A Preliminary Study on GAN-Augmented Column Generation

Mick Molitor[1 [0009−0001−9309−8179]] and Natalia Kliewer[2 [0000−0002−7537−3309]]

Freie Universität Berlin, Germany

Abstract. This work investigates the use of synthetic data generation as a warm-start mechanism for Column Generation (CG), focusing on the LP relaxation of the one- and two-dimensional cutting stock problem. We propose a hybrid framework in which a generative adversarial network (GAN), trained on heuristically generated cutting patterns, is used to produce candidate columns during the initial iterations of CG before switching to exact pricing. The objective is not to replace exact pricing, but to reduce the number of expensive pricing calls required to reach LP optimality. Initial experiments across instances of increasing scale show that GAN-based warm starts seem to improve early convergence and reduce the remaining number of exact pricing calls. While heuristics remain slightly more effective in the majority of instances, the generative approach reproduces the same qualitative trends without relying on their handcrafted designs. The results demonstrate the feasibility of learning-based CG and highlight its potential for more complex optimization problems where pricing is costly and data scarcity is prevalent.

Keywords: Column Generation · Generative Adverserial Networks · Synthetic Data

1 Introduction

In many modern domains such as transportation, healthcare, and logistics, the quality and quantity of available data play a decisive role in shaping effective decision-making processes. Yet, data scarcity is a persistent challenge: either comprehensive datasets are missing entirely or the available data is too limited, constraining the quality of downstream models and solutions.

Column generation (CG) is a powerful technique in large-scale combinatorial optimization problems, enabling practitioners to solve linear programs (LP) with an astronomical number of variables by iteratively introducing (or "generating") only the most promising variables (columns). The original problem is therefore decomposed into a (restricted) master problem (RMP), which selects an optimal subset of columns, and a pricing problem, which generates additional promising columns for the (restricted) master problem. The exchange of dual price information between the master and pricing problems ensures the convergence of the method. By decomposing the original, intractable problem and gradually refining the solution space, CG effectively can solve LP with a large number of

variables.
However, CG's efficiency relies heavily on the quality and the solvability of the underlying pricing problem. If the pricing problem is computationally expensive, executing a large number of iterations quickly results in substantial computational overhead. Alot of work in CG has focused on improving pricing quality, either through the use of exact methods and heuristic approaches to identify promising new columns, or by employing machine learning techniques to predict the potential quality improvement associated with candidate columns. Docking on this idea, this works tries to use synthetic data generation, especially with the use of generative adversarial networks (GAN) [11], to learn from historical solutions and propose novel, diverse columns. By generating synthetic columns that capture useful structures from historical data without duplicating them, generative models may provide a principled way to inject high-quality candidate solutions into the optimization process.

In this work, we explore whether integrating synthetically generated columns, derived from GANs, can be used as a way of circumventing the classical pricing problem and act as a generator for new columns and if these generated columns have any benefit in the later convergence of the problem. We use the fundamental Cutting Stock Problem (CSP) as a benchmark problem to analyze whether GAN-generated columns, when incorporated into the solving process, can help identify promising solutions earlier and can improve convergence speed. The focus of this work is on the solution of the LP-relaxed CSP.

In this work we try to bridge the gap between synthetic data generation and operations research, proposing an integration of GAN-generated columns into the classical CG framework.
We compare generative, heuristical and exact methods for initial CG, analyzing their effect on convergence speed and solution quality on a well known problem. We provide initial empirical data on the pros and limitations of synthetic CG in optimization problems.

2 Related Work

2.1 Foundations and Practical Challenges of Column Generation

CG was originally introduced in the work of [8, 9] for the CSP and has since evolved into a core methodology for solving large-scale LP with an exponential number of variables. Comprehensive surveys by [13], as well as [3] in the context of branch-and-price, highlight both the theoretical foundations and the practical challenges of CG.

During the initial iterations, dual variables are typically unstable and poorly informative, which can lead to slow progress or excessive oscillation. As a consequence, substantial research has focused on warm-start strategies, column pool reuse, and dual stabilization techniques[4, 10]. These approaches do not alter the optimality guarantees of CG but can significantly affect convergence speed and computational efficiency in practice.

