how many instances an organisation solves and at what rate the instance distribution changes. This trade-off is well-known in principle but rarely quantified empirically under identical experimental conditions for 3D-BPP specifically.

This paper places both algorithms on the same fifty synthetic offline 3D-BPP instances, evaluates them with the same composite scoring function, and reports (i) packing quality statistics, (ii) instance-level win/loss counts, (iii) runtime distributions, and (iv) a break-even analysis quantifying how many instances PPO must solve to recover its training cost relative to ACO. The paper is structured as follows: Section 2 reviews the relevant literature; Section 3 describes algorithm implementations and the benchmark protocol; Section 4 reports results; Section 5 discusses findings and limitations; Section 6 concludes.

LITERATURE REVIEW

Problem Definition and Typology

Wäscher, Haußner and Schumann [2007] provide the canonical taxonomy of cutting-and-packing problems, classifying them by objective function, item–container relationship and temporal availability of item data. Under that taxonomy, 3D-BPP is an offline V-type problem: all box dimensions w_i, h_i, d_i are known at the start, the algorithm may open as many bins of given dimensions as needed, and the

goal is to minimise the bin count. Earlier typologies by Dyckhoff [1990] and Dyckhoff and Finke [1992] established the terminological foundation. NP-hardness of 3D-BPP follows from the one-dimensional case [Garey, Johnson 1979]; Christensen et al. [2016] survey approximation results and complexity boundaries. In practice, instances with more than a few dozen items require heuristic or learning-based solvers.

Swarm Intelligence Approaches

ACO was introduced by Dorigo and Gambardella [1997] for the Travelling Salesman Problem and adapted to bin packing by Levine and Ducatelle [2004] for the one-dimensional case. For 3D-BPP the dominant hybrid couples ACO with the three-dimensional heuristic algorithm (3DHA) of Silveira et al. [2013]: ants determine the box sequence and 3DHA resolves placement via an extreme-point first-fit rule. Viegas et al. [2014, 2015] applied this hybrid to the steel-cutting variant of 3D-BPP, finding ACO competitive with tabu search and simulated annealing within practical time budgets. Li and Zhang [2015] extended the approach to heterogeneous containers. Maia and Borenstein [2024] combined ACO with column generation, obtaining improved lower bounds on large instances. Dorigo and Stützle [2018] survey pheromone update variants including the MAX–MIN Ant System (MMAS) used in the present study.

Deep Reinforcement Learning Approaches

Hu et al. [2017] first applied DRL to a 3D packing variant, training a policy to sequence boxes so as to minimise bin surface area. Schulman et al. [2017] introduced PPO as a policy-gradient method that clips the surrogate objective to keep updates within a trust region, achieving stable convergence on high-dimensional action spaces. For 3D-BPP, the standard design factorises the action space: a deterministic geometric routine identifies feasible placement anchors, and the neural network selects among (item, orientation) pairs. This keeps the learned component compact while offloading geometric feasibility checking to deterministic code [Murdivien, Um 2023]. Que et al. [2023] coupled a transformer encoder with PPO (TransPack), achieving state-of-the-art volume utilisation on offline instances by using multi-head attention to represent item sets of variable size. Xiong et al. [2024] extended this to the online setting (GOPT), where items arrive sequentially; their generalisation across bin sizes stems from the size-agnostic nature of the transformer. Zhao et al. [2022] trained an actor–critic network that predicts placement coordinates successively (length, width, rotation), using the critic to veto infeasible moves.

Gap in the Literature

Direct, standardised comparisons between ACO and PPO on identical 3D-BPP instances are absent from the literature. Viegas et al. [2014] benchmark ACO against tabu search but not against DRL methods. Que et al. [2023] compare

transformer-PPO against earlier DRL approaches but not against metaheuristics. The present paper fills this gap with a controlled head-to-head experiment that also quantifies the economic break-even point of PPO training.

