SCIENTIFIC OASIS

Journal of Operations Intelligence

Journal homepage: www.jopi-journal.org
eISSN: 3009-4267

Optimization of Stowage Strategy for Ship Loading Tasks

Zhuangle Yao[1], Lubin Wu[1], Meng Yuan[1,*] Zehui Wu[2], Shancheng Jiang[2], Zhen-Song Chen[3,*]

1 COSCO SHIPPING Specialized Carriers Co., Ltd., Guangzhou, China
2 School of Intelligent Systems Engineering, Sun Yat-Sen University, No. 135, Xingang Xi Road, Guangzhou, 510275, China
3 School of Civil Engineering, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Article history:
Received 14 May 2024
Received in revised form 30 June 2024
Accepted 21 July 2024
Available online 4 August 2024

Keywords:
Ship Stowage; Bin Packing Problem; Optimization Algorithm.

ABSTRACT

In recent years, the logistics industry has witnessed rapid growth, drawing considerable attention to pre-transport cargo loading as a focal point of research for scholars globally. However, the escalating complexity of actual transportation processes and the diversification of application scenarios have rendered traditional loading methodologies inadequate to meet the evolving demands of enterprises. To address this issue, this paper presents a solution to the loading problem of rectangular pulp bales in non-rectangular complex cargo holds, effectively meeting the demand for optimization algorithms and thereby conserving manpower and resources. The paper introduces a packing algorithm tailored to custom regions, which integrates the optimized Skyline algorithm with domain-search iterative strategies to achieve optimal loading outcomes. Furthermore, the efficacy of the algorithm is validated using real maritime data, demonstrating its robust performance and loading efficiency.

1. Introduction

In the freight transportation industry, including air and sea shipping, transporting a wide variety of goods is one of the most fundamental tasks. Driven by globalization and e-commerce, the impact of transportation on logistics, trade, and the economy is becoming increasingly significant, making the pre-transport cargo stowage process particularly crucial. In the loading area, workers face different loading zones and varying cargo daily. Relying solely on workers' experience for stowage can be burdensome and reduce efficiency. Designing a program that combines workers' stowage experience with optimization algorithms to generate better stowage plans under different conditions would alleviate workers' burden and improve stowage efficiency and utilization, ultimately reducing transportation costs and significantly enhancing industrial and transportation efficiency. Therefore, automating the generation of loading plans and researching optimization methods for related strategies has become an important research direction in this field.

* *Corresponding author.*
E-mail address: zschen@whu.edu.cn, yuan.meng@coscoshipping.com

https://doi.org/10.31181/jopi21202425

The core issue in this scenario is the bin packing problem (BPP), a complex discrete combinatorial optimization problem characterized by numerous local optima. BPP is non-differentiable, discontinuous, multidimensional, constrained, and highly nonlinear, qualifying as an NP-hard problem [1,2]. Given a set of items and several bins, BPP seeks a packing configuration that minimizes the number of bins used while maximizing utilization. Depending on whether the items to be packed are known in advance, BPP can be classified into two types: Online-BPP and Offline-BPP. Specifically, in Offline-BPP, all item information is known beforehand, whereas in Online-BPP, items arrive sequentially, with workers only aware of the current item and having no information about future items. A typical application of Online-BPP is stacking, such as immediately packing items transported by a conveyor belt onto a fixed-bottom pallet, where each pallet can hold dozens of items. In contrast, Offline-BPP is used in real-world scenarios like loading goods into containers for truck or ship transport, with each container possibly holding hundreds of items. This paper focuses on the two-dimensional Offline-BPP, aiming to pack all regular rectangular cargo into an irregular space while maximizing space utilization under various operational and maritime transport constraints.

In this study, we first model the bin packing scenario for ship stowage based on real-world application scenarios in maritime transport. We implemented the generation of ship stowage areas for different types of plate areas and designed a general framework for ship stowage areas to meet diverse needs, improving algorithm efficiency. Next, we designed a bin packing algorithm for non-rectangular areas based on the designed ship stowage areas. Initially, we generated an initial stowage plan using an optimized Skyline algorithm. Then, we optimized the initial plan with a stowage optimization iterative strategy based on local search and employed post-processing strategies for gap optimization. Finally, we validated the effectiveness of our algorithm using real ship data from the industry, visualized the stowage strategies, and evaluated each stage of the optimization algorithm using quantitative metrics derived from actual transport experience.

The rest of this paper is organized as follows. Section 2 summarizes the related works of this study. Section 3 shows the problem description. Section 4 describes the details of the proposed methodology. Section 5 shows the experiment results. Section 6 concludes the paper.