2.2 Heuristic and Sampling-Based CG

Beyond research on exact pricing in CG, a substantial body of work has examined heuristic and sampling based strategies for CG, motivated by the observation that solving the pricing subproblem to optimality is often computationally prohibitive in large scale applications. Heuristical sampling and restricted pricing schemes aim to explore only carefully selected regions of the search space, thereby accelerating the identification of improving columns while accepting a controlled loss of optimality guarantees. Common mechanisms include premature termination after the first or kth negative reduced cost column is detected or selective state expansion based on dual information [12]. While these approaches frequently yield early improving columns that stabilize the master problem and promote rapid convergence, they also introduce challenges such as the potential regeneration of previously identified columns, sensitivity to parameter choices, and difficulties in balancing exploration and exploitation. Consequently, heuristic CG is best understood as a complementary mechanism to exact pricing, particularly effective in early iterations where computational efficiency and the rapid discovery of promising candidates outweigh the need for provable optimality[6].

2.3 Machine Learning for CG and Pricing

In recent years, machine learning has been increasingly integrated into CG and branch-and-price frameworks. The majority of contributions focus on accelerating or approximating the pricing problem, which is often the dominant computational bottleneck. Representative approaches include learning to predict whether a candidate column is likely to have negative reduced cost [14–16], learning to accelerate convergence of an IP [17], and reinforcement learning methods that approximate pricing decisions directly[5, 1].

These methods share a common structure in which the classical pricing problem remains central, while machine learning serves as a mechanism for acceleration or selection. Empirical results indicate that such hybrid approaches can substantially reduce the number of pricing calls without compromising solution quality.

In contrast, comparatively little attention has been paid to learning mechanisms that directly synthesize new columns independently of the pricing problem. This distinction is critical. While learning-assisted pricing aims to solve the same subproblem more efficiently, learning-based column synthesis targets a different phase of CG, namely the orcale function and early exploration of the solution space. The present work contributes to this underexplored direction by investigating generative models as direct column proposal mechanisms.

2.4 Generative Models for Combinatorial Optimization Problems

Generative models and combinatorial or mixed-integer programming have so far only been combined in a limited number of studies, and, to the best of the authors' knowledge, an explicit integration of generative models within a CG

framework has not yet been reported in the literature. Nonetheless, the general idea of coupling generative learning techniques with exact or heuristic optimization methods has gained increasing attention in recent years. Several recent contributions explicitly explore how generative models can be embedded into optimization pipelines, typically as auxiliary components that support solution construction, initialization, or search guidance, rather than as replacements for classical optimization algorithms.

Several studies have demonstrated the viability of GANs in operational decision-making contexts where optimization models remain central. For instance, [18] propose a data-driven system for cooperative bus route planning in which GANs are used to generate realistic demand scenarios. GAN-based approaches have also been employed more directly at the solution-construction level. [20] integrate a GAN into a genetic algorithm for the three-dimensional bin packing problem. In their approach, the generator produces high-quality packing configurations that are injected into the evolutionary population, improving both exploration and convergence speed. Beyond direct solution generation, generative models have been combined with reinforcement learning to learn improvement heuristics. [19] introduce a generative inverse reinforcement learning framework for routing problems, where GANs are used to infer reward functions and policies for classical 2-opt local search heuristics without relying on manually specified rewards. Taken together, these works suggest a unifying interpretation of generative models in combinatorial optimization. Rather than acting as replacements for exact or heuristic optimization algorithms, generative models function as distributional learners that bias the search toward promising regions of the solution space. Their outputs may be infeasible, suboptimal, or redundant, but when combined with filtering, evaluation, or repair mechanisms, they can substantially improve early exploration and convergence behavior.

This interpretation aligns naturally with the general concept of this work within CG. Columns synthesized by a generative model can be viewed as heuristic proposals sampled from a learned distribution of historically effective patterns (or possible related data from other problems). Feasibility and reduced-cost optimality are then enforced by the RMP and pricing logic, ensuring that the theoretical properties of CG are preserved.