METHODOLOGY

Problem Formulation

The benchmark uses an offline, heterogeneous-bin 3D-BPP. Each instance specifies a set of boxes with dimensions (w_i, h_i, d_i) and a set of bins with dimensions (W_k, H_k, D_k), all provided simultaneously. Boxes must be placed with edges parallel to bin faces, without mutual overlap and without exceeding bin boundaries. All six axis-aligned rotations are permitted. The ACO solver additionally prefers lower placement positions to simulate gravity. The objective is the composite score:

$$score = 0.65 \cdot p + 0.25 \cdot u - 0.30 \cdot b - 0.25 \cdot \mu w \quad (1)$$

Placement ratio p = (boxes placed)/(boxes total); volume utilisation u = (Σ box volumes placed)/(Σ bin volumes); bin penalty b = (bins used)/(bins available); mean per-bin waste $\mu w = (1/m) \cdot \Sigma_i(1-u_i)$ where m is the number of bins used. The coefficient on p (0.65) is the largest because complete placement is the primary industrial requirement; b (0.30) and μw (0.25) penalise inefficient bin usage; u (0.25) rewards density. An earlier formulation with equal weights on p and u (0.40 each) produced an agent that opened extra bins to maintain high utilisation per bin; the revised weights eliminate this incentive.

PPO Implementation

At each decision step a deterministic corner-search routine identifies the first geometrically feasible anchor point in the current bin, reducing the learning problem to selecting an (item, orientation) pair. This factorisation keeps the action space compact and avoids learning geometric feasibility, which would require far more training data [Que et al. 2023; Murdivien, Um 2023].

The actor–critic network consists of six fully connected layers with ReLU activations (layers 1–4: 256 units; layers 5–6: 128 units). Actor and critic share all six layers and diverge only at the output projection. A binary placement mask zeroes out already-packed items, so the softmax over action logits considers only unpacked boxes and their six orientations. Weight sharing halves parameter count and couples the value estimate to the same feature space used for action selection.

Training uses 2,000 episodes drawn from 700 knapsack instances [Beasley 1990–2018, MIT licence] augmented with randomly generated cases in a 1:2 ratio. PPO clipping threshold: $\varepsilon = 0.10$; discount factor: $\gamma = 0.99$; learning rate halved at episode 1,000. The step reward is the increment in composite score; a terminal bonus of +0.1 is added when all boxes are placed. Total training time: approximately 5.5 hours on the experimental CPU.

The Markov Decision Process: state s_t = [placement mask, box dimensions, bin dimensions, current score]; action a_t = index of a feasible (item, orientation) pair; transition: deterministic geometry check then state update; reward $r_t = score_t - score_{t-1}$ + completion bonus.

ACO Implementation

The solver implements MMAS [Dorigo, Stützle 2018]. Pheromone levels are stored in a hash-map keyed by (box index, extreme point) pairs, keeping memory proportional to the number of extreme points rather than to all possible placements [Silveira et al. 2013]. Each iteration: (1) evaporate all trails by factor 0.8; (2) each ant samples a box permutation proportional to pheromone$^{\alpha}$ · heuristic$^{\beta}$; (3) 3DHA decodes the permutation into placements via extreme-point first-fit; (4) the top five ants deposit pheromone proportional to their normalised score and rank. If the best score does not improve for 10% of iterations, all trails are halved (soft reset).

Colony size adapts to instance complexity via:

$$ants = clamp(round(b \cdot log_2(n+1) \cdot s), 50, 8000) \quad (2)$$

$$iterations = clamp(round(n \cdot log_2(b+1) \cdot 4s), 150, 5000) \quad (3)$$

where n = box count, b = bin count, s = 2. This contrasts with prior work that fixes colony size regardless of instance size [Viegas et al. 2014; Maia, Borenstein 2024]. The extreme-point list is capped at 1,000 entries per iteration to bound runtime on large instances. The solver is written in Rust and called from Python via PyO3 bindings, which eliminates CPython interpreter overhead in the inner loop.

Note on statistical validation of the dynamic colony formula: a controlled ablation study comparing the dynamic formula against a fixed baseline (e.g. 100 ants, 300 iterations) across instances of varying size would quantify the runtime savings and any score trade-off. Such an experiment is a necessary next step before the formula can be recommended as a general design principle; it is identified here as immediate future work.

