



Article

Optimization of the One-Size-Fits-All Layout Problem Based on Preparing Material for Steel Bridges

Zhikui Dong ^{1,*} , Chunjiang Liu ¹, Yongkuan Sun ¹, Xuedong Li ¹, Kai Zhang ¹ and Yunhong Jiang ^{2,*} 

¹ School of Mechanical Engineering, Yanshan University, Hebei Street 438, Qinhuangdao 066004, China; 13402431900@163.com (C.L.); 18931350254@163.com (Y.S.); lixuedong2023@163.com (X.L.); 15934236568@163.com (K.Z.)

² Department of Applied Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, UK

* Correspondence: 13513357441@163.com (Z.D.); yunhong.jiang@northumbria.ac.uk (Y.J.); Tel.: +86-135-1335-7441 (Z.D.)

Abstract: Before the construction of a bridge begins, workers arrange the necessary parts and then cut and process them. The quality of the cutting layout directly affects the material utilization rate and the efficiency of the subsequent processes. During bridge construction, an intelligent part layout can improve work efficiency, save time, and reduce the labor intensity and production costs for the company. In this study, we studied a layout optimization algorithm, focusing on rectangular parts in the material preparation process. A mathematical model for the rectangular layout problem was constructed, and a hybrid genetic whale optimization algorithm is proposed that is a combination of the whale optimization algorithm and the genetic algorithm. Based on the “one size fits all” layout strategy, the materials are divided into strips, which are further divided into stacks, serving as the positioning strategy to determine the positional relationships of the parts. Test cases and actual engineering data were used to compare the layouts generated using different algorithms. The results show that the genetic whale algorithm proposed in this paper results in a high utilization rate and is highly effective.

Keywords: preparation of steel bridge materials; rectangular parts; one-size-fits-all layout problems



Citation: Dong, Z.; Liu, C.; Sun, Y.; Li, X.; Zhang, K.; Jiang, Y.

Optimization of the One-Size-Fits-All Layout Problem Based on Preparing Material for Steel Bridges. *Appl. Sci.* **2024**, *14*, 4891. <https://doi.org/10.3390/app14114891>

Academic Editors: José António Correia and Paulo Santos

Received: 24 March 2024

Revised: 21 May 2024

Accepted: 29 May 2024

Published: 5 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

When determining the layout of steel bridge components, the goal is to achieve maximum material utilization by adjusting the positions of the parts while considering the processing techniques. The primary objectives are to optimize the use of steel, reduce material waste, and lower production costs. The layout problem falls under the category of NP-complete (nondeterministic polynomial complete) problems, which are complex combinatorial optimization problems. Solutions to these problems typically cannot be obtained directly through calculations but are instead derived through indirect “guesswork,” hence the term nondeterministic. There is no known polynomial-time algorithm that can solve all forms of layout problems. Instead, heuristic algorithms, approximation algorithms, or other optimization techniques are usually employed to find approximate solutions. These algorithms include linear programming methods, genetic algorithms, tabu search algorithms, local search algorithms, and ant colony algorithms [1].

Research on two-dimensional layout problems is mainly split into two areas. On the one hand, there are sequencing algorithms, which determine the order in which parts are placed. Common sequencing algorithms include genetic algorithms, ant colony algorithms, simulated annealing algorithms, and particle swarm algorithms. Each of these algorithms has its own strengths and weaknesses when solving sequencing problems. Nowadays, when addressing sequencing problems, researchers often combine two types of algorithms, such as the genetic simulated annealing algorithm or the hybrid genetic-variable neighborhood search algorithm, to develop a more efficient and effective algorithm. On the

other hand, there is the positioning strategy. Positioning is a method of establishing the position on the motherboard of parts that have been determined in order. Commonly used positioning strategies include the bottom-left algorithm, the lowest horizontal line algorithm, and the critical polygon algorithm [2]. Research on layout problems mainly involves sheet metal cutting, which has important significance in theory and in practice.

The layout problem can be traced back to 1939, but it did not significantly gain attention until the 1960s. Gilmore and Gomory [3–6] conducted research on one-dimensional cutting stock problems and two-dimensional rectangular cutting problems, proposing the use of linear programming methods to solve these layout issues. In 1988, to better study layout problems, scholars founded the Europe Special Interest Group on Cutting and Packing (ESICUP) at an international conference in Paris. This organization focuses specifically on researching and discussing layout problems. Over the years, ESICUP has collected a significant amount of research findings from scholars and research institutions, providing crucial support for subsequent studies. Furthermore, with the rapid development of computer technology, scholars have begun using intelligent algorithms to optimize the layout of parts.

In layout problems, positioning strategies and sequencing algorithms are crucial. In terms of the positioning strategy, Art [7] first proposed a method called “left-side docking”, which involves moving the laid-out parts to the left as much as possible. Thus, the parts are concentrated as much as possible to reduce the blank area. Later, Dowsland [8] and others improved the left-side docking method. Based on the left-side docking method, if there is a cavity in the part, the part is placed into the cavity to improve resource utilization. Baker [9] and others proposed the BL algorithm. Simply put, its aim is to position the parts as far left and down as possible. This algorithm is easy to implement, has low time costs, and is widely used in rectangular layout design.

