Carpentry software designing and development with pane cutting optimization functionality

Bence Hodák
Obuda University
John von Neumann Faculty of Informatics
Budapest, Hungary
hodak.bence@stud.uni-obuda.hu

Elemér Balázs
Obuda University
John von Neumann Faculty of Informatics
Budapest, Hungary
balazs.elemer@uni-obuda.hu

2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI) | 979-8-3503-2110-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/SACI58269.2023.10158536

***Abstract*— In this paper, a carpenter software is designed and implemented that supports the daily management processes (customers, orders, inventory) and generates a pattern from the product descriptions based on the parameters specified. During the pattern generation process, a two-dimensional optimization is performed to generate a pattern that includes horizontal or vertical cuts, depending on the target machine, on which the position of the required parts is placed, generates the smallest material loss and uses leftover materials from the past. From the application, the pattern can be viewed and extracted online or in PDF format. This all is implemented in a layered web application with a modern approach.**

Keywords— carpentry software, 2D optimization, optimizer algorithms, cutting pattern, pane cutting.

I. Introduction

Nowadays, decision-support-systems and software are supporting everyday work and help to make production or product manufacturing as efficient as possible are almost without exception in modern business management. Some of these can be carried out by monitoring dynamic parameters, adaptively during production or with the help of artificial intelligence [1]. These systems usually perform optimization along certain parameters, allowing an organization to produce less waste or save time and thus be more successful.

To understand the topic, it is important to look around the subject of pattern making. You need to look at the most common methods. You need to highlight the parameters of the products you want to process and the cutting solution you want to use. It is necessary to mention these because some of them show the grain of the wood, so that the orientation of the part in the pattern is not always the same. Such materials include laminated chipboard, veneered panels and plywood.

Most of the panel cutting machines work with either a saw blade or a band saw, which in practice means that when designing the pattern, you may consider that the starting of the sawing must be somewhere on the edge of the material, so you need an entry point(s) where you can start cutting. There will also need to be an exit point(s), as you cannot turn around or go back through the cut, as you may damage the material. Unlike with CNC and other machines for cutting, it is necessary to plan exactly where the cuts will go. The best possible layout, on a cutting machine, may not be cut as it is on the drawing. Today, horizontal and vertical cutting machines are the most used in the industry.

The horizontal saw machine is an improved version of the traditional saw. The saw blade or band is in a fixed position, and a mechanism is used to move the material. It has the advantage of being more flexible, more suitable for custom cutting patterns. The vertical panel slicer differs in that the workpiece is in a fixed position and cutting unit is moved. Less flexible, it is best suited for cutting rectangular parts.

It is essential to produce a pattern from which the parts can be cut by a saw blade or a band saw, not just a CNC machine, because of arrangement, and to take into consideration the grain of the material, the width of the saw blade. The cutting pattern generated by the software requires a drawing of the optimal layout of the parts with their corresponding dimensions and, for identification purposes, the name of the part. Information from [2].

II. Related work

In this section, some of the similar software on the market will be presented and the algorithms that solve the optimization problems mentioned previously will be explained.

A. Similar solutions

There are several types of sheets cutting software on the market with different features, explained in the following chapter.

"**BIPPO** is a furniture panel cutting optimization program that allows fast, error-free, mixed-fiber optimization and graphical, to-scale cutting plan visualization." [3]

According to the description on **PaneCutter**'s [4] official site, it uses a recursive algorithm to find the optimal cutting plan. In doing so, it strives to minimize waste. The output is the first optimal solution and then a choice of other good solutions.

"The **AMORF** software generates optimal cutting plans quickly and automatically, with negligible cutting loss and no accumulation of scraps." [5] Parameters that can be entered: direction of grain (longitudinal and transversal), difference between pattern and finished size, set of leftovers (optional), set of full panels (in case of shortage of material, fictitious set).

Optimik 4 [6] is, according to its website, a modern multifunctional sheet cutting application with the possibility to create projects, manage inventory and finances, among other things.

Using the application, you can create projects (furniture) and then add parts, set prices, and set up materials. You can set several parameters for the patterns (for example, cutting method) [7].

You can find the comparison of similar software in *Table I*.

Table I
Comparison of similar software

	BIPPO	PaneCutter	AMORF	Optimik 4
Input	Manually, from Excel file	Manually	Manually	Manually, From database
Output	On screen, In printable format	On screen, In printable format	On screen, In printable format, Can be sent to machine	On screen, In printable format
Cutting pattern	Keeping the orientations	Keeping the orientations With multiple options, Without label	Keeping the orientations of the parts, With multiple options, With labels	Keeping the orientations of the parts, With multiple options, With labels
Functions	Cutting pattern generating	Cutting pattern generating	Cutting pattern generating, Inventory management	Cutting pattern generating, Inventory and finances management

B. Algorithms

In the following chapter, the different solutions and the optimization problem are discussed.