Benchmark Design

Fifty independent instances were generated synthetically with box and bin dimensions drawn from a uniform distribution subject to the constraint that each box fits inside at least one bin. For each instance the benchmark driver ran PPO inference first (weights loaded from a .pt checkpoint), then the ACO solver on the same instance. Wall-clock CPU time was recorded separately for each algorithm. All results were written to a single CSV file. Hardware: 6-core/12-thread CPU at 3.6 GHz base clock, GPU disabled, so both algorithms face identical latency constraints. Software: Python 3.11, PyTorch 2.7.0 [Ansel et al. 2024], gym-BinPack3D environment [2024].

Statistical note: the paired Wilcoxon signed-rank test [Wilcoxon 1945] is the appropriate non-parametric test for comparing the two algorithms, because each of the 50 instances was solved by both methods under identical conditions, producing

matched pairs of scores. The test requires per-instance score pairs rather than aggregate statistics. Those data are stored in the benchmark CSV and should be analysed in the final version of the paper; the present analysis is therefore confined to descriptive statistics.

RESULTS

Overall Composite Score

Table 1 reports descriptive statistics for the composite scores. PPO's mean of 0.346 exceeds ACO's mean of 0.283 by 0.063 points (22% relative). PPO's standard deviation (0.042) is half of ACO's (0.085), indicating that PPO's performance varies less across instances. ACO's maximum (0.458) marginally exceeds PPO's (0.417), showing that the colony can occasionally find tighter packings, but ACO's minimum (0.141) is 0.064 below PPO's floor (0.205). Whether the 0.063-point gap in means is statistically significant requires a paired Wilcoxon test on per-instance scores (see Section 3.4); the descriptive evidence is consistent with a systematic advantage for PPO.

Table 1. Composite score statistics across 50 benchmark instances

Metric	Mean	Median	Std. Dev.	Min	Max
PPO score	0.346	0.350	0.042	0.205	0.417
ACO score	0.283	0.275	0.085	0.141	0.458
Δ (PPO−ACO)	0.063	0.089	0.081	−0.041	0.064

Source: own calculations

Head-to-Head Comparison

Table 2 records win/loss outcomes per instance. PPO wins on 38 of 50 instances (76%) with a mean winning margin of 0.101. ACO wins on 12 instances (24%) with a mean margin of 0.056. There are no ties. PPO's winning margin (0.101) is 1.8× larger than ACO's (0.056), meaning PPO not only wins more often but wins by larger amounts. The 12 ACO victories cluster on instances where box and bin geometry aligns with 3DHA's first-fit scanning direction (see Section 5.2).

Table 2. Win/loss record across 50 instances

Method	Wins	Ties	Win rate	Mean margin when winning
PPO	38	0	76%	0.101
ACO	12	0	24%	0.056

Source: own calculations

Placement Ratio

Both algorithms achieve a placement ratio of 1.0 on all 50 instances: every box is packed in every run. Score differences between PPO and ACO therefore reflect bin-count efficiency and void management exclusively, not item completeness.

Bin Penalty

Table 3 shows bin penalty statistics. PPO's mean bin penalty of 0.516 is 0.298 points below ACO's mean of 0.814, indicating that PPO uses on average 30 percentage points fewer of the available bins. PPO's lower standard deviation (0.155 vs. 0.198) confirms more predictable bin usage. The training diagnostics show that the agent learned to return to partially filled bins rather than opening new ones, a behaviour driven by the b coefficient in the reward function.

Table 3. Bin penalty statistics across 50 instances

	Mean	Median	Std. Dev.	Min	Max
PPO	0.516	0.500	0.155	0.200	1.000
ACO	0.814	0.800	0.198	0.400	1.000

Source: own calculations

Runtime and Break-Even Analysis

Table 4 shows inference runtimes. PPO's mean of 13.2 s and median of 10.8 s contrast sharply with ACO's mean of 359 s and median of 96.5 s. ACO's distribution is heavily right-skewed (max 1,706 s), driven by large instances where the dynamic colony formula allocates up to 8,000 ants. PPO's maximum of 57 s reflects a fixed network forward pass whose cost scales with box count but not quadratically.

