**The report of project 5：**

**Strip Packing**

**Written by Group 6**

**Summary:** In this experiment,the strip packing problem, we implemented the Next-fit decreasing-height(NFDH), First-fit decreasing-height (FFDH) and Sleator's algorithm, also, an evolutionary version of it. We also built some data sets to explore the influence of different input data on algorithm running time and approximation ratio.Therefore, we generated (uniform distribution) random, width scale factor sequence, length scale factor sequence, all square input and other test data sets between 10-10000 to explore the influence on each algorithm index.Finally, we also find some existing paper data sets for testing and comparison.

**Chapter 1 introduction:**

The strip packing problem is a 2-dimensional geometric minimization problem[[1]](#footnote-0).To simplify the problem, let's just use rectangles as a reference. In the problem, we will give a series of rectangles, their respective lengths and widths (they may be not the same size) and a container with the maximum width it can hold (the height of the container is not limited); Our task is to find a way to stack rectangles in such a way that containers use the lowest height, which is also known as the strip packing problem.

The question itself is abstract in the field of engineering, given the premise of a material, whether there is a way to maximize the use of the same material, that is, to waste the least amount of scraps.

In addition, this problem was first studied in 1980[1]. This is an NP-hard problem, and it can be shown that the approximation ratio of this problem is not less than 3/2 under the premise that P is not equal to NP; However, the best approximation ratio of polynomial algorithm we can achieve at present is only (5/3 + ε)[2], so there may still be a large space for research on this problem.

**chapter 2 analysis of algorithm:**

（需要填充，另外结果分析的地方仅作辅助吧，如果准备展示的时候发现有问题，都可以改）

**chapter 3 detailed description of our dataset:**

In this section, we take a more detailed look at the reference and generation patterns of our dataset.

The first is the data set generated by ourselves. We mainly adopt five data set generation methods to explore the factors that may affect the algorithm performance and operation results.

For the first two generation methods, we assume that the generated width of 90% matrix is only between 1 and 11, and the generated width of the other 10% matrix is highly uncertain and can be distributed in any integer value between 1 and maximum width. The following is the specific method of generating data.

First, (uniform distribution) random generation. In this generation mode, we use random functions to directly generate (width, length) pairs for testing, The randomly generated test data of different sizes can be used for comparing the running time performance of various algorithms.

Secondly, we also explore some special input sequences, and add some special restrictions to them, which may have a great impact on the running results of some algorithms. Therefore, we directly generate square sequences with equal width and length, and test the response performance of different algorithms to them.

Thirdly and fourthly, we consider that the occasional occurrence of large width or height rectangles may have a significant impact on algorithm performance (as the approximation method used in this experiment often comes from the largest rectangle in the worst case). Therefore, we consider introducing corresponding width and height factors for generating large rectangles to explore potential constraints. We set the large width rectangle generation factor and the high height rectangle generation factor equally from 0.1 to 1.0, respectively, to generate a large-scale dataset for testing.

Fifthly, considering the generated dataset mentioned above, we cannot manually determine the size of its optimal solution, so the test data can only be compared between different algorithms; But in the fifth generation method, we used a simple method of randomly cutting large rectangles into small rectangles, ensuring the premise of knowing the optimal solution, which can better analyze the gap between the algorithm results and the optimal solution.