This problem is referred to in the corresponding publications as the two-dimensional cutting stock or two-dimensional bin packing problem, which is not only a problem in board cutting, but also in glass, steel cutting and container packing. The goal is to cut as many different sized rectangles as possible, from larger sized rectangles (raw material boards), using as little as possible.

Given a finite number n_E of rectangles, $E = \{1, \dots, n_E\}$, each element has a height $h_i \in \mathbb{N}^+$, a width $w_i \in \mathbb{N}^+$and how many pieces of it we need $d_i \in \mathbb{N}^+$. The raw material has height $H \in \mathbb{N}^+$, and width $W \in \mathbb{N}^+$. The elements, in this case, are considered to be rotatable in 90°, which means that we assign a rotated version to each rectangle. The aim is to find the optimal cutting pattern P with no overlapping rectangles, the minimum number of boards used, and the loss is minimal.

Approaches to this problem are offered by two- and three-phase Guillotine cuts, the Corner Placement Algorithm, the Pane Division Method, and Heuristic Genetic Algorithm-based solutions.

1) 2-phase guillotine cut

In the case of the 2-phase guillotine cut [8], the solution consists of two phases, a horizontal and a vertical slice. First, we reduce the problem from two-dimensional to one-dimensional. This is done by dividing the board into strips along one of the sides, so that the other elements to be placed fit on those strips. The parts are then placed on these slices as a one-dimensional cutting stock problem. This solution works fine, but in cases where there are no elements with the same side length, a lot of waste is left over (Fig. 1). The solution to this problem is the 3-phase Guillotine cut.

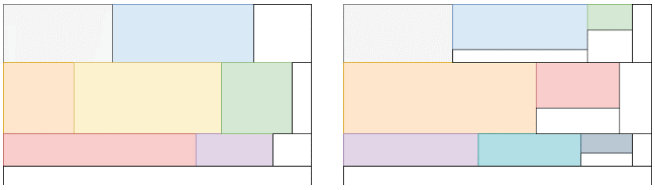


Figure 1 - Results of the 2-phase guillotine cut with optimal part sizes (on the left) and with not optimal part sizes (on the right)

2) 3-phase-guillotine cut

Using the 3-phase guillotine cut [9], the cut consists of 3 phases, the first and the third horizontal, the second vertical. The cutting pattern is determined from a cutting tree. The top left corner of the raw material board is (0; 0) and the bottom right corner is (H; W). On each used board, the height of the leftover strip from the (0; 0) coordinate is $\rho_j\ j = 1, \dots, n$ if there is no piece of waste, $\rho_j = 0$, c(P) is the largest piece of waste, determined by equation (1).

$$c(P) = \min\left(n - \frac{max_{j=1,\dots,n}\rho_j}{H}\right) \qquad (1)$$

The algorithm generates the cutting tree as follows. We create s_1, this is the board we start from, the root of the tree. Then we cut horizontally, these will be the s_{1n}boards. Then we cut vertically and then we cut horizontally again. These will be the s_{1nm} and s_{1nmk} boards (Fig. 2). From this we get the optimal placement.

3) Corner Placement Algorithm

Corner Placement Algorithm [10] provides an alternative strategy, that might be to sort the parts in order of some kind of fitness. Then, selecting the best item from this, place it in one corner of the material board. A corner is defined as the place where the component touches either two sides of the board, or one side of the board and one side of another component, or one side of two parts. Since our board is empty at startup, there are 4 free corners of a rectangle, so we have four possibilities, however, if we can rotate the components but not the base, we have eight. Since we don't know which solution will be optimal, we will continue with all the possibilities. After choosing the next best element, we also need to find a corner for it. However, in this case, there are already 5 locations that can be considered as corners, which, due to the rotatability of the component, makes 10. A rule is needed to narrow down the possibilities.

For example, select the positions with the smallest distance to another part. We will create a tree with all the possible solutions until we have used up all our elements. At the end, we compare the solutions and select the ones with the highest fitness value (Fig. 3).