Table 4. Inference runtime (seconds) across 50 instances

	Mean	Median	Std. Dev.	Min	Max
PPO	13.17	10.79	12.09	0.15	56.73
ACO	359.14	96.46	451.18	1.73	1705.72

Source: own calculations

Table 5 presents the break-even analysis. PPO's training phase required approximately 5.5 hours (19,800 s) on the experimental hardware. The mean time saving per instance relative to ACO is 359.14 − 13.17 = 345.97 s. Dividing training cost by saving per instance gives a break-even of ⌈ 19,800 / 345.97 ⌉ = 58 instances. An organisation that solves more than 58 similar packing instances recovers the PPO training investment and thereafter saves 345.97 s per instance. At the median ACO

runtime (96.5 s) the break-even rises to 241 instances, reflecting the fact that easy instances provide smaller time savings.

Table 5. Break-even analysis: PPO training cost vs. inference time saving

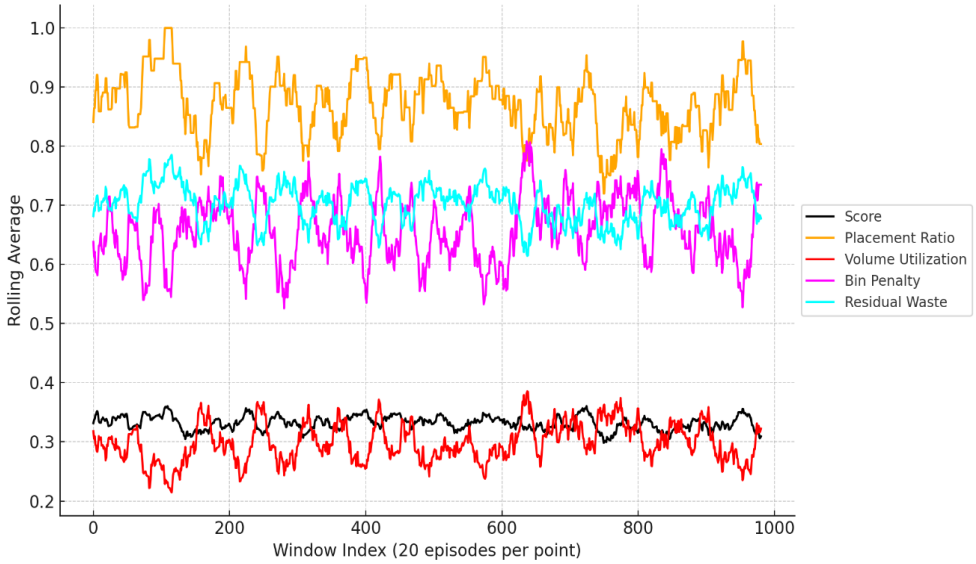
Parameter	Value
PPO training time	19,800 s (5.5 h)
Mean ACO inference time per instance	359.14 s
Mean PPO inference time per instance	13.17 s
Mean time saving per instance	345.97 s
Break-even (mean runtime)	58 instances
Break-even (median runtime, 96.5 s ACO)	241 instances

Source: own calculations. Break-even = ⌈ training time / mean saving per instance⌉ .