In terms of sequencing algorithms, since different layout sequences can significantly impact the layout results, many scholars have conducted research on these algorithms. Delchambre [10] applied the genetic algorithm to tackle the layout problem, while Błazewicz [11] utilized the tabu search algorithm. Leung [12] and others introduced the local search algorithm to address layout challenges; Solimanpur [13] employed the ant colony algorithm to solve the layout problem with dynamic constraints. These algorithms were applied to the two-dimensional layout problem and achieved good results.

The two-dimensional layout problem is a combinatorial optimization problem. Currently, there is no known algorithm that can find the solution to the problem in polynomial time. The solution process is relatively complex and has significant research importance. The methods for solving combinatorial optimization problems include exact solutions and heuristic algorithms. Although exact solutions often yield better results, the time-consuming nature of their use escalates exponentially with the complexity of the problem. This drawback undermines their practical applicability in production scenarios.

The layout problem studied in this study is based on the layout problem of preparing material for steel bridges. During the production process, a substantial number of parts need to be laid out; when preparing steel bridge parts, not just one, but many components are required to produce each part. Since the motherboard for the layout is a steel plate, it takes a long time to cut the parts. Therefore, when choosing a layout, we should arrange the parts as neatly as possible while maintaining a high overall utilization rate to facilitate part cutting. Thus, based on actual production needs, in this study, a heuristic algorithm was used to study the two-dimensional layout problem. By adopting a “one size fits all” layout strategy and utilizing a genetic whale algorithm, we aimed to find the optimal solution to this problem.

2. Mathematical Model of Rectangular Layout Problem

The layout problem can be divided into two categories according to the motherboard type. The first is the boxing layout, in which the layout is determined after determining the shape and size of the motherboard. If the parts exceed the boundary of the motherboard,

they are placed on the next one. Another type of layout problem is the band layout, where the width of the motherboard is predetermined but its height is unlimited. The final height (after all the parts are arranged) is the height of the motherboard. In the following, we use the band layout as an example to establish a mathematical model for the rectangular layout problem.

The two-dimensional rectangular layout can be described as follows: n rectangles of different sizes, denoted as $\{\pi_1, \pi_2, \dots, \pi_n\}$, are placed onto a rectangular sheet S without overlapping, where π represents the parts. The width of the sheet is W , and the height is unlimited. Under the constraint conditions, the height H of the sheet that is used is minimized to maximize the utilization rate of the sheet.

The following constraints must be met during the placement of rectangular parts onto the motherboard:

- (1) Different rectangular parts π_i and π_j must not overlap each other;
- (2) The rectangular piece π_i must be inside the motherboard S ;
- (3) The rectangular piece π_i can be rotated 90° .

This can be explained more in depth as follows: Assume that the width of the rectangular mother plate S is the x -axis, the height is the y -axis, and the plate's lower left corner is the origin of the x, y coordinate system. Suppose that $\{\pi_1, \pi_2, \dots, \pi_n\}$ is a set of n rectangular pieces, and $\pi_i = \{x_i, y_i, w_i, h_i, \theta_i\}$, $i \in \{1, 2, \dots, n\}$, where x_i represents the horizontal coordinate of the bottom left corner of the part, y_i represents the vertical coordinate of the bottom left corner of the part, w_i represents the width of the part, and h_i represents the height of the part. θ_i denotes the rotation angle of the part, and π_i is the rotation angle. To minimize gaps and overlaps between parts and maximize material utilization, rotation angles of 0° and 90° were selected, and $\theta_i = \{0, 2/\pi\}$. Therefore, the mathematical model of the two-dimensional rectangular layout problem can be established as follows:

$$f = \max \frac{\sum_{i=1}^n (w_i \times h_i)}{W \times H} \tag{1}$$

$$x_i + w_i \leq W, i = 1, 2, \dots, n \tag{2}$$

$$\text{s.t.} \begin{cases} x_i \geq 0, y_i \geq 0 \\ x_i + w_i \leq x_j \text{ or } x_j + w_j \leq x_i \\ y_i + h_i \leq y_j \text{ or } y_j + h_j \leq y_i \\ i \neq j, \text{ and } i, j = 1, 2, \dots, n \end{cases} \tag{3}$$

$$\theta_i = \{0, \pi/2\}, i = 1, 2, \dots, n \tag{4}$$

Equation (1) represents the optimization objective function for a rectangular layout, which is the ratio of the area occupied by the parts on the master plate after placement. Equation (2) indicates that the parts should not exceed the boundary range of the master plate when placed. Equation (3) represents that there should be no overlapping between the placed parts. Equation (4) indicates that the parts can be rotated by 90° .

3. Layout Strategy

Since the parts for bridge materials are cut from steel plates, the layout needs to facilitate the subsequent cutting processes. Therefore, in this study, the "one size fits all" layout strategy was applied. The basic idea of this algorithm is to cut through the entire plate along the layout path from one end of the plate to the other during each cutting process. This method is called "one size fits all" because it uses the same direction at each cutting stage to ensure that the entire plate is cut while meeting the process requirements. We call cutting in the same direction a stage.