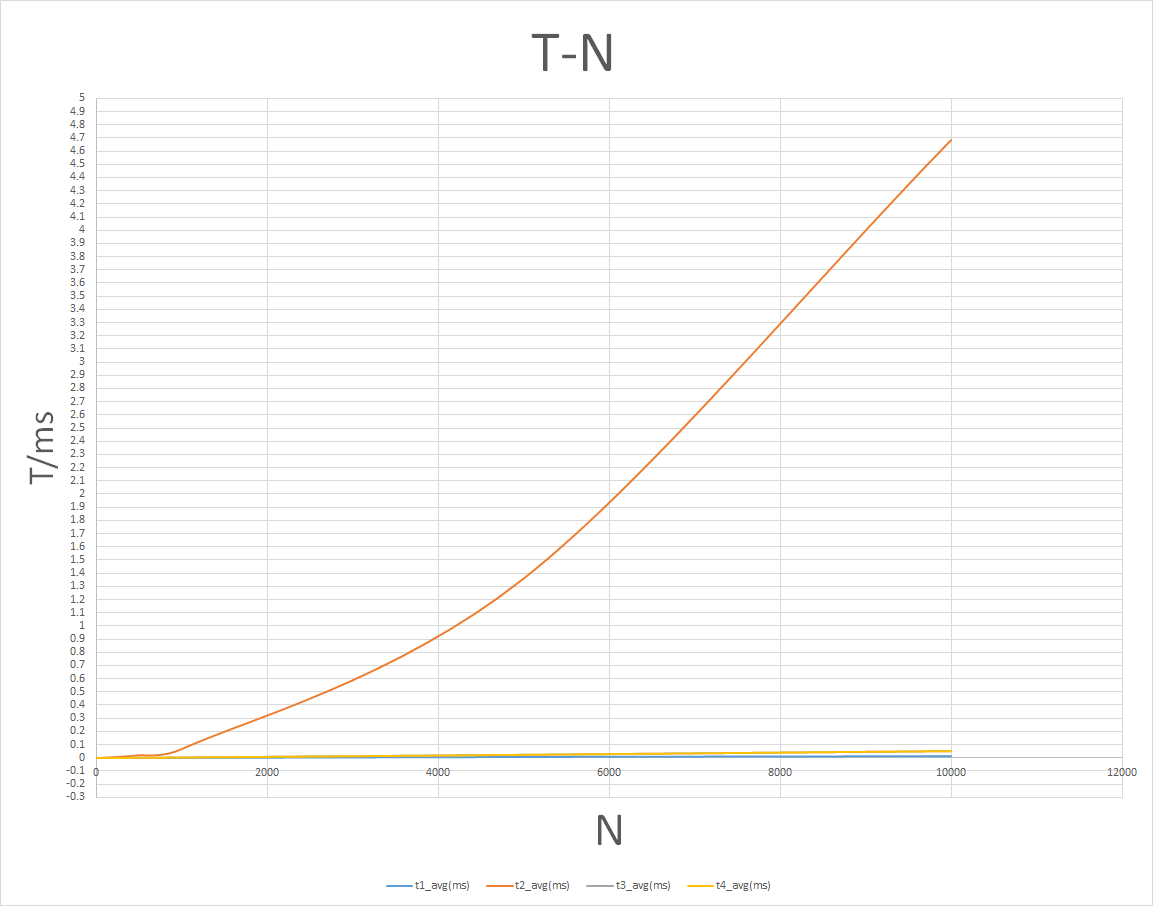
Finally, we also found some existing data sets in the paper for testing[[2]](#footnote-1), not only to test the correctness of our algorithm implementation, but also to compare and explore the possible defects of the algorithm under the input of some data sets.

Now let us look at the test results, The analysis of test data is presented in Chapter 5.

**chapter 4 comparison of these algorithms:**

4.1 Line chart on running time performance

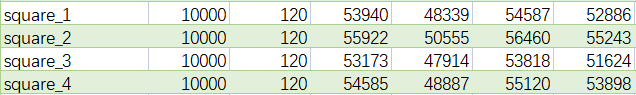
We generated a series of test datasets with a number of inputs ranging from 10 to 10000, each of which corresponds to three sets. We ran these datasets using different algorithms, averaged the results of the same number of inputs, and plotted the running time of the algorithms under different numbers of inputs. We can obtain the following results.