Training Dynamics

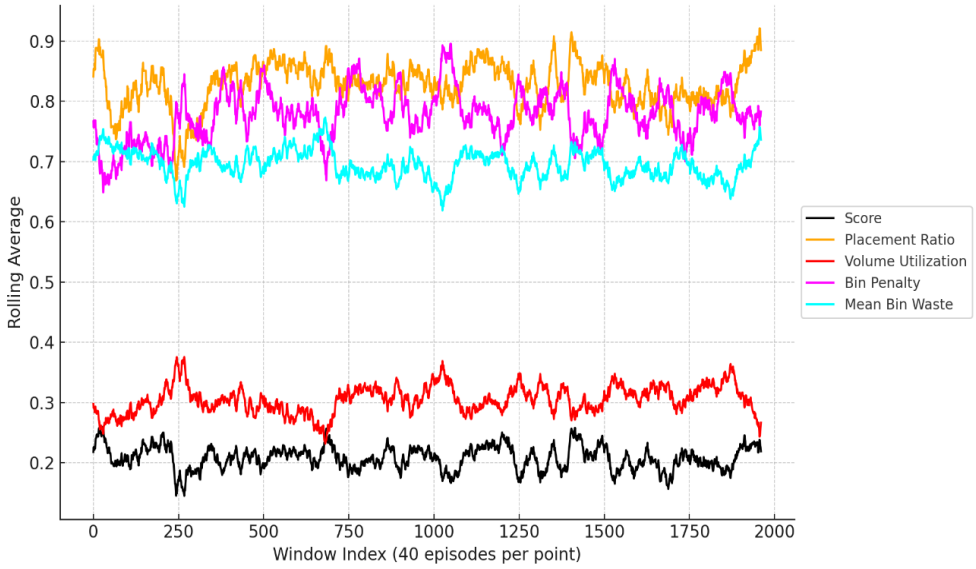
Figures 1–3 show rolling-average training curves for the five metrics (p, u, b, μw, composite score) across three reward formulations. In the first formulation (equal weights 0.40 on p and u), the agent raised placement ratio rapidly but did not reduce bin count, because opening a new bin had no cost that outweighed utilisation gains. Replacing the aggregate waste term with per-bin waste μw and increasing the bin-penalty coefficient shifted the equilibrium: the final training run (Figure 3) shows b declining gradually while p stays above 0.95, confirming that the agent learned to consolidate items into fewer bins rather than maximising per-bin fill in isolation.

Figure 1. Rolling averages (20-episode window) of training metrics, initial reward formulation, 1,000 episodes



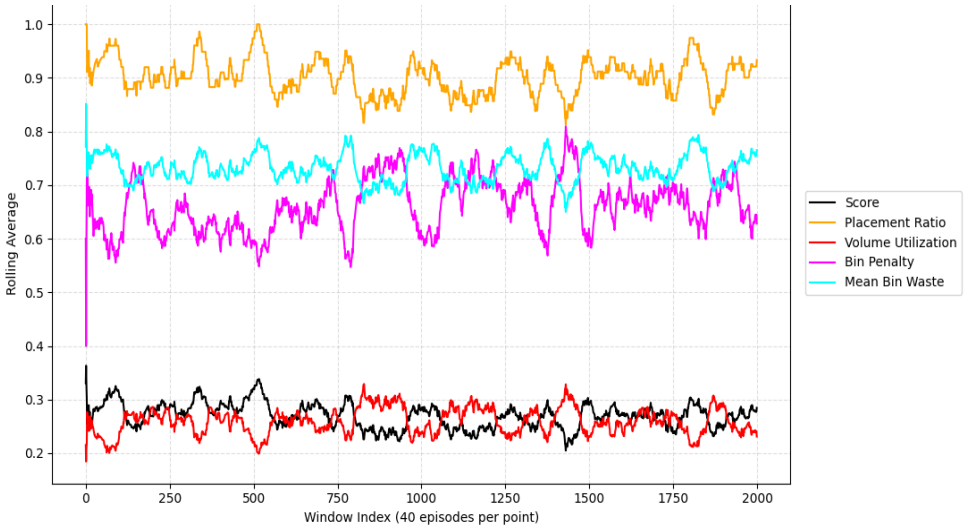
Source: own preparation

Figure 2. Rolling averages (40-episode window), intermediate formulation, 2,000 episodes.