2. Related Literature

For the container loading problem in ship stowage, extensive research over the years has led scholars to propose numerous algorithms based on different strategies, primarily including exact algorithms and approximate algorithms.

2.1 Exact Algorithms

When addressing combinatorial optimization problems, exact algorithms are methods that guarantee finding the optimal solution. These algorithms typically consider all possible solutions within the entire solution space and employ a certain search strategy to evaluate and compare the quality of each solution individually. Exact algorithms can determine the optimal solution through exhaustive search or other deterministic techniques. Notable exact algorithms include branch-and-bound methods, tree-based search algorithms, and dynamic programming. Given that the bin packing problem is NP-hard, the application of exact algorithms is relatively limited. For instance, FleSzar [3] proposed a new mixed-integer programming model for the bin packing problem with conflicts and item fragmentation (BPPC-IF). Beasley [4] introduced a tree-based search algorithm in his study of the two-dimensional bin packing problem. Lesh [5] developed a hybrid algorithm combining classical branch-and-bound and pruning methods. Additionally, Wei [6] proposed a hybrid

algorithm for solving the two-dimensional bin packing problem, integrating dynamic programming and branch-and-bound techniques.

2.2 Approximate Algorithms

Although exact algorithms can provide precise optimal solutions, they may face computational complexity challenges when dealing with large-scale problems. For NP-hard problems, the time complexity of exact algorithms is typically exponential, making them impractical for large-scale applications. In such cases, approximate algorithms and heuristic methods are often employed as effective means to find near-optimal solutions. These approaches can yield solutions close to the optimal within a reasonable time frame and are widely utilized in practical applications.

The time complexity of approximate algorithms is generally polynomial, meaning that the time required is significantly reduced compared to exact algorithms when addressing large-scale problems. While approximate algorithms may not always guarantee finding the optimal solution, they can ensure the solutions are close to the optimal. Numerous approximate algorithms have been proposed for the bin packing problem. Notable examples include the Bottom-Left (BL) algorithm introduced by Baker [7], the improved Bottom-Left Fill (BLF) algorithm by Berkey et al. [8], the Best Fit algorithm by Burke et al. [9], and the Next Fit Decreasing Height (NFDH) and First Fit Decreasing Height (FFDH) algorithms by Coffman et al. [10].

Jylänki [11] provided a comprehensive comparison of various bin packing algorithms, highlighting the strengths and weaknesses of the Maximal Rectangles algorithm [12] and the Skyline algorithm [13]. Both algorithms are widely used heuristic methods for solving two-dimensional rectangular bin packing problems. They employ different strategies and techniques for rectangle placement, each with its unique advantages. The Maximal Rectangles algorithm is typically more efficient in space utilization, minimizing space wastage, particularly with the Best-Fit Maximal Rectangles variant. Conversely, the Skyline algorithm features lower computational complexity compared to the Maximal Rectangles algorithm, and it is easier to implement and adjust.

3 Problem Description

3.1 Bin packing Problem Description

The bin packing problem is a classic combinatorial optimization problem in which a set of items with varying sizes must be packed into a set of containers with limited capacity. The items must be completely packed into the containers, and the total volume of items in each container cannot exceed its capacity, while also satisfying certain constraints. Typically, the goal is either to minimize the number of containers used or to maximize the packing efficiency [14]. There are various variants of the bin packing problem, such as the Single Container Bin Packing Problem and the Multiple Container Bin Packing Problem. In the Single Container Bin Packing Problem, there is only one container, and the objective is to pack all items into the container as efficiently as possible. In the Multiple Container Bin Packing Problem, multiple containers are available, and the items must be distributed among these containers.

3.2 Problem Constraints

In the bin packing problem, different constraints are often set depending on the specific problem and situation. Similar problems can have different solution spaces due to differing constraints, resulting in vastly different solutions. The problem constraints include: (1) Weight constraints[15-17] (2) Loading priority constraints[18-21] (3) Position constraints[22-24].

When specifying a cargo hold for ship stowage, the captain designates the deck of the cargo hold, and the stowage area cannot exceed the boundaries of this deck. Due to the architectural design of the ship and fixed obstacles within the ship, the shapes of different cargo holds can vary significantly. Therefore, the actual stowage areas present different scenarios, as illustrated in Figure 1.