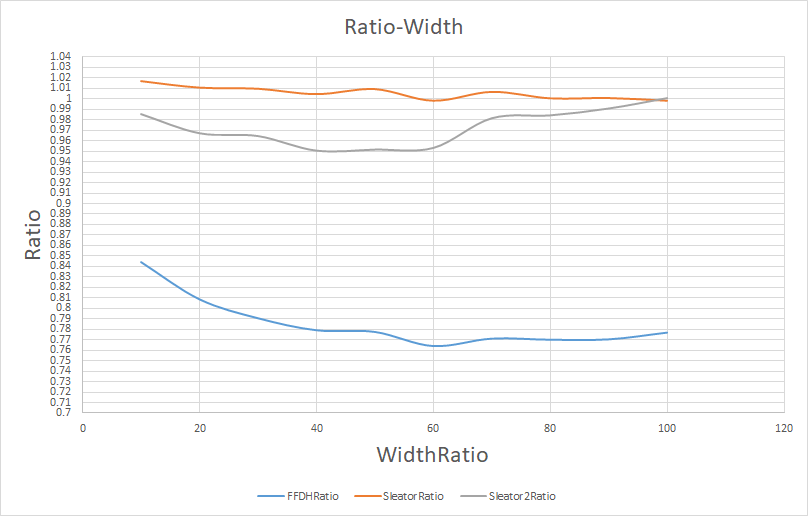
4.2 Result of full square input

We randomly generated four sets of square input sequences, with 10000 inputs and a total container width of 120 assumed.

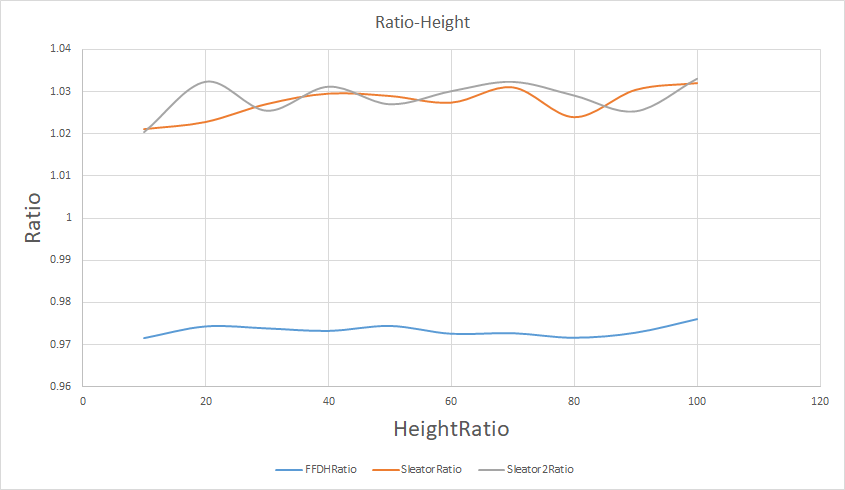
The seven columns of data in the table are the test file name, total number of inputs, total width size, NFDH algorithm calculation results, FFDH algorithm calculation results, the calculation results of the Sleator algorithm and those of our improved Sleator algorithm.



4.3 Width Generation Factor and Height Generation Factor Results

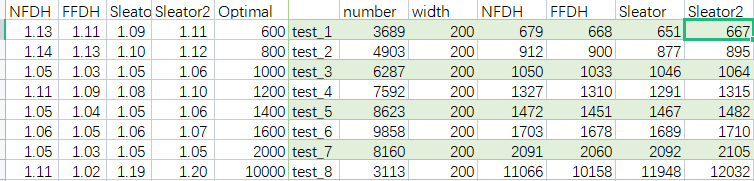
As introduced earlier, we take the 0.1-1.0 equidistant values for the height and width generation factors of large rectangles, and generate test datasets with the maximum input size for each. After running the algorithm, we can obtain the following results.

（大矩形宽度生成因子结果图）

(大矩形高度生成因子结果图）

4.4 Rectangular cutting generation

For the dataset generated by cutting large rectangles, we used large rectangles such as 200 \* 600,200 \* 800,200 \* 1000 (width represented by the front and height represented by the back) and cut them into different small rectangular datasets for testing. And we can obtain the following results(The specific header is shown in the figure).