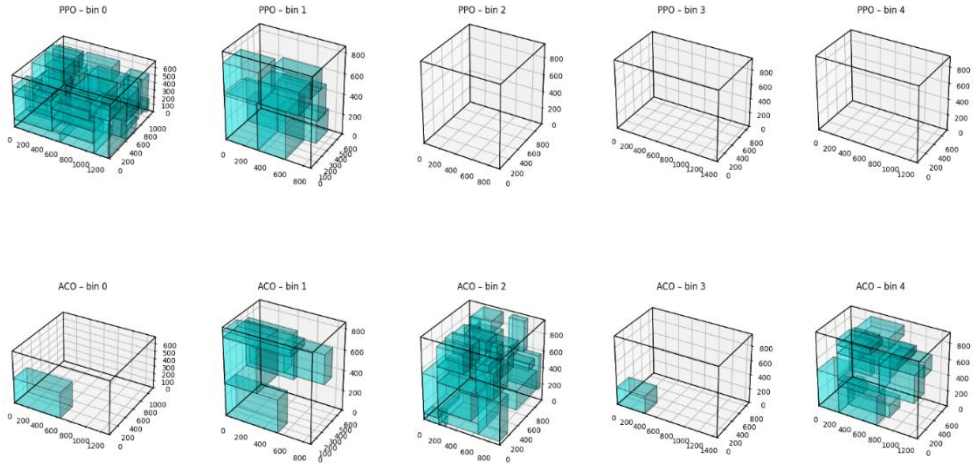
Source: own preparation

Figure 3. Rolling averages (40-episode window), final reward formulation, 2,000 episodes



Source: own preparation

Figure 4. Visual representation of packing solutions: PPO (left) and ACO (right) on a representative instance



Source: own preparation

DISCUSSION

Why PPO Scores Higher

Three implementation choices contributed to PPO's advantage. First, the action-space factorisation (deterministic corner search + learned item–orientation selection) eliminates infeasible placements before they reach the network. This keeps the gradient signal clean and reduces the number of training episodes needed to learn a useful policy [Que et al. 2023; Murdivien, Um 2023]. Second, the step reward $r_t = \text{score}_t - \text{score}_{t-1}$ provides a non-zero learning signal at every step, avoiding the sparse-reward problem that would arise if reward were given only at episode end. Third, the shared actor–critic backbone couples the value function to the same feature representation used for action selection, which stabilises advantage estimation and reduces variance in policy gradient updates.

Why ACO Scores Lower on Average

ACO's higher variance stems from two sources. First, the dynamic colony-size formula may under-allocate ants on instances that appear small by box and bin count but have a complex feasible-placement geometry, causing pheromone trails to converge prematurely on a sub-optimal sequence. Second, 3DHA places each box at the first feasible extreme point in a fixed scanning order; if that order does not match the geometry of a specific instance, no amount of additional ant iterations can recover the missed placement. The 12 ACO victories are consistent with instances where 3DHA's scanning order happens to align with the optimal packing direction, yielding a tight arrangement that the PPO policy, trained on a broader distribution, does not replicate.

Practical Deployment Criteria

The break-even analysis in Table 5 translates the algorithmic comparison into an operational decision rule. PPO is cost-effective when: (a) the organisation solves more than 58 instances of comparable size and structure before the policy requires retraining, and (b) inference latency below 60 s is compatible with the dispatch scheduling window. ACO is preferable when: (a) fewer than 58 instances are expected, (b) the box–bin size distribution changes rapidly (e.g. new product lines introduced weekly), making retraining impractical, or (c) no GPU or training infrastructure is available.

The complementary performance profiles also suggest a practical hybrid: use PPO to generate an initial packing within seconds, then run ACO for a fixed time budget as a local search phase. On the 12 instances where ACO currently wins, its margin averages 0.056; whether an ACO post-processing pass on a PPO solution can capture similar improvements is an empirical question for future work.

Limitations

Four limitations constrain the generalisability of these results. First, all instances were generated from a single synthetic distribution; the algorithms have not been tested on real warehouse order data where box-size distributions are typically skewed by fast-moving SKUs. Second, the lack of per-instance score data in the current thesis prevents a formal paired Wilcoxon test; the 0.063-point mean difference and 76% win rate are descriptively convincing but not inferentially confirmed. Third, both algorithms delegate placement to a deterministic heuristic (corner search for PPO, 3DHA for ACO); end-to-end learned placement such as GOPT [Xiong et al. 2024] may close or reverse the observed gap. Fourth, the dynamic colony-size formula has not been validated against a fixed-size baseline; it is possible that 100 ants with 300 iterations would match or exceed the dynamic formula on most instances at lower computational cost.