In the actual production process, in order to improve the efficiency, the cutting process is usually divided into three or four stages. To achieve this, the three-stage precise nesting and the three-stage non-precise nesting methods are commonly used as cutting strategies. Figures 1 and 2 illustrate these two nesting methods. In these diagrams, rectangles with numbers represent parts, where the same number indicates parts of the same size. The vertical lines outside the rectangles represent cuts, with different numbers denoting different cutting stages. The arrows of different colors represent different stages. During cutting, we refer to cuts in the same direction as one stage. In the three-stage precise layout method, square pieces can be cut to an accurate size within three stages. This method ensures a high production efficiency while meeting specification requirements as closely as possible. In the non-precise method, an additional fourth stage of cutting may be required for some square pieces to meet the specification requirements.

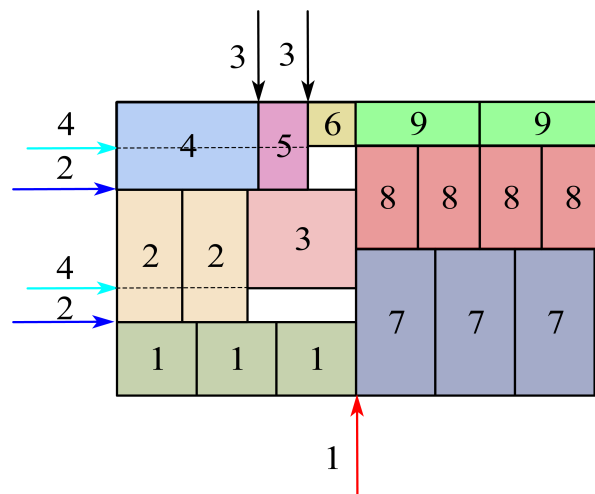


Figure 1. Three-stage non-precise layout.

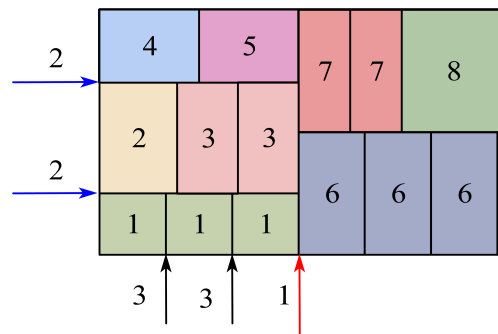


Figure 2. Three-stage precise layout.

The “one size fits all” layout strategy not only considers the process requirements and production efficiency but also focuses on meeting product specifications. When preparing materials for steel bridges, manufacturing companies choose an appropriate cutting stage and layout method according to specific requirements to achieve the best production results. This integrated approach to process and specification provides greater flexibility in the production process to accommodate different types of products and their requirements.

Due to the difference in the number of stages, different authors use different names for each stage. Figure 3 shows specific details of the key stage modules. During the actual cutting process, the initial cut can be made along either the long or short side. Figure 3 illustrates an example perpendicular to one of the sides.

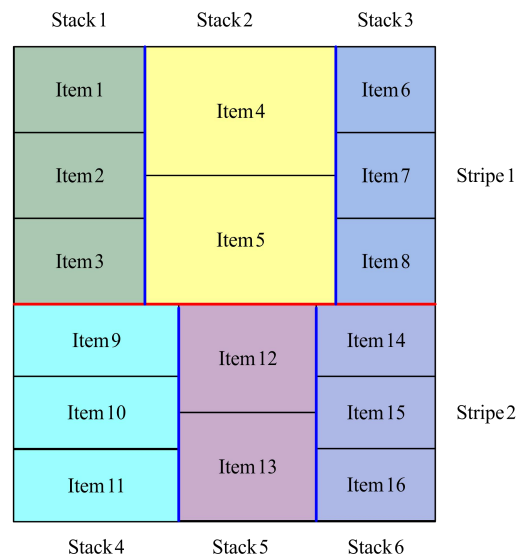


Figure 3. Definitions of different cutting stages.

In taking the three stages of the cutting process as an example (see Figure 3), the first stage involves horizontal cutting to generate modules that we call stripes. The steel plate is divided into Stripe1 and Stripe2 along the red lines. The second stage involves vertical cutting to generate modules called stacks. For example, Stripe1 is further cut along the blue lines to form Stack1, Stack2, and Stack3. The third stage involves horizontal cutting to generate modules we call items. For instance, Stack1 is further cut along the black lines to form Item1, Item2, and Item3. The cutting process in these three stages facilitates the subsequent cutting of parts for preparing material for steel bridges.

The layout used in this study was a three-stage non-precision layout, which can be described as follows: n rectangular parts $\{Item_1, Item_2, \dots, Item_n\}$ of different sizes, with a quantity $\{m_1, m_2, \dots, m_n\}$, are placed without overlapping onto a rectangular motherboard S with width W and an unlimited height. Under the condition that the “one size fits all” constraint is met, the height H of the board is minimized, thereby maximizing the utilization rate of the motherboard.