3 Methodology

3.1 Problem Formulation

We both consider the one and two-dimensional CSP [8, 7] (CSP & 2D-CSP) where rectangular items must be cut from stock sheets of fixed dimensions. Following the Dantzig-Wolfe decomposition, we reformulate the problem as a set covering model where each column represents a feasible cutting pattern.

Let $\mathcal{P}$ denote the set of feasible patterns, where each pattern $p \in \mathcal{P}$ is characterized by a vector $a^p = (a_1^p, \ldots, a_m^p)$ indicating how many items of each type

i are cut. The master problem (MP) minimizes the number of sheets used:

$$\min \quad \sum_{p \in \mathcal{P}} x_p \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}} a_{i\,p}^{p} \geq d_i, \quad \forall i \in \{1, \ldots, m\} \tag{2}$$

$$x_p \geq 0 \tag{3}$$

Since $|\mathcal{P}|$ is exponentially large, we solve the LP relaxation via CG, iteratively adding patterns with negative reduced cost $\bar{c}_p = 1 - \sum_i \pi_i a_i^p$, where π_i are the dual prices.

3.2 Pricing Oracle Based on the Gilmore Gomory Two Stage Guillotine Model

For exact pricing, we restrict attention to two-stage guillotine cutting patterns in the sense of [7]. In this class of patterns, the stock sheet of width W and height H is first partitioned into vertical strips whose widths are chosen from the set of item widths. Each strip is then cut orthogonally into items of matching width, which leads to a natural decomposition of the pricing problem into two nested one-dimensional knapsack problems.

For a given candidate strip width W_s, the value of a strip is obtained by solving the knapsack problem V_s which determines the optimal filling of the strip along the height dimension. The strip values are then combined by solving a second knapsack problem over the sheet width, V^*. If the reduced cost condition $1 - V^* < 0$ is satisfied, the corresponding two-stage guillotine pattern is generated and added to the master problem. This approach yields exact pricing over the two-stage guillotine pattern space while remaining computationally tractable.

3.3 Training Data Generation

To generate diverse training data for our generative model, we use stochastic heuristics that preserve the guillotine structure. Specifically, we employ a greedy heuristic that, for a given strip width, packs items sequentially, ordered by value density or at random, until the height capacity is reached, and then combines strips greedily across the sheet width. In addition, we use a randomized heuristic that increases diversity by permuting the item order, sampling strip widths, and probabilistically accepting suboptimal items.

These heuristics generate $N = 500$ training patterns that are feasible by construction, providing the foundation for learning-based approaches.

3.4 GAN-Augmented CG

We train a Wasserstein GAN (WGAN) [2] on the heuristically generated patterns to learn the distribution of feasible columns. The generator maps latent noise to

pattern space, while the critic distinguishes real from generated patterns. During CG, we employ a warm start strategy in which the first t pricing iterations rely on GAN-based CG, after which we switch to exact pricing. Preliminary experiments indicated that, for larger instances, relying exclusively on GAN-generated columns would not guarantee convergence to the optimal solution. Although candidate columns are generated from a learned latent space, convergence to the LP optimum is preserved, since each generated column is evaluated using reduced costs computed from the dual variables of the RMP, thereby maintaining the standard coupling between the primal master problem and the pricing step. This hybrid approach aims to reduce the number of expensive pricing calls while maintaining solution quality. We evaluate against baselines using random sampling, greedy heuristics, and direct use of training data, from which we would randomly sample.

4 Computational Study

4.1 Objectives

The objective of the experimental evaluation is to assess whether generative models can serve as an effective warm-start mechanism in CG by reducing the number of expensive exact pricing calls required to reach LP optimality. The goal is not to replace exact pricing in terms of solution quality, but to evaluate to what extent learning-based CG can accelerate early convergence and improve downstream computational efficiency. Consequently, comparisons are normalized by the number of exact pricing calls, which dominate computational cost in the two-dimensional CSP. All methods are evaluated on randomly generated instances of the one- and two-dimensional CSP with fixed random seeds to ensure reproducibility. For the two-dimensional case, random instances of increasing scale are considered, ranging from small to extra-large, with the "largeness" depending on the number of unique item configurations to fit in the CSP. Warm-start strategies are tested for different warm-start lengths $t \in \{1, 3, 5, 10, 20\}$, after which the algorithm switches to exact pricing until convergence. Each configuration is evaluated over multiple independent runs to account for stochastic variability during training and sampling by the generative model, and all reported results are averaged accordingly. All experiments were implemented in Python. The GAN was implemented with `PyTorch`. The RMP and PP were solved with `Gurobi 13.0` via its `gurobipy` interface under an academic license. The code underlying this study will be made publicly available upon publication.