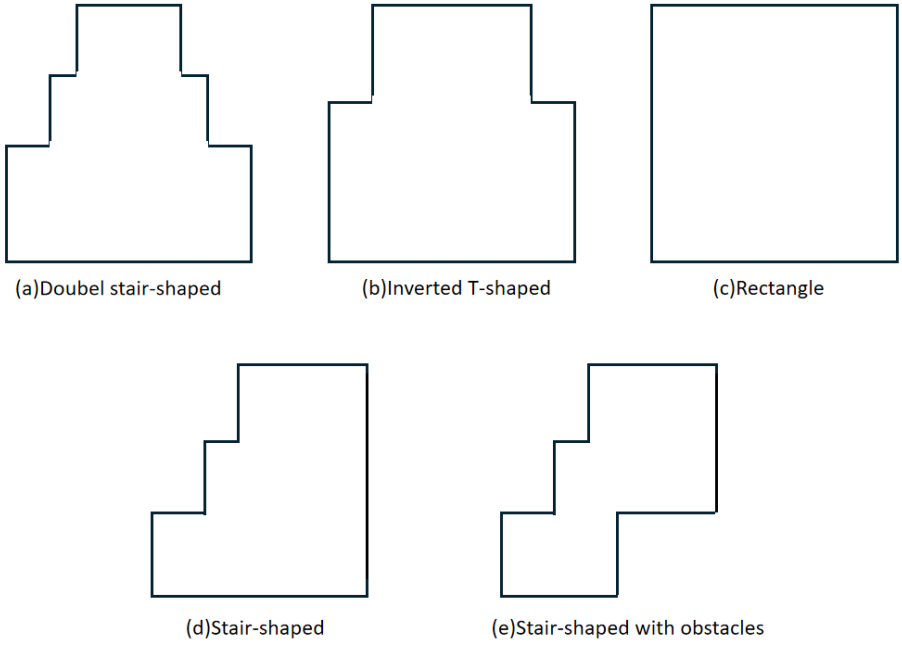

Fig. 1. Types of Cargo Hold Areas

Pulp Bales Center of Gravity Constraints: Due to the requirements of the stowage equipment, the placement of pulp bales is subject to certain restrictions. Assuming the mass distribution of the pulp bales is uniform, meaning the center of gravity is located at their geometric center, it is required that during stowage, a single pulp bale may intersect with the orange area shown in Figure 2, but its center of gravity must not fall within the orange area.

Gaps Between Pulp Bales Constraints: In the practical stowage of pulp bales, it is impossible to achieve a completely seamless stowage. Additionally, a buffer layer is necessary between pulp bales to protect them and prevent economic losses due to collisions. Therefore, the optimization of stowage plans for pulp bales must also meet the following two requirements:(1)Move the pulp bales as close to the warehouse walls as possible. (2) Arrange the gaps between pulp bales at the trisection points of the warehouse's long or short edges.

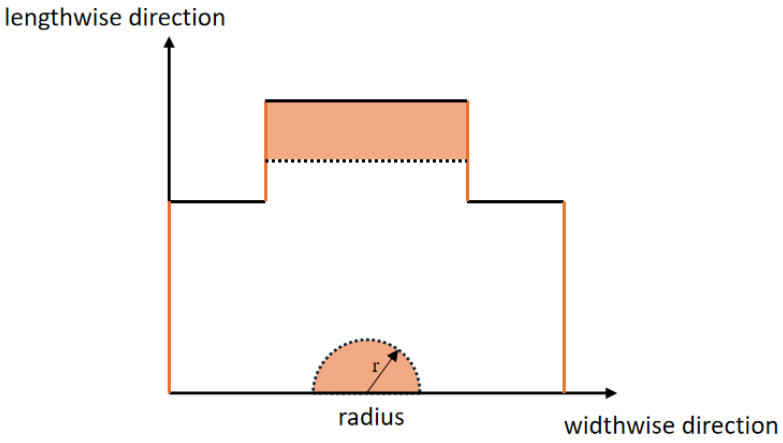

Fig. 2. Introduction to Blind Spot Information

4. Methods

4.1 Skyline Algorithm

The Skyline algorithm is a simple and fast packing method. Its core idea is to abstract the packing process using two concepts: Skyline and candidate positions. The Skyline refers to horizontal line segments from left to right within the cargo hold of a ship, representing the heights or highest points at various positions within the hold. The Skyline reflects the available space in the cargo hold in the horizontal direction. It is represented by a set of parallel lines indicating the upper boundaries of the topmost rectangles, representing the current packing pattern. Candidate positions represent the bottom-left and bottom-right coordinates of all Skylines.