SUMMARY

This paper compared a PPO agent and an ACO solver on fifty offline heterogeneous-bin 3D-BPP instances under identical scoring and hardware conditions. The main findings are as follows.

PPO achieves a mean composite score of 0.346 vs. 0.283 for ACO (22% higher) and wins on 76% of instances with a mean margin of 0.101, compared to ACO's mean margin of 0.056 in its 24% of wins. Both algorithms achieve a perfect placement ratio of 1.0 on all instances; the score gap is driven by bin-count efficiency (PPO mean bin penalty 0.516 vs. ACO 0.814). PPO resolves instances in a mean of 13.2 s vs. 359 s for ACO; the PPO training cost of 19,800 s is recovered after 58 instances at mean ACO runtime.

Three actions are required before the results can be considered fully rigorous: (1) extract per-instance score pairs from the benchmark CSV and compute a paired Wilcoxon signed-rank test; (2) run an ablation study comparing the dynamic colony-size formula against a fixed baseline (100 ants, 300 iterations) across instances of varying size; (3) evaluate both algorithms on real warehouse order data to assess external validity.

Subject to these qualifications, the results support the following deployment recommendation: PPO is the preferred solver for operations that process more than approximately 60 packing instances of stable composition, because training cost is recovered within the first 60 runs and inference remains below 60 s on commodity hardware. ACO is preferable for low-volume or rapidly-changing environments where policy retraining is impractical. The complementary win profiles of the two methods suggest that a PPO–ACO hybrid merits empirical investigation.

REFERENCES

Ansel J., Yang E., He H., Gimelshein N., Jain A., Voznesensky M., … Chintala S. (2024) PyTorch 2: Faster Machine Learning through Dynamic Python Bytecode Transformation and Graph Compilation. Proceedings of ASPLOS '24. ACM. https://doi.org/10.1145/3620665.3640366

Christensen H. I., Khan A., Pokutta S., Tetali P. (2016) Multidimensional Bin Packing and Other Related Problems: A Survey. Computer Science Review, 24.

Dorigo M., Gambardella L. M. (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1(1), 53-66.

Dorigo M., Stützle T. (2018) Ant Colony Optimization: Overview and Recent Advances. [in:] Handbook of Metaheuristics. Springer, 311-351.

Dyckhoff H. (1990) A Typology of Cutting and Packing Problems. European Journal of Operational Research, 44(2), 145-159.

Dyckhoff H., Finke U. (1992) Cutting and Packing in Production and Distribution: A Typology and Bibliography. Springer.

Elhedhli S., Gzara F., Yildiz B. C. (2019) Three-Dimensional Bin Packing and Mixed-Case Palletization. INFORMS Journal on Optimization, 1(4), 323-352.

Garey M. R., Johnson D. S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.

gym-BinPack3D [Source Code] (2024) https://github.com/ylchan87/gym-BinPack3D

Gzara F., Elhedhli S., Yildiz B. C. (2020) The Pallet-Loading Problem: Three-Dimensional Bin Packing with Practical Constraints. European Journal of Operational Research, 287, 545-565.

Hu H., Zhang X., Yan X., Wang L., Xu Y. (2017) Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method. arXiv preprint arXiv:1708.05930.

Kanna S. K., Jaisree A. D., Balasundaram K., Kumar S. B. (2015) Optimization of 3D Constrained Rectangular Bin Packing Problem using Recursive Ant Colony Algorithm. IOSR Journal of Mechanical and Civil Engineering, 12(4), 65-70.

Karaboga D., Basturk B. (2008) On the Performance of Artificial Bee Colony (ABC) Algorithm. Applied Soft Computing, 8(1), 687-697.