4.2 Early Convergence Analysis

We first analyze early convergence behavior by comparing objective values attained after the warm-start phase. The objective is to understand how effectively different warm-start lengths guide the RMP toward high-quality regions of the solution space at the beginning of the problem.

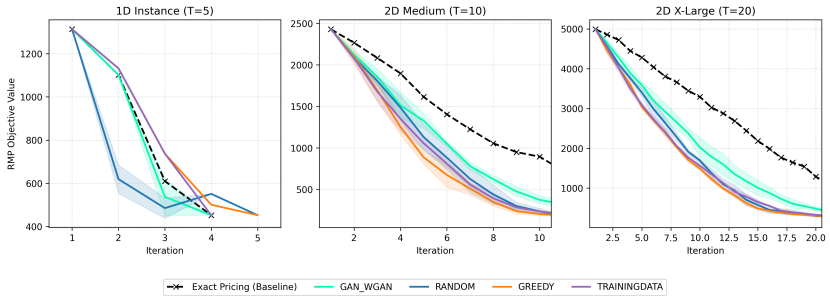

Fig. 1: Early convergence behavior for representative instances under different warm-start lengths.

Across 1D, 2D small, 2D medium, 2D large and 2D extra large instances, GAN-based warm starts consistently achieved better objective values during the initial iterations than exact pricing alone. Some examples can be observed in the $t = 5$ 1D early convergence plot 1, the $t = 10$ 2D medium early convergence plot 1 and the $t = 20$ 2D xlarge 1 . For the 1D and 2D small instance we can achieve LP-optimality with the GAN generated columns as sampling, but for larger instances, this could not be done within our previously defined sampling budget t.

This behavior suggests that generative models are effective at exploring early on, but struggle to produce increasingly improving columns as dual prices become more refined. Nevertheless, occasional large jumps in objective value indicate that the GAN is still capable of generating particularly strong patterns, that havent been already incorporated during previous iterations albeit less consistently than greedy heuristics.

4.3 Quality of GAN-Generated Columns

To contextualize the observed convergence behavior, we analyze the quality of GAN-generated columns for the two-dimensional CSP. Table 1 reports duplication, feasibility, and effective novelty metrics for the small and extra-large instances across different warm-start lengths. Metrics are calculated on raw batches of generated patterns prior to rejection sampling. Feasibility % denotes the proportion of patterns satisfying problem constraints (width limit or packing). Uniqueness % measures internal diversity as the ratio of distinct patterns within a single generated batch, while Duplicates % indicates the rate of redundancy mode collapse. In the 1D case, we observe a sharp drop in unique pattern generation. Rather than indicating mode collapse in the traditional adversarial sense, this likely reflects the exhaustion of the finite solution space inherent to this very small 1D instance. Since the pool of feasible cutting patterns is combinatorially small, the generator effectively memorizes and replicates the complete set of valid permutations found in the training phase ($n = 500$ training patterns), resulting in high redundancy during sampling. For the 2D-CSP instances, the

generative model produces a highly diverse and feasible set of patterns. Duplicate rates remain consistently below 20% for all warm-start lengths, indicating limited memorization of the training data. At the same time, feasibility rates exceed 80% on average, demonstrating that the GAN successfully learns the structural constraints imposed by guillotine cuts. Consequently, more than 80% of generated patterns are both feasible and unique, which explains why GAN-generated columns alone can be sufficient to drive early convergence and, in some cases, achieve LP optimality without invoking exact pricing. These findings suggest that the generative capacity of the GAN is robust to the dimensionality of the randomly generated 2D CSP problem instances. Provided that the combinatorial space is extensive enough to support diverse sampling, the model achieves a consistent yield of high-quality patterns. This stability across increasing instance sizes implies that the architecture captures the fundamental rules of the packing problem, allowing it to scale effectively without degrading in terms of feasibility or diversity.