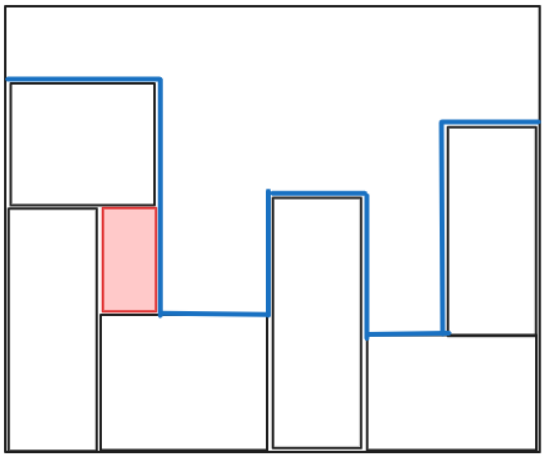

Fig. 3. Skyline Algorithm Diagram

The Skyline algorithm (Figure 3) maintains a data structure representing the skyline within the cargo hold to guide placement decisions for cargo items. When placing new cargo, the algorithm determines the optimal position within the ship's cargo hold based on the dimensions of the cargo and the height of the skyline, aiming to maximize the loading efficiency. After a rectangle is placed along the skyline, the skyline is cut where the rectangle is placed, effectively shortening the skyline based on the placement of the rectangle. The top edge of the rectangle then becomes a new skyline added to the list. When it becomes apparent that a rectangle cannot be placed on a particular skyline,

the skylines need to be merged. Specifically, this involves raising the skyline to the height of the other skylines and then merging multiple skylines into one.

The Skyline algorithm is widely used in ship stowage as it leverages skyline information within the cargo hold to provide efficient cargo placement solutions. It can assist shipping companies and freight agents in improving loading efficiency, reducing space wastage, and lowering transportation costs.

4.2 Generation of Loading Strategies Based on the Skyline Algorithm

Through iterations, one sequentially selects a combination of rectangular packages from the input list and adds them to the sequence. The number of iterations equals the specified length of the rectangular package sequence. During this process, to achieve the goal of minimizing the number of placed combination rectangular packages, priority is given to larger-sized combination rectangular packages. This is achieved by calling a probability distribution generation function, where larger-sized rectangular packages have a higher probability of being selected. This ensures that larger-sized combination rectangular packages are prioritized in the generated sequence. Additionally, considering the significant differences in the shapes and sizes of the ship's fore and aft holds compared to the midship hold, different probability distribution functions are used for the fore and aft holds and the midship hold.

For different ship stowage areas, based on the available rectangular package list within the current stowage area, this project generates a specified length of rectangular sequence by calling a probability distribution generation function for sequential loading attempts. Furthermore, for stowage areas with significant differences in shape and size, different probability distribution generation functions are designed and implemented.

Given the information on the stowage areas and the sequences of available rectangular packages, this project proposes an optimization algorithm based on the Skyline algorithm, considering the stowage area requirements and loading constraints. While the Skyline algorithm is suitable for the rectangular packing problem, in this project, the stowage space within the cargo hold includes various shapes such as inverted T-shaped and stair-shaped. Additionally, there are constraints such as the restrictions on the center of gravity of rectangular packages. Therefore, optimizations are made to adapt the Skyline algorithm to these requirements.

To improve loading efficiency and utilization of stowage areas, the project adopts a heuristic approach to minimize wasted area. This involves calculating the wasted area when placing a rectangle on all segments of the skyline list and selecting the segment with the smallest wasted area to place the currently traversed rectangle.

For blind spot loading, where the center of gravity of a rectangle cannot be placed within a blind spot, the project designs obstacle avoidance algorithms for different shaped blind spots, including circular blind spots, stair-shaped stowage areas.

For circular blind spots, obstacle avoidance is achieved by moving the center of gravity of the rectangle to the edge of the circular blind spot. For blind spots in stair-shaped stowage areas, they are considered as lower blind spots. The algorithm first loads the blind spot, then loads the remaining stowage space. If the center of gravity falls within the blind spot, it is replaced with a rectangle of appropriate size to avoid the blind spot.

Combining the Skyline algorithm with the above optimizations, the overall process is as follows (Figure 4):