Lampropoulos A. S., Tsihrintzis G. A. (2015) Machine Learning Paradigms: Applications in Recommender Systems. Springer International Publishing.

Levine J., Ducatelle F. (2004) Ant Colony Optimization and Local Search for Bin Packing and Cutting Stock Problems. Journal of the Operational Research Society, 55(7), 705-716.

Li X., Zhang K. (2015) A Hybrid Differential Evolution Algorithm for Multiple Container Loading Problem with Heterogeneous Containers. Computers & Industrial Engineering, 90, 305-313.

Maia D. B., Borenstein D. (2024) A Hybrid Approach Combining Ant Colony Optimization and Column Generation for Solving the 3D Bin Packing Problem with Rotation. Available at SSRN 4978748 [preprint].

Manthey B., van Rhijn J., Safari A., Vredeveld T. (2025) Convergence and Running Time of Time-Dependent Ant Colony Algorithms. arXiv preprint arXiv:2501.10810 [preprint].

Murdivien S. A., Um J. (2023) BoxStacker: Deep Reinforcement Learning for 3D Bin Packing Problem in Virtual Environment of Logistics Systems. Sensors, 23(15), 6928.

Ojha V. K., Abraham A., Snášel V. (2014) ACO for Continuous Function Optimization: A Performance Analysis. Proceedings of the 14th International Conference on Intelligent Systems Design and Applications, 145-150. IEEE.

Que Q., Yang F., Zhang D. (2023) Solving 3D Packing Problem using Transformer Network and Reinforcement Learning. Expert Systems with Applications, 214, 119153.

Sangeetha V., Krishankumar R., Ravichandran K. S., Cavallaro F., Kar S., Pamucar D., Mardani A. (2021) A Fuzzy Gain-Based Dynamic Ant Colony Optimization for Path Planning in Dynamic Environments. Symmetry, 13(2), 280.

Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. (2017) Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

Silveira M. E., Vieira S. M., Da Costa Sousa J. M. (2013) An ACO Algorithm for the 3D Bin Packing Problem in the Steel Industry. [in:] Recent Trends in Applied Artificial Intelligence: IEA/AIE 2013, 535–544. Springer Berlin Heidelberg.

Singh N. K., Baidya S. (2013) A Novel Work for Bin Packing Problem by Ant Colony Optimization. International Journal of Research in Engineering and Technology, 2(2), 71-73.

Skackauskas J., Kalganova T., Dear I., Janakiram M. (2022) Dynamic Impact for Ant Colony Optimization Algorithm. Swarm and Evolutionary Computation, 69, 100993.

Viegas J. P., Vieira S. M., Sousa J. M., Henriques E. M. (2014) Metaheuristics for the 3D Bin Packing Problem in the Steel Industry. 2014 IEEE Congress on Evolutionary Computation (CEC), 338-343. IEEE.

Viegas J. L., Vieira S. M., Henriques E. M., Sousa J. M. (2015) A Tabu Search Algorithm for the 3D Bin Packing Problem in the Steel Industry. CONTROLO 2014 – Proceedings of the 11th Portuguese Conference on Automatic Control, 355-364. Springer.

Wäscher G., Haußner H., Schumann H. (2007) An Improved Typology of Cutting and Packing Problems. European Journal of Operational Research, 183(3), 1109-1130.

Wilcoxon F. (1945) Individual Comparisons by Ranking Methods. Biometrics Bulletin, 1(6), 80-83.

Xiong H., Guo C., Peng J., Ding K., Chen W., Qiu X., … Xu J. (2024) GOPT: Generalizable Online 3D Bin Packing via Transformer-Based Deep Reinforcement Learning. IEEE Robotics and Automation Letters.

Zeineldin R. A., Morsy A. M. (2015) A Modified Artificial Bee Colony for Solving the Container Loading Problem. International Journal of Computer Applications, 114(3).

Zhao H., Zhu C., Xu X., Huang H., Xu K. (2022) Learning Practically Feasible Policies for Online 3D Bin Packing. Science China Information Sciences, 65(1), 112105.