The following constraints must be met during the placement of rectangular parts onto the motherboard:

- (1) Different rectangular parts $Item_i$ and $Item_j$ must not overlap each other;
- (2) The rectangular piece $Item_i$ must be inside the motherboard S ;
- (3) The highest point of the rectangular part $Item_i$ cannot exceed the height of the stripe where it is located;
- (4) The width of the rectangular piece $Item_i$ cannot exceed the width of the stack;
- (5) Parts of the same type in the same stripe must have the same rotation angle.

Assume that the width direction of the rectangular motherboard S is the x -axis, the height direction is the y -axis, the lower left corner of the board is the origin of the x, y coordinate system, and the motherboard S is divided into multiple stripes. $\{Item_1, Item_2, \dots, Item_n\}$ indicates n rectangular parts, where their arrangement has been optimized; they only need to be arranged in sequence. $Item_i = \{x_i, y_i, w_i, h_i, \theta_i, m_i\}$, $i \in \{1, 2, \dots, n\}$, where (x_i, y_i) represents the coordinates of the lower left corner of $Item_i$; (w_i, h_i) represents the width and height of $Item_i$; θ_i is the rotation angle of $Item_i$, $\theta_i = \{0, \pi/2\}$; and m_i is the number of $Item_i$. The specific process is as follows:

- (1) Place the first $Item_1$ in the bottom left corner of Stripe1; the width of the first stack, Stack1, in Stripe1 is the same as the width of $Item_1$, and the height of Stack1 is equal to the height of Stripe1. Then, the remaining $m_1 - 1$ parts, denoted as $Item_1$, are sequentially placed into Stack1. As shown in Figure 4, during the queuing process,

when the remaining height of Stack1 cannot accommodate Item₁, then Item₁ will be queued into the next stack Stack2, and the width of Stack2 is the width of Item₁.

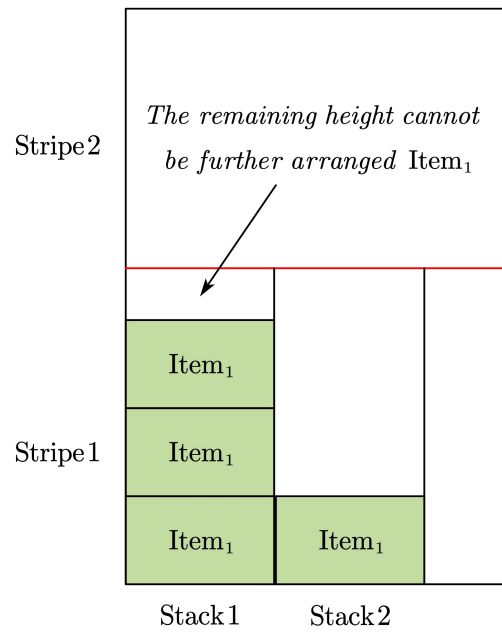


Figure 4. Item1 cannot be queued into Stack1.

- (2) As shown in Figure 5, if all parts of Item₁ are arranged without exceeding the height of Stack1, Item₂ will be placed there. First, it is determined whether the width of Item₂ is greater than the width of Stack1. If the width of Item₂ is not greater than the width of Stack1, then Item₂ will be placed in Stack1. Similarly, when the remaining height of Stack1 cannot accommodate Item₂, then Item₂ will be placed in the next stack, Stack2, and the width of Stack2 is the width of Item₂. If the width of Item₂ is greater than the width of Stack1, then Item₂ is arranged in Stack2, and the width of Stack2 is the width of Item₂.

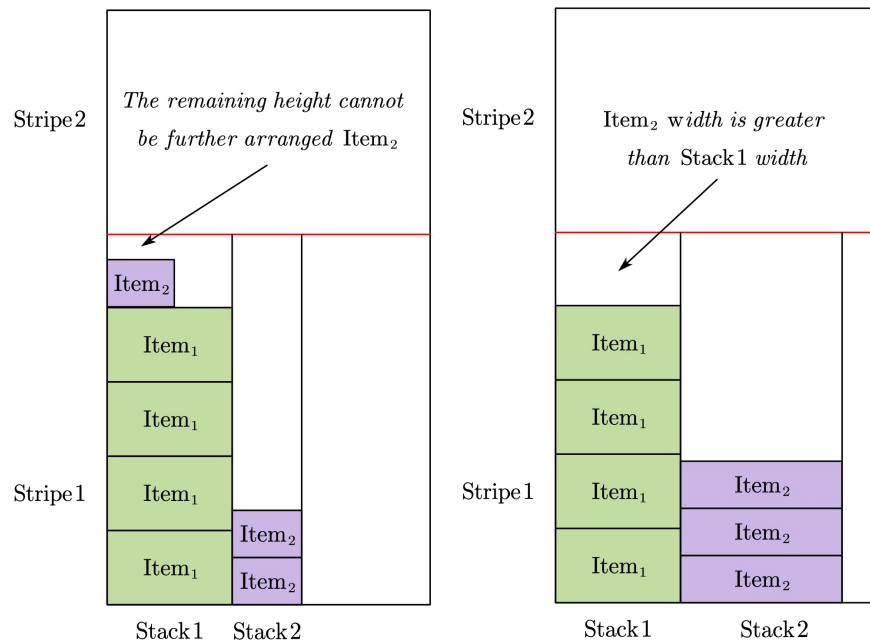


Figure 5. Parts are arranged into Stack1.