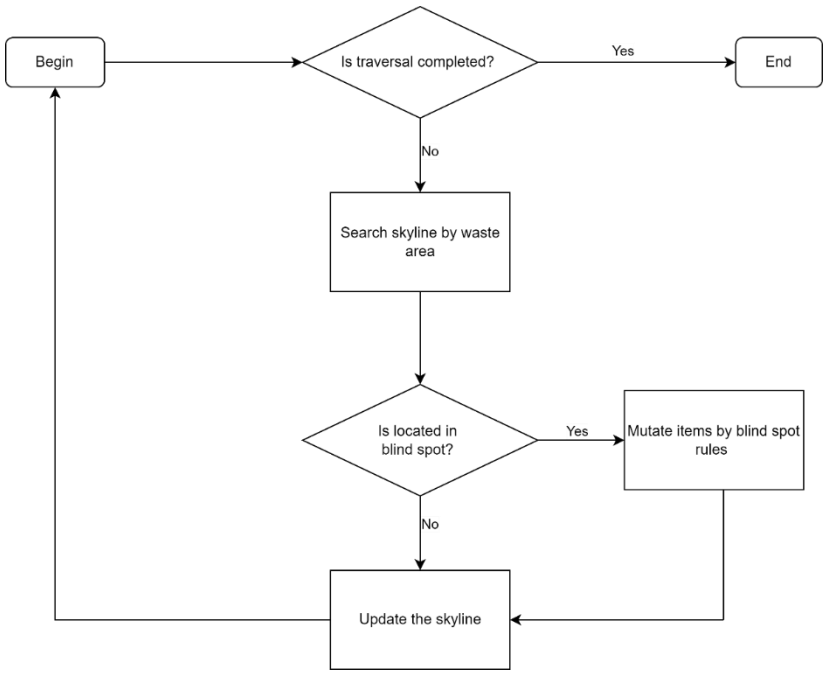

Fig. 4. Flowchart of the Loading Sequence Based on the Skyline Algorithm

4.3 Loading Iterative Optimization Strategy Based on Local Search

After optimizing based on the Skyline algorithm, an initial loading plan can be generated, but this plan is typically not optimal. Further optimization is needed for the initial plan, where the algorithm performs local search operations to find the optimal solution.

Local search is conducted by swapping, replacing, or other operations on the rectangles in the sequence of combined rectangular packs. When the number of cycles is less than 1/3 of the maximum number of cycles, random order swapping of pulp packs in the loading area is performed. When the number of cycles reaches between 1/3 of the maximum number of cycles and 2/3 of the maximum number of cycles, with a specific probability distribution, the pulp packs in the loading area are mutated into other sizes. When the number of cycles is between 2/3 of the maximum number of cycles and the maximum number of cycles, random mutations are performed with a uniform distribution.

To assess the quality of the solutions, the project has designed an evaluation function to score and compare the solutions before and after local search operations. Firstly, the number of basic rectangular packages loaded and the number of combined rectangular packages loaded are calculated before and after the local search operations.

For the solutions before and after the operations, they are ranked based on the criteria of having more basic rectangular packages and fewer combined rectangular packages. If the solution after the local search operation is better, the loading plan is updated; otherwise, the original loading plan is retained.

The evaluation function of the algorithm comprehensively considers the number of rectangular packages loaded, loading constraints, and algorithm complexity. It selects results with a higher number of loaded packages while meeting the loading constraints. This quantifies the loading information and explores more potential solution spaces, thus enhancing search efficiency. Through continuous iterative optimization, the algorithm gradually seeks the optimal solution.

4.4 Post-processing Strategy

The post-processing strategy involves further optimization after iterative iterations fail to achieve the desired loading results for certain hull types. Upon observation, it was noticed that for these suboptimal loading plans, the gaps between rectangles could be further optimized. Therefore, a post-processing algorithm based on "gap shifting + probing filling" was implemented to detect and attempt to merge unused gaps between rectangles.

Additionally, besides further optimizing the loading plan, the consideration of gap identification and adjustment aligns with practical requirements of the project. During the actual transportation of vessels, there is inevitably some degree of turbulence. Moreover, the cargo being loaded is not unilaterally distributed. If there are excessively large gaps in a layer of the loading plan, it could lead to instability in the center of gravity due to turbulence. This instability may cause the upper layer of cargo to tilt, resulting in economic losses during transportation. Adjusting the gaps to reduce their volume allows for the use of airbags to fill the gaps, preventing damage to the cargo.

Fig. 5. Gap example

The steps for gap (Figure 5) identification are as follows:

Firstly, determine whether these gaps are located between 1/3 and 2/3 of the ship's length. If this condition is not met, adjustments need to be made to the positions of the rectangular packages that form the gap, as follows:

(1) If the gap runs along lengthwise direction, check the x-coordinate of the gap to see if it falls within 1/3 to 2/3 of the ship's width. If the x-coordinate of the gap is less than 1/3 of the ship's width, move the rectangular packages to the right of the gap to the left until the gap is completely covered; if the x-coordinate of the gap is greater than 2/3 of the ship's width, move the rectangular packages to the left of the gap to the right until the gap is completely covered. After covering the gap, check if moving the rectangular packages will cause collisions between them. If no collisions occur, keep the move.

(2) If the gap runs along the widthwise direction, check the y-coordinate of the gap to see if it falls within 1/3 to 2/3 of the ship's length. If the y-coordinate of the gap is less than 1/3 of the ship's length, move the rectangular packages below the gap upwards until the gap is completely covered; if the y-coordinate of the gap is greater than 2/3 of the ship's length, move the rectangular packages above the gap downwards until the gap is completely covered. After covering the gap, check if moving the rectangular packages will cause collisions between them. If no collisions occur, keep the move.