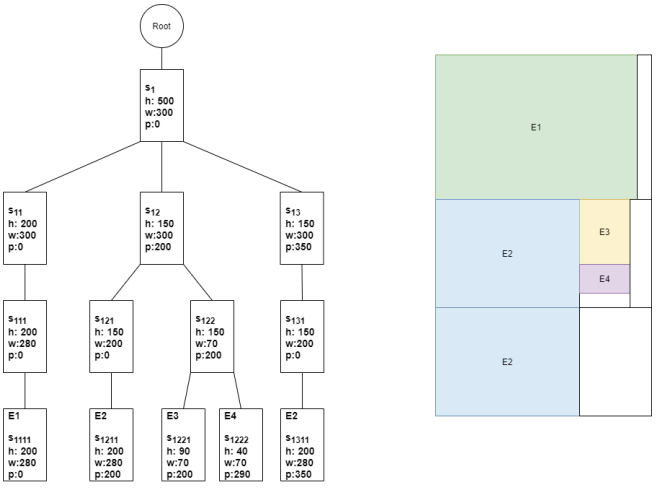


Figure 2 - The cutting tree (on the left) and the result (on the right) of the 3-phase-guillotine cut

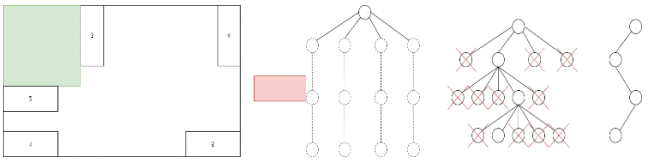


Figure 3 - Corner Placement Algorithm's part position options (on the left) with the tree (in the middle), and how we can extract the solution from it (on the right)

4) Pane Division Method

The method below is a recursive algorithm based on our own idea and similar in functionality to the solutions already discussed. The parts are arranged in descending order by width. The widest part is placed in the upper left corner, then extending its right side downwards, two free areas are formed, one below the placed part and the other adjacent to it. Then the first phase begins: we start placing the parts under the first element, always using the space underneath it, and then extending the right side of the next parts, until we run out of space. If we run out, we lay down the next element on the plane next to the last element we placed. In the case that you can fit a next component below then, just as you did for the first phase, you start the next phase and place the components underneath until you have space. When you are out of space, you look at the part next to the last placed part and start the next phase. If we cannot place an element there, we look at the second to last element placed and start the next cycle with that. We step back and start the next recursion in a back track way until we run out of the items to place (Fig. 4(a)).

5) Heuristic Genetic Algorithms

Bottom-Left Algorithm [11] means to start from the top right to find the ideal position and keep moving towards the bottom left corner to find the most left and bottom position for the part using Genetic Algorithm. We can see that this will not give an ideal layout under all circumstances, as it could create large areas of empty fields, which will not be filled due to the way the algorithm is designed (Fig. 4(b)). Since it is a genetic algorithm, it is important to look at the runtime. If we take N to be the total number of elements placed, then the runtime is $O(N^2)$.

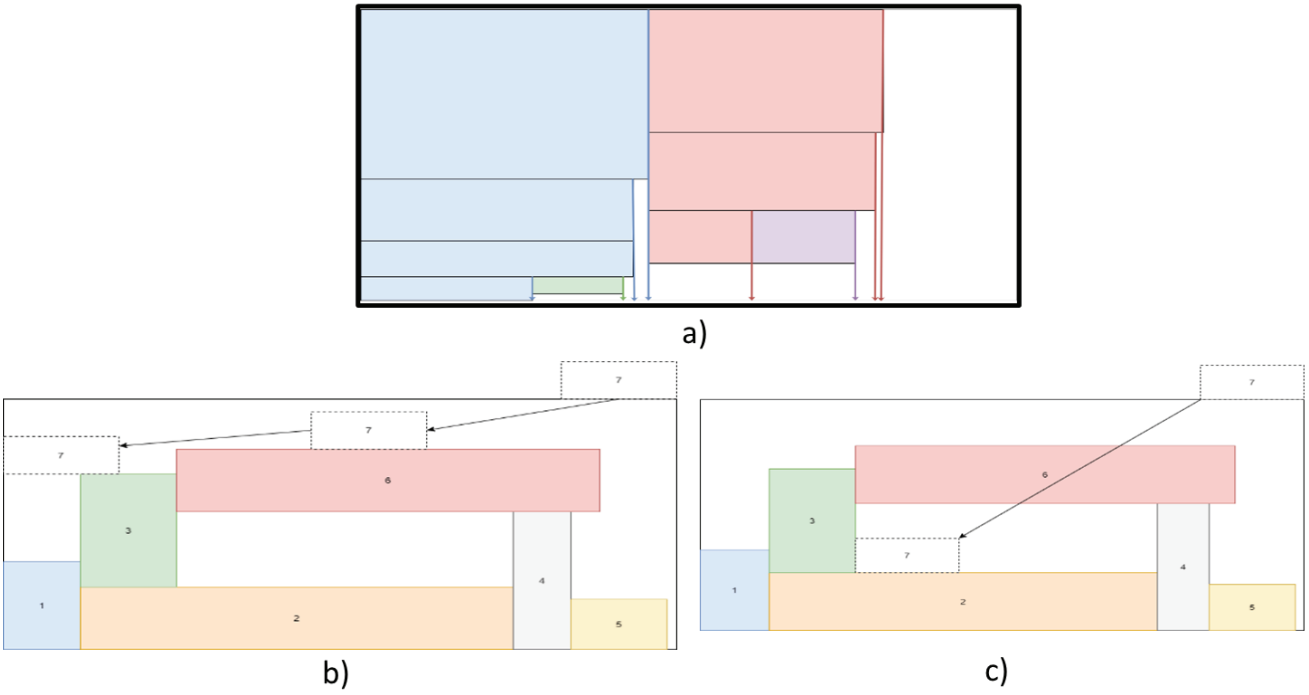
The solution for the space improper fill is the **Bottom-Left-Fill** [11] algorithm, an improved version of this method. The improvement means that we try to fill the existing gaps, using the most left and bottom down approach (Fig. 4(c)).