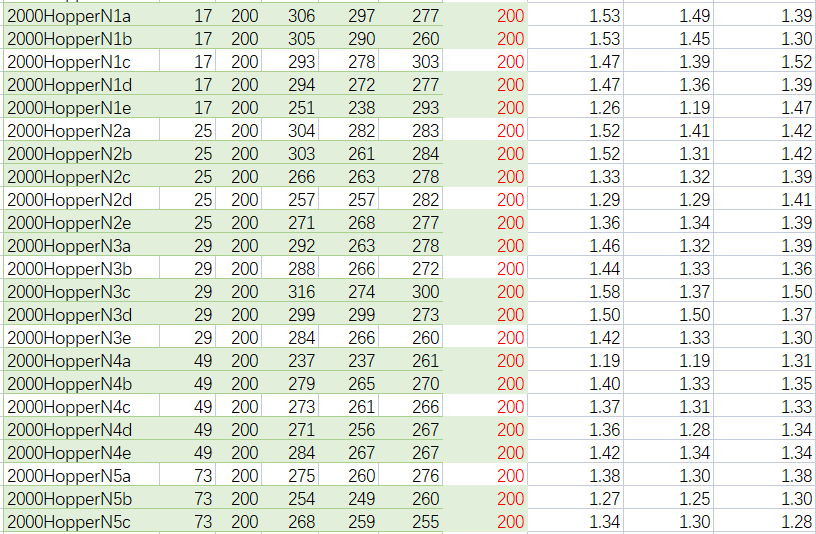


4.5 Paper data citation test

For the test data in this section, we have cited experimental data from professors such as Christofides and Whitlock[3], Bengtsson[4], Hopper and Turton. Due to space constraints, we will only compare the experimental results in a column. The results are shown below.

Firstly, we need to clarify that this is only part of Professor Hooper's testing data from that year.

Each column of data includes the file name of the test data, the total number of inputs, the total width of the container, the output results of the four algorithm mappings (in the same order as before), the number of optimal solutions(the red column), and the ratio relationship between each algorithm and the optimal solution (calculated in order).

****

**chapter 5 The result analysis:**

Correspondingly, we also analyzed the results one by one in the order of the dataset.

The first is a comparison graph of the performance of randomly generated sequences on algorithm runtime.The orange curve represents the FFDH algorithm, the blue curve represents the NFDH algorithm, and the yellow curve represents the sleator algorithm; From the graph, we can clearly see that with the expansion of the dataset, the time consumption of NFDH and sleator algorithms has not significantly improved; On the contrary, the FFDH algorithm has a significant improvement in time consumption compared to other algorithms, which is also related to the algorithm details that the FFDH algorithm needs to re traverse every time an element is added.

Next, we will analyze the square input sequence. From the experimental results, it can be seen that although there is a significant difference in absolute values between different algorithms, in terms of relative proportion, the difference between optimal and worst is not more than 10%. Overall, the results are within an acceptable range. From the specific data, we can see that the FFDH algorithm has always maintained a relatively good result calculation, and in the input case of a full square sequence, the sleator algorithm is dwarfed and even weaker than the NFDH algorithm.

We will continue to analyze the impact of the width and height generation factors of large rectangles on the algorithm results. Firstly, it should be noted that the drawing of these two result graphs is based on the NFDH algorithm, and the running results of different algorithms are compared with them. Finally, a two-dimensional curve graph can be obtained with the width (height) factor as the horizontal axis and the algorithm / NFDH as the vertical axis.

Through the comparison of the two graphs, it can be seen that the influence of width factor on the algorithm results is much greater than that of height factor. More precisely, compared to the increase of height factor, the increase of width factor will make the result advantage of FFDH algorithm significant, from a ratio of about 0.97 to about 0.80, while the results of NFDH and sleator algorithm may be in the same changing trend. In addition, when we look at the result curve of the FFDH algorithm separately, although it only shows a stable upward and downward trend in the height factor result graph; However, in the width factor result graph, FFDH showed a significant decrease in the width factor from 0.10 to 0.40, indicating that the FFDH algorithm is more suitable for the presence of large width rectangles.