- (3) As shown in Figure 6, parts are arranged in sequence according to the above two steps. When the remaining width of Stripe1 cannot accommodate more parts, the parts are arranged into Stripe2, and the height of Stripe1 is updated. The height of Stripe1 is the highest point in Stripe1. All the parts are arranged according to the above steps.

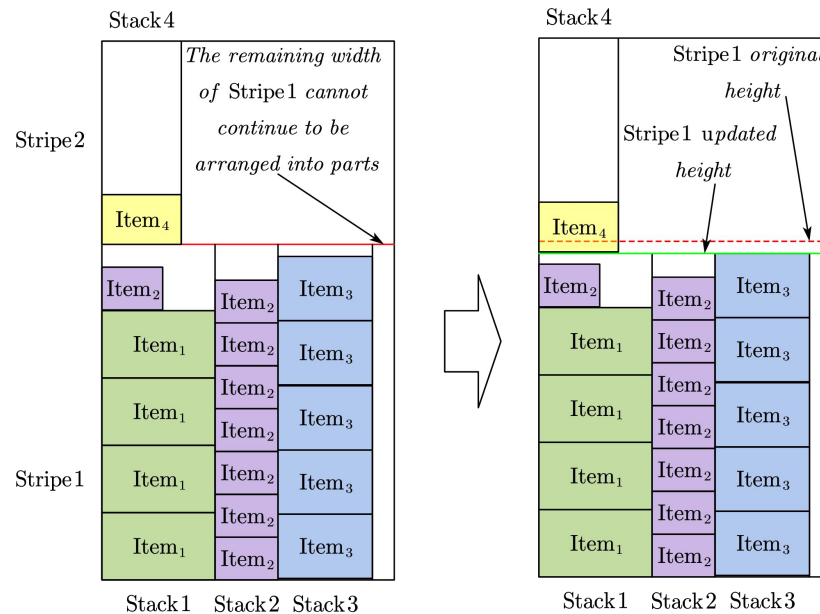


Figure 6. Stripe1 parts are filled.

4. Hybrid Genetic Algorithm

(1) Basic principles of the genetic whale algorithm

The genetic whale algorithm proposed in this article combines the genetic algorithm (GA) with the whale optimization algorithm. Its fundamental principle is utilizing the genetic algorithm (GA) as the outer loop and integrating the whale optimization algorithm (WOA) into the GA. This design aims to make full use of the global search capability of the GA and the local search capability of the WOA to avoid falling into the local optimal solution during the search process. This enables the genetic whale algorithm to more effectively uncover the optimal solution on a global scale and to fully exploit the global search capabilities of the GA. During the entire algorithm operation, the GA not only dominates the outer loop but also incorporates the WOA, so the two work together to significantly improve the efficiency of the final solution. Therefore, the genetic whale algorithm is initiated from a randomly generated or a customized initial population. In each cycle, the individuals of the population are selected and crossed, the whale hunting process is then added, and finally, the mutation process is performed; the above is then repeated. A fixed number of iterations is used while monitoring the convergence of the objective function and the diversity of the population. The loop is terminated when any condition is met to obtain the optimal solution. The basic steps of the genetic whale algorithm are shown in Figure 7.

(2) Initialize the population

In this algorithm, the quality of the initial population directly affects its search performance. The initial population should have a certain diversity and should also meet the constraints of the problem. On the basis of conventional layout constraints, as shown in Figure 8, this study introduced “stripe” as a height-limiting factor, employing a random generation method. In the population initialization stage, the stripe height is randomly generated after the part sequence is randomly generated. The stripe height is determined using a random generation range based on the size of the parts and the width of the plate in the actual problem. The sequence of parts and the height of the stripe together form an individual. The number of stripes should be slightly larger to ensure that all parts can be arranged. As shown in Figure 8, the numerical values in the sequence of previously placed

parts represent the part numbers. A positive value indicates that the part’s rotation angle is 0°, while a negative value indicates that the part’s rotation angle is 90°. The numerical values in the sequence of stripe heights represent the heights of each stripe.

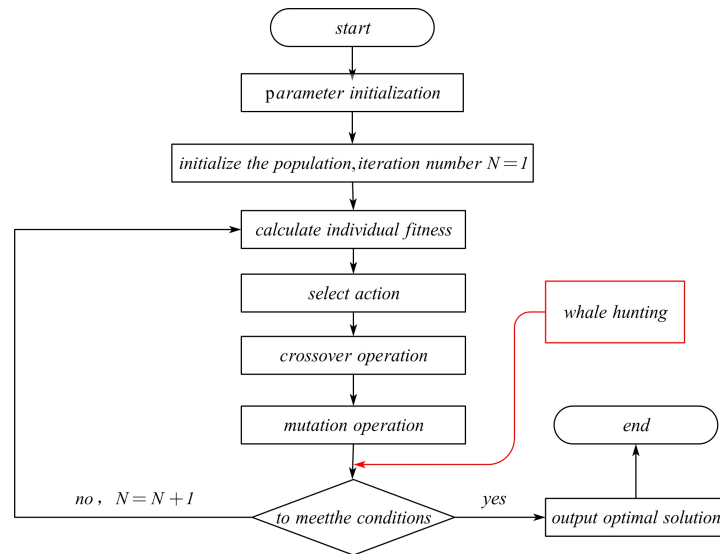


Figure 7. Genetic algorithm flow chart.