These movements continue until all gaps are located between 1/3 and 2/3 of the ship's length. Stop the looping operation and return the final positions of the rectangular packages.

5 Experiment

5.1 Experimental Data Acquisition and Setup

In validating the design of the packing algorithm in this paper, the data used are based on actual ship data provided by the enterprise. The CAD files provided by the enterprise include the actual structure, shape, and coordinate data of the ship's various positions. To facilitate calling in the algorithm, this paper integrates the data in a structured manner based on the CAD files and enters the required data in JSON format, according to types such as side-passing board areas and cargo hold areas.

To specifically display the quantitative indicators of each stage packing strategy, this paper also designs a visualization tool to visualize the packing strategies of each algorithm. For the visualization tool, this paper mainly relies on Matplotlib, a GUI library within Python, consistent with the implementation of the algorithms in this paper. The visualization part uses Python 3, and the implementation tool is Visual Studio Code.

5.2 Results and Comparison

In the algorithm design based on Section 4, to demonstrate the optimization effects of each stage algorithm, this paper designs multiple quantitative indicators to evaluate each stage algorithm and verify the effectiveness of the optimization algorithms designed in this paper for loading plans. Quantitative indicators include the number of loaded rectangles, and the number of packings after packing. The number of loaded rectangles describes the number of cargoes that can be loaded in the same loading area, directly showing the loading effect of the algorithm. The number of packings after packing describes the number of operations required during the actual loading process, affecting the time consumption in actual loading. By comparing these indicators, the optimization effect of the algorithm can be effectively demonstrated.

5.2.1 Optimization algorithm results and comparison

As shown in Figure 6., with the increase in the number of iterations, the first decrease in the number of packed items occurs, indicating that the algorithm sacrifices some performance first to meet some hard constraints. After meeting the hard constraints, the number of packed items gradually increases, and at the same time, the number of packed rectangles also increases. After the number of packed items gradually stabilizes, the number of packed rectangles begins to decrease, and the loading plan gradually approaches optimal.

In summary, the optimization algorithm of local search not only ensures that the loading plan meets the hard constraints, but also balances the number of packings under the condition of maximizing the number of packed items, gradually approaching the optimal loading plan.

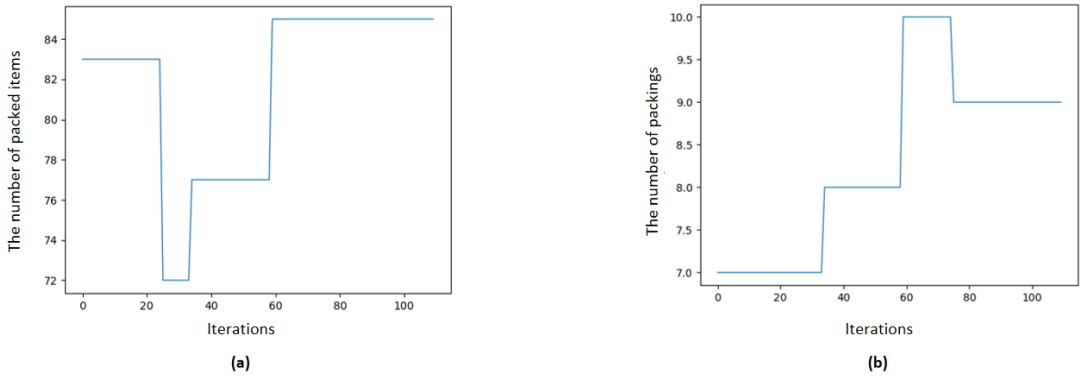

Fig. 6. Metric Changes During the Iterative Process of the Local Search Optimization Algorithm