Next, let's take a look at the results of the sleator algorithm. Firstly, for the height factor, both the sleator algorithm and our improved sleator algorithm exhibit a stable fluctuation trend, and their overall performance has always been inferior to the NFDH algorithm; However, in the result graph of the width factor, we can see that the performance of the sleator algorithm has always been inferior to the NFDH algorithm. However, our appropriately improved sleator algorithm performs well when the width factor is not large, especially in the range of 0.40 to 0.60. Although the performance of the three algorithms (including the NFDH algorithm) tends to be consistent when the factor approaches 1.0, this also demonstrates the effectiveness of our improvement.

It may be that our method of cutting rectangles is too simple, resulting in low quality of the generated dataset. Therefore, even in the case of a large number of inputs, all three algorithms (one of which is an improvement) can run and achieve good results. However, this method of cutting large rectangles can use the optimal solution as a known condition, which is still helpful for our data generation and analysis.

Finally, let's focus on the test data from previous papers, of course, only a portion of which is listed above due to space limitations. Here, we use the dataset used by Professor Hopper in 2000. The process of generating the dataset is similar to that of cutting a large rectangle, but it determines the number of small rectangles that are ultimately cut from a large rectangle; And it is superior to our cutting method. It does not completely cut large rectangles, but rather appropriately discards some remaining area when the number of cuts is reached, which is more in line with the actual situation in engineering.

From the running results, it can be seen that although the algorithm we implemented is approximately large, even close to 2.7/3.0, in the vast majority of small data runs, the ratio of them to accurate results is often around 1.2 - 1.5.In such a performance, the test data of the paper dataset may not make a significant contribution to our exploration of influencing factors, and can only be used as an optimal cut large rectangular reference with the optimal solution.

**chapter 6 conclusion:**

After our testing and analysis, it can be seen that although FFDH pays the cost of runtime for the strip packing problem, it often performs better among the four algorithms we implement. Although the running time of the NFDH algorithm and the sleator algorithm remains similar, the performance of the sleator algorithm is often slightly better. However, with our appropriately improved sleator algorithm, while maintaining a certain level of optimal performance, for certain specific situations (such as large width rectangle generation factor in the range of 0.40 to 0.60 damage), better results can be obtained.

In summary, using the FFDH algorithm is often a better choice; But when considering both time and some special input sequences (such as square sequences, large width rectangular sequences, etc.), our improved sleator algorithm may achieve higher cost-effectiveness results.

**Reference:**

1. *Baker, Brenda S.; Coffman Jr., Edward G.; Rivest, Ronald L. (1980). "Orthogonal Packings in Two Dimensions". SIAM J. Comput.****9****(4): 846–855. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)" \o "Doi (identifier)):[10.1137/0209064](https://doi.org/10.1137/0209064).*
2. *Harren, Rolf; Jansen, Klaus; Prädel, Lars; van Stee, Rob (February 2014). "A (5/3 + epsilon)-approximation for strip packing". Computational Geometry. 47 (2): 248–267. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)" \o "Doi (identifier)):[10.1016/j.comgeo.2013.08.008](https://doi.org/10.1016/j.comgeo.2013.08.008).*
3. *Christofdes N, Whitlock C (1977) An algorithm for two-dimensional cutting problems. Oper Res 2530–44*
4. *Bengtsson B (1982) Packing rectangular pieces—a heuristic approach. Comput J 25:353–357*

1. From wikipedia https://en.wikipedia.org/wiki/Strip\_packing\_problem [↑](#footnote-ref-0)
2. Thanks to Professor Jan H van Vuuren for data integration, relevant data sources http://www.vuuren.co.za/main.php [↑](#footnote-ref-1)