4.4 Effect of Warm Starting on Pricing Convergence

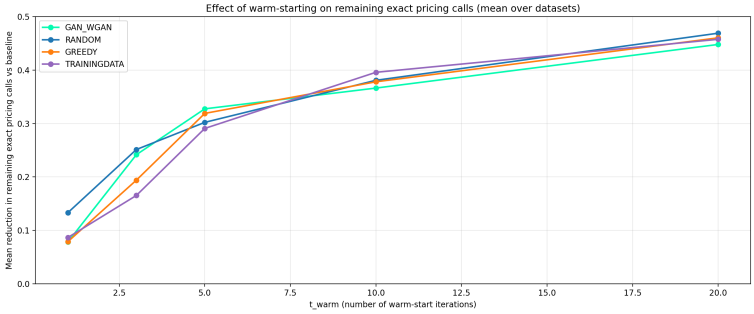

Fig. 2: Impact of warm-start length on solver efficiency

Figure 2 summarizes the mean relative reduction in remaining exact pricing calls achieved by different warm–start strategies as a function of the number of warm iterations.t_{warm}, averaged across all considered datasets, with the baseline defined as solving the RMP using only exact pricing. The figure shows a clear monotone trend for all methods: increasing t_{warm} consistently reduces the number of remaining exact pricing calls, with marginal gains diminishing as t_{warm} grows. For small warm budgets ($t_{\text{warm}} = 1, 3$), reductions are limited to roughly 8–25%, whereas for larger budgets ($t_{\text{warm}} = 20$) all methods achieve substantial reductions of approximately 45–47% on average. Importantly, the GAN-based warm start follows the same qualitative pattern as the heuristic baselines and yields comparable reductions, but does not exhibit a uniformly dominant advantage over them in terms of remaining exact pricing calls.

Table 1: Quality and diversity metrics of generated pattern batches

	$t = 1$			$t = 3$		
Instance	Feas.	Unique	Dupl.	Feas.	Unique	Dupl.
1D CSP	19.89	17.74	82.26	19.78	14.98	85.02
2D Small	82.91	82.08	17.92	80.06	79.09	20.91
2D Medium	82.01	80.36	19.64	82.07	79.78	20.22
2D Large	83.11	81.45	18.55	83.60	82.39	17.61
2D XLarge	83.07	81.66	18.34	83.36	81.79	18.21

	$t = 5$			$t = 10$		
Instance	Feas.	Unique	Dupl.	Feas.	Unique	Dupl.
1D CSP	20.66	14.42	85.58	20.39	14.02	85.98
2D Small	81.01	79.62	20.38	82.76	80.79	19.21
2D Medium	83.67	81.86	18.14	81.95	80.41	19.59
2D Large	82.88	80.76	19.24	81.17	79.55	20.45
2D XLarge	81.66	80.25	19.75	82.43	80.34	19.66

	$t = 20$		
Instance	Feas.	Unique	Dupl.
1D CSP	20.56	14.03	85.97
2D Small	83.21	81.16	18.84
2D Medium	83.19	80.95	19.05
2D Large	82.29	80.65	19.35
2D XLarge	82.80	81.10	18.90

5 Discussion

Overall, these initial empirical results show that learning-based CG is most effective during the early iterations of the algorithm, where exploratory sampling and column diversity reduce the number of subsequent exact pricing calls. For small instances, GAN-generated columns are often sufficient to drive early convergence and may even eliminate the need for exact pricing during the warm-start phase. As instance size increases, however, the combinatorial complexity of the pricing space limits the consistency with which the generative model produces improving columns, and exact pricing remains necessary to guarantee convergence to LP optimality. When inexpensive, problem-specific heuristics are available, these approaches remain marginally superior on average. Nevertheless, the GAN-based method consistently reproduces the same qualitative trends observed for heuristic warm starts, indicating the possibility of learning and generating feasible columns that could be beneficial for driving a faster convergence.

6 Conclusion and Future Work

This work demonstrates that synthetic data generation can be integrated into CG as a viable warm-start mechanism. While GAN-based CG is not really competitive with heuristical columns for the CSP, it consistently reduces the number of required exact pricing calls in early iterations and mirrors the convergence behavior of the heuristic approaches.