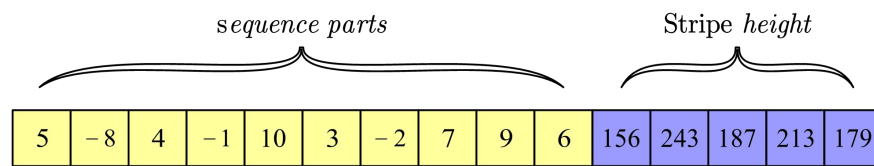


Figure 8. Stripe height generation.

The initialized stripe height cannot be the final stripe height. Just as the parts are arranged in the sequence, the height of the stripe needs to be optimized through the crossover and mutation of the algorithm.

(3) Fitness function calculation

In the layout problem studied in this study, the fitness was the final utilization rate of the motherboard. The design of the fitness function is as follows:

$$fitness = \frac{AreaSum}{W \times L} \tag{5}$$

Here, *AreaSum* represents the sum of the areas of all discharged parts, *W* is the width of the motherboard, and *L* is the final length after all parts are arranged on the motherboard. The larger the *fitness*, the higher the utilization rate and the better the layout effect. Since the width *W* of the motherboard is a fixed value, we can also regard *L* as the adaptation value. The smaller the *L*, the higher the adaptation value.

(4) Select operation

This study used the roulette selection method to select individuals. In this process, the cumulative fitness value of all individuals is first calculated (the cumulative fitness value of the *i*th individual is $F_i = \sum_{n=1}^i f_i$), and then a random number *rand* is generated. When $F_{i-1} < rand \leq F_i$, the *i*th individual is selected, and the probability of each individual being selected is p_i .

$$p_i = \frac{f_i}{\sum_{n=1}^n f_i} (0 < i \leq n) \tag{6}$$

(5) Crossover operation

The two-point crossover method was selected here. Two crossover points are selected based on the crossover probability. The crossover points of the two parent chromosomes are the same, and the genes at the two crossover points of the two parent chromosomes are swapped. When the number of parts is N and the crossover probability is P ($0 < P < 1$), an integer a is randomly generated as the first intersection point $0 < a < N - P * N$, and the other intersection point b is $b = a + P * N$. Since the nodes of the chromosomes in this article represent the order in which the parts are arranged, the sequence numbers of the two nodes cannot be the same, so during the crossover process, each sequence number needs to be exchanged according to the corresponding position of the other chromosome. Only the stripe heights at the intersection points need to be exchanged in the individual during the crossover stage. This process is illustrated in Figure 9, where parts in the input sequence are swapped according to the blue arrows, exchanging the positions of part numbers. The heights in the stripe height sequence are swapped at the crossover positions.

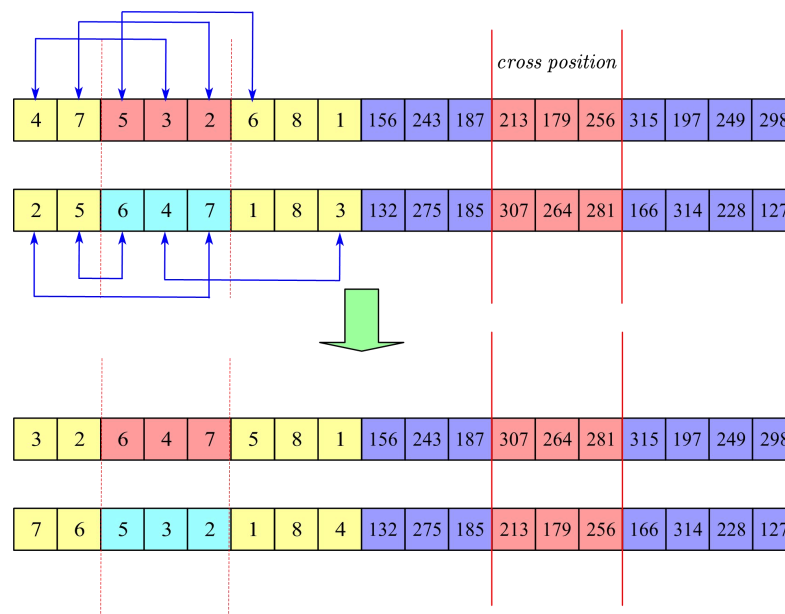


Figure 9. Schematic of the crossover operation.

(6) Mutation operation

As shown in Figure 10, in the problem studied in this study, since the encoding values in the part input sequence represent the part numbers, each part needs to be placed onto the master plate. Therefore, the gene values cannot be repeated. We chose to randomly exchange two gene positions to perform the mutation operation. In the stripe height sequence, a random gene value is altered.