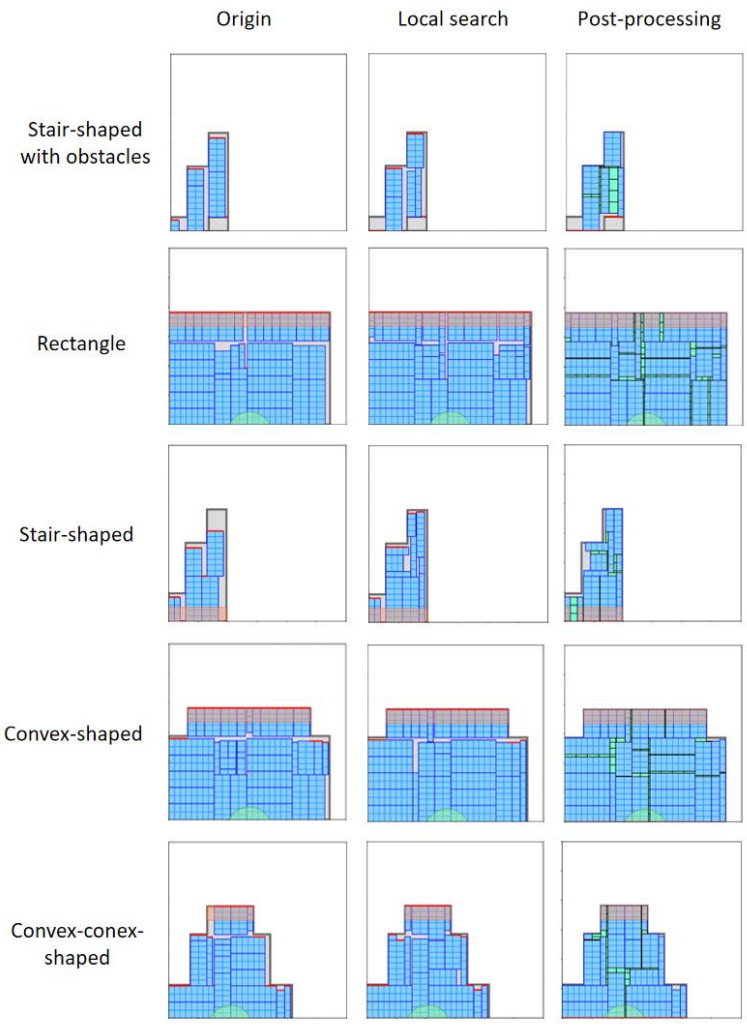

Fig. 7. Performance Verification Effect Graph of the Different Algorithm

5.2.2 Post-processing algorithm results and comparison

In the post-processing algorithm, adjustments are made to the positions of goods based on the optimized loading plan from the local search optimization algorithm, ensuring a relatively rational distribution of goods. As seen in the visual depiction of the loaded configurations in Figure 7, compared to the unprocessed loading plan, the post-processed loading plan achieves the following: (1)Quantitative indicators are at least not lower than those of the unprocessed plan. (2)The post-processed loading plan moves the generated gaps as close as possible to the middle of the loading area. Additionally, for gaps that can be split, they are separated, reducing the occurrence of large-area gaps.

6 Conclusion

This paper solves the problem of loading rectangular pulp bales in complex non-rectangular loading environments based on actual scenarios provided by enterprises in logistics transportation. It meets the enterprise's demand for loading optimization algorithms, saving manpower and resources. Firstly, this paper designed the generation of ship stowage areas to meet different requirements, implementing strategies for generating ship stowage areas for specified cargo holds

and bypass plates. Based on the similarity of the ship stowage areas generated for these two types of requirements, a general ship stowage area framework was established, enhancing the efficiency of subsequent algorithms. Secondly, this paper developed a bin packing algorithm tailored for custom stowage areas. Utilizing the general ship stowage area framework designed in Section 3, the Skyline algorithm was optimized to handle non-rectangular areas. An optimization strategy based on local search was then designed to refine the initially generated stowage plans in pursuit of the optimal solution. Following this, a post-processing strategy was employed on the optimized bin packing plans to adjust the positioning of rectangular cargo, ensuring that the gaps between items met the practical requirements of the transportation process. This not only optimized the stowage plans but also reduced potential losses during actual transport. Finally, to validate the effectiveness of the optimization algorithm, it was tested using real-world data from the industry. Quantitative metrics were devised based on practical transportation experience to evaluate the algorithm. The initial and optimized stowage plans were analyzed using these metrics, demonstrating the effectiveness of the optimization algorithm.

Author Contributions

Conceptualization, M.Y. and Z.Y.; methodology, L.W.; software, Z.W.; validation, Z.Y, and S.J.; formal analysis, Z.Y.; investigation, M.Y.; resources, M.Y.; writing—original draft preparation, Z.Y.; writing—review and editing, Z.W.; project administration, M.Y.; funding acquisition, M.Y. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data Availability Statement

Data will be made available on request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was not funded by any grant.

References

[1] Garey, M. R., & Johnson, D. S. (1981). Approximation algorithms for bin packing problems: A survey. In Analysis and design of algorithms in combinatorial optimization (pp. 147-172). Vienna: Springer Vienna. https://doi.org/10.31181/dmame181221015k.