Several directions for future work arise from these findings. While the current results are promising, additional experiments on a broader range of randomly generated instances and problem classes are needed to provide stronger empirical support for the proposed method. Improving the quality of generated columns appears central, for example by conditioning generative models on structural features or dual information, or by incorporating contrastive learning signals that distinguish high-quality, low-quality, and infeasible patterns. Finally, a systematic analysis of generated columns in terms of reduced cost distributions and immediate dual improvement would provide deeper insight into the strengths and limitations of learning-based pricing strategies.

In summary, this study should be interpreted as a proof of concept highlighting the potential of synthetic data generation within CG frameworks. Although the approach does not outperform heuristics on the CSP, it opens promising avenues for more complex settings where pricing is substantially more expensive and the construction of effective heuristics is nontrivial.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abouelrous, A., Bliek, L., Gabor, A.F., Wu, Y., Zhang, Y.: Reinforcement learning for solving the pricing problem in column generation for routing. Operations Research Perspectives **15**, 100364 (2025)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International conference on machine learning. pp. 214–223. PMLR (2017)
3. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations Research **46**(3), 316–329 (1998)
4. Ben Amor, H., Desrosiers, J., Valério de Carvalho, J.M.: Dual-optimal inequalities for stabilized column generation. Operations Research **54**(3), 454–463 (2006)
5. Chi, C., Aboussalah, A., Khalil, E., Wang, J., Sherkat-Masoumi, Z.: A deep reinforcement learning framework for column generation. Advances in Neural Information Processing Systems **35**, 9633–9644 (2022)
6. Desrosiers, J., Lübbecke, M.E.: A primer in column generation. In: Column generation, pp. 1–32. Springer (2005)
7. Gilmore, P.C., Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. Operations research **13**(1), 94–120 (1965)
8. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem. Operations Research **9**(6), 849–859 (1961). https://doi.org/10.1287/opre.9.6.849

9. Gilmore, P.C., Gomory, R.E.: A linear programming approach to the cutting-stock problem, part ii. Operations Research **11**(6), 863–888 (1963). https://doi.org/10.1287/opre.11.6.863
10. Gondzio, J., González-Brevis, P., Munari, P.: New developments in the primal–dual column generation technique. European Journal of Operational Research **224**(1), 41–51 (2013)
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020)
12. Lübbecke, M.: Engine scheduling by column generation. Cuvillier Verlag (2001)
13. Lübbecke, M.E., Desrosiers, J.: Selected topics in column generation. Operations Research **53**(6), 1007–1023 (2005)
14. Morabit, M., Desaulniers, G., Lodi, A.: Machine-learning–based column selection for column generation. Transportation Science **55**(4), 815–831 (2021)
15. Morabit, M., Desaulniers, G., Lodi, A.: Machine-learning–based arc selection for constrained shortest path problems in column generation. INFORMS Journal on Optimization **5**(2), 191–210 (2023)
16. Shen, Y., Sun, Y., Li, X., Eberhard, A., Ernst, A.: Enhancing column generation by a machine-learning-based pricing heuristic for graph coloring. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 9926–9934 (2022)
17. Václavík, R., Novák, A., Šcha, P., Hanzálek, Z.: Accelerating the branch-and-price algorithm using machine learning. European Journal of Operational Research **271**(3), 1055–1069 (2018)
18. Wang, J., Zhang, Y., Xing, X., Zhan, Y., Chan, W.K.V., Tiwari, S.: A data-driven system for cooperative-bus route planning based on generative adversarial network and metric learning. Annals of Operations Research **339**(1), 427–453 (2024)
19. Wang, Q., Hao, Y., Zhang, J.: Generative inverse reinforcement learning for learning 2-opt heuristics without extrinsic rewards in routing problems. Journal of King Saud University-Computer and Information Sciences **35**(9), 101787 (2023)
20. Zhang, B., Yao, Y., Kan, H., Luo, W.: A gan-based genetic algorithm for solving the 3d bin packing problem. Scientific Reports **14**(1), 7775 (2024)