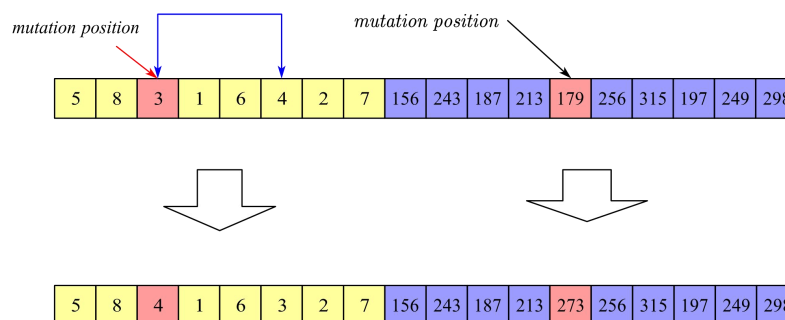


Figure 10. Schematic of the mutation operation.

(7) Whale hunting

In the whale hunting stage, some ordinary individuals need to be moved closer to the optimal individuals, and some positions of ordinary individuals need to be changed into the corresponding positions of the optimal individuals, as shown in Figure 11.

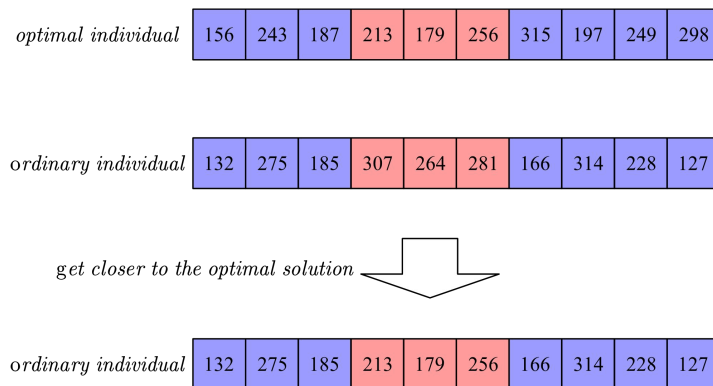


Figure 11. Optimization of the whale hunting process.

5. Experiment

To further validate the proposed algorithm’s effectiveness in solving real-world layout problems and to enhance the algorithm’s performance, we used the test case from reference [14] as Experiment 1. This case includes eight types of parts, totaling 131 pieces, and the width of the master plate was 2000 cm. The part information is shown in Table 1.

Table 1. Case data set.

Part Types	Length (cm)	Width (cm)	Quantity
1	120	90	30
2	150	100	30
3	280	120	28
4	250	150	20
5	60	50	10
6	150	50	5
7	220	180	4
8	55	80	4

In reference [14], three different algorithms were tested and compared using this case study. These three algorithms are the maximal rectangle algorithm, greedy algorithm, and simulated annealing algorithm. To evaluate the results of this experiment, the test results of our algorithm were compared with those of the algorithms in reference [14]. Table 2 shows the test results of each algorithm.

Table 2. Results of experiment for each algorithm.

Layout Results of Different Algorithms	Maximal Rectangle Algorithm	Simulated Annealing Algorithm	Greedy Algorithm	Simulated Annealing Algorithm
Utilization Rate (%)	93.39	83.17	94.36	98.12
Time (S)	0.398	43.860	0.378	17.259

As shown in Table 2, the layout generation time of the algorithm in this study is longer than those of the maximal rectangle algorithm and the greedy algorithm in reference [14], but shorter than that of the simulated annealing algorithm. However, the layout utilization rate is 98.12%, which is higher than those of the other three algorithms.

To further validate the effectiveness of the algorithm in solving real-world layout problems during the preparation of material for steel bridges, we conducted Experiment 2 using a set of real preparation data from an enterprise. There are 14 types of parts, comprising 266 pieces in total. The detailed data are shown in Table 3.

Table 3. Actual engineering data set.

Part Name	Width (mm)	Long (mm)	Quantity
E58-T5	1900	450	2
E56-T1	650	300	28
E98-P1	650	360	24
THL1-N8a	384	640	4
A55-T2	350	450	20
A58-T1	300	650	18
TPL1-T1	280	330	16
TPL2-P4	280	240	4
TPL1-8-T1	280	160	16
A55-T1	250	650	22
THL1-N8	150	640	8
A55-P7	80	270	34
A55-N10	750	590	40
THL12B-N8a	750	640	30

To evaluate the performance of the genetic whale algorithm, experiments were conducted using this algorithm, and the results were compared with the results from using the whale optimization algorithm (WOA) and genetic algorithm (GA). Each experiment was repeated 10 times, and the experimental data are shown in Table 4.

Table 4. Results of experiments for each algorithm.