[2] Garey, M. R., & Johnson, D. S. (1990). A Guide to the Theory of NP-Completeness. Computers and intractability, 37-79. https://doi.org/10.1137/1024022

[3] Fleszar, K. (2022). A MILP model and two heuristics for the bin packing problem with conflicts and item fragmentation. European Journal of Operational Research, 303(1), 37-53. https://doi.org/10.1016/j.ejor.2022.02.014

[4] Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. Management science, 44(3), 388-399. https://doi.org/10.1287/mnsc.44.3.388

[5] Lesh, N., Marks, J., McMahon, A., & Mitzenmacher, M. (2004). Exhaustive approaches to 2D rectangular perfect packings. Information Processing Letters, 90(1), 7-14. https://doi.org/10.1016/j.ipl.2004.01.006

[6] Wei, L., Oon, W. C., Zhu, W., & Lim, A. (2013). A goal-driven approach to the 2D bin packing and variable-sized bin packing problems. European Journal of Operational Research, 224(1), 110-121. https://doi.org/10.1016/j.ejor.2012.08.005
[7] Baker, B. S., Coffman, Jr, E. G., & Rivest, R. L. (1980). Orthogonal packings in two dimensions. SIAM Journal on computing, 9(4), 846-855. https://doi.org/10.1137/0209064
[8] Berkey, J. O., & Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. Journal of the operational research society, 38, 423-429. https://doi.org/10.2307/2582731
[9] Burke, E. K., Kendall, G., & Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. Operations Research, 52(4), 655-671. https://doi.org/10.1287/opre.1040.0109
[10] Coffman, Jr, E. G., Garey, M. R., Johnson, D. S., & Tarjan, R. E. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing, 9(4), 808-826. https://doi.org/10.1137/0209062
[11] Jylänki, J. (2010). A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing. retrived from http://clb. demon. fi/files/RectangleBinPack. pdf.
[12] Naamad, A., Lee, D. T., & Hsu, W. L. (1984). On the maximum empty rectangle problem. Discrete Applied Mathematics, 8(3), 267-277. https://doi.org/10.1016/0166-218X(84)90124-0
[13] Wei, L., Zhang, D., & Chen, Q. (2009). A least wasted first heuristic algorithm for the rectangular packing problem. Computers & Operations Research, 36(5), 1608-1614. https://doi.org/10.1016/j.cor.2008.03.004
[14] Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading–A state-of-the-art review. European Journal of Operational Research, 229(1), 1-20. https://doi.org/10.1016/j.ejor.2012.12.006
[15] Gehring, H. (1997). A genetic algorithm for solving the container loading problem. International transactions in operational research, 4(5-6), 401-418. https://doi.org/10.1016/S0969-6016(97)00033-6
[16] Terno, J., Scheithauer, G., Sommerweiß, U., & Riehme, J. (2000). An efficient approach for the multi-pallet loading problem. European Journal of Operational Research, 123(2), 372-381. https://doi.org/10.1016/S0377-2217(99)00263-5
[17] Chan, F. T., Bhagwat, R., Kumar, N., Tiwari, M. K., & Lam, P. (2006). Development of a decision support system for air-cargo pallets loading problem: A case study. Expert Systems with Applications, 31(3), 472-485. https://doi.org/10.1016/j.eswa.2005.09.057
[18] Bischoff, E. E. (2006). Three-dimensional packing of items with limited load bearing strength. European Journal of Operational Research, 168(3), 952-966. https://doi.org/10.1016/j.ejor.2004.04.037
[19] Bischoff, E. E., & Ratcliff, M. S. W. (1995). Loading multiple pallets. Journal of the Operational Research Society, 46, 1322-1336. https://doi.org/10.1057/jors.1995.181
[20] Ren, J., Tian, Y., & Sawaragi, T. (2011). A tree search method for the container loading problem with shipment priority. European Journal of Operational Research, 214(3), 526-535. https://doi.org/10.1016/j.ejor.2011.04.025
[21] Bortfeldt, A., & Gehring, H. (1999). Zur Behandlung von Restriktionen bei der Stauraumoptimierung am Beispiel eines genetischen Algorithmus für das Containerbeladeproblem. Logistik Management: Intelligente I+ K Technologien, 83-100. https://doi.org/10.1007/978-3-642-60184-2_8
[22] Hodgson, T. J. (1982). A combined approach to the pallet loading problem. IIE Transactions, 14(3), 175-182. https://doi.org/10.1080/05695558208975057
[23] Haessler, R. W., & Talbot, F. B. (1990). Load planning for shipments of low density products. European Journal of Operational Research, 44(2), 289-299. https://doi.org/10.1016/0377-2217(90)90364-H
[24] Bortfeldt, A., & Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research, 131(1), 143-161. https://doi.org/10.1016/S0377-2217(00)00055-2