Number of Experiments	Utilization Rate (%)		
	Genetic Whale Algorithm	WOA	GA
1	97.84	94.48	93.76
2	96.37	95.34	94.56
3	97.41	94.86	95.62
4	97.65	93.57	92.39
5	96.22	93.98	94.24
6	96.86	95.43	92.79
7	99.18	92.98	93.18
8	97.15	96.48	95.75
9	98.55	94.68	94.33
10	97.46	95.49	94.93
Aaverage value	97.45	95.12	94.16

From Table 4, it can be observed that the average utilization rate of the genetic whale algorithm is 97.45%, which is 2.33% and 3.29% higher than the average utilization rate of the whale optimization algorithm (WOA) and genetic algorithm (GA), respectively. Moreover, in the 10 experiments, the highest utilization rate for the genetic whale algorithm was 99.18%, which is sufficient to prove its effectiveness. Figure 12 shows the optimal layout results.

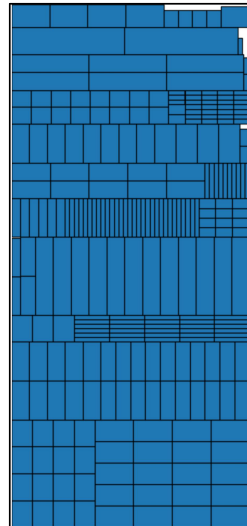


Figure 12. Sample arrangement results of Experiment 2.

6. Conclusions

In this study, we delved into the fundamental theory of the two-dimensional layout problem, developed a mathematical model for the rectangular layout problem, explored a strategy to solve the layout problem, and considered the impact of the height limit. We integrated the genetic algorithm with the whale optimization algorithm to address layout problems, generating the genetic whale algorithm, and introduced a coding method to optimize part sequencing on the motherboard. According to the experimental data, the average utilization rate of the genetic whale algorithm is 97.45%, which is 2.33% and 3.29% higher than the average utilization rate of the WOA and GA, respectively. Compared with the other two algorithms, it can obtain better results. Our algorithm's highest utilization rate is 99.18%, demonstrating that it can effectively solve the layout problem under the studied constraints. The genetic whale algorithm generates better quality solutions than the genetic and whale algorithms.

In employing advanced layout algorithms and fully automated, intelligent equipment, the production efficiency and precision of steel cutting and material preparation can be enhanced, significantly reducing steel wastage, lowering material costs, and improving project efficiency. Through comparative experiments, the effectiveness and practicality of the algorithm proposed in this paper were demonstrated, providing valuable insights for future implementations of the automated preparation of materials for steel bridges.

Author Contributions: Conceptualization, C.L. and Y.S.; methodology, Y.S. and Z.D.; software, K.Z. and C.L.; validation, Y.S., Z.D., and X.L.; formal analysis, C.L.; investigation, C.L. and Y.S.; resources, Y.J.; data curation, Z.D.; writing—original draft preparation, Z.D. and Y.S.; writing—review and editing, Y.J.; visualization, X.L. and C.L.; supervision, Y.S.; project administration, Y.J.; funding acquisition, K.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and analysed during the current study are not publicly available due to signing a confidentiality agreement with Party A, but are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Davis, L. Job shop scheduling with genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*; Psychology Press: London, UK, 2014; pp. 136–140.
2. Li, W. Genetic simulated annealing hybrid solution algorithm for rectangular part layout problem. *Forg. Technol.* **2021**, *46*, 70–76.
3. Gilmore, P.C.; Gomory, R.E. A linear programming approach to the cutting-stock problem. *Oper. Res.* **1961**, *9*, 849–859. [[CrossRef](#)]
4. Gilmore, P.C.; Gomory, R.E. A linear programming approach to the cutting stock problem—Part II. *Oper. Res.* **1963**, *11*, 863–888. [[CrossRef](#)]
5. Gilmore, P.C.; Gomory, R.E. Multistage cutting stock problems of two and more dimensions. *Oper. Res.* **1965**, *13*, 94–120. [[CrossRef](#)]
6. Gilmore, P.C.; Gomory, R.E. The theory and computation of knapsack functions. *Oper. Res.* **1966**, *14*, 1045–1074. [[CrossRef](#)]
7. Art, R.C., Jr.; Gomory, R.E. An Approach to the Two Dimensional Irregular Cutting Stock Problem. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1966.
8. Dowsland, K.A.; Dowsland, W.B. *Working Paper EBMS/13*; European Business Management School: Swansea, UK, 1993.
9. Baker, B.S.; Coffman, E.G., Jr.; Rivest, R.L. Orthogonal packings in two dimensions. *SIAM J. Comput.* **1980**, *9*, 846–855. [[CrossRef](#)]
10. Falkenauer, E.; Delchambre, A. A genetic algorithm for bin packing and line balancing. *ICRA* **1992**, 1186–1192.
11. Błażewicz, J.; Hawryluk, P.; Walkowiak, R. Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Ann. Oper. Res.* **1993**, *41*, 313–325. [[CrossRef](#)]
12. Leung, S.; Lin, Y.; Zhang, D. Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Comput. Oper. Res.* **2012**, *39*, 678–686. [[CrossRef](#)]
13. Solimanpur, M.; Vrat, P.; Shankar, R. Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing. *Eur. J. Oper. Res.* **2004**, *157*, 592–606. [[CrossRef](#)]
14. Hong, Z. Research and Implementation of Two-Dimensional Sheet Layout Algorithms for Intelligent Factory Applications. Master’s Thesis, Beijing Jiaotong University, Beijing, China, 2022.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.