C. Summary of the related work

In the section of the related work, a number of similar programs and methods for solving the problem have been examined. The conclusion of our research is that what the industry needs is a program to unify the useful functions of these programs. It is necessary to manage inaccuracies caused by the saw, the rotatability of the elements, to save the leftover pieces of board after cuts for reuse, in order to minimize unnecessary waste. It is necessary to manage inaccuracies caused by the saw, the rotatability of the elements, to save the residual pieces of board after cuts for reuse, in order to minimize unnecessary residual training. Looking at the pattern, it would be important to label the elements on it, as well as to indicate their dimensions. An important constraint for the pattern is that it should be possible to export it in printable form. Looking at the optimization methods, the overall impression is that a Guillotine cut like algorithm would be the most useful, since this method always has a through cut, crucial because you cannot go back and turn with the cutting machine. The goal is to have a simple interface application with only well-defined and relevant functions, providing a fast and optimal solution to the problem.

III. Methodology

The Plane Division method will be used to implement the optimization, which will be compared with the two-phase Guillotine cut algorithm at Chapter IV. The Pane Division algorithm's pseudo code could be seen in *Algorithm 1*.



4. Figure - a) The flow of the Pane Division Method, blue is the first phase, green is the second, red is the third and purple is the fourth; b) The flow of the Bottom-Left Algorithm; c) The flow of Bottom-Left-Fill Algorithm

Algorithm 1 Pane Division

Input: *as reference* board– **T**, *as reference* parts– **T list**, *as reference* optimized_parts – **T list**, *as reference waste*_pieces – **T list**, board_count - **number**

```
1:  function PaneDivision(board, parts,
    optimized_parts, waste_pieces, board_count)
2:     loop i from 0 to parts.lenght
3:        actual_part ← parts[i]
4:        if ¬(actual_part.X = 0 ∧ actual_part.Y = 0 ∧
          actual_part.Width = 0 ∧ actual_part.Height =
          0) ∧ actual_part.Width < board.Width ∧
          actual_part.Height < board.Height then
5:           parts \ actual_part
6:           actual_part.X ← board.X
7:           actual_part.Y ← board.Y
8:           actual_part.Board_count ← board_count
9:           horizontal_board ← T (X: board.X +
             actual_part.width, Y: board.Y, Width:
             board.Width - actual_part.Width, Height:
             board.Height)
10:          vertical_board ← T (X: board.X, Y: board.Y
             + actual_part.Height, Width:
             actual_part.Width, Height: board.Height -
             actul_part.Height
11:          optimized_parts ∪ actual_part
12:          PaneDivision(vertical_board, parts,
             optimized_parts, waste_pieces,
             board_count)
13:          PaneDivision(horizontal_board, parts,
             optimized_parts, waste_pieces,
             board_count)
14:       end if
15:    end loop
16:    if board ∉ waste_pieces then
17:       waste_pieces ∪ board
18:    end if
19: end function
```

T: An object that has x, y coordinate, height, width and table_number properties.

The optimization success rate is determined in the following way. The total residual area is divided by the area of the rectangular raw materials used. Then, the area of each residual part is divided by the area of the largest part, added together and averaged. Finally, this average is multiplied by the first result (eq. (2)).

$$result = \frac{A_{waste}}{A_{used}} \times \frac{\sum_{i=0}^{n} \frac{A_{waste_i}}{A_{part_{max}}}}{n} \qquad (2)$$

If all the raw materials are used, the result will be 0, the more unused areas we have, and the more raw materials we have on the board, the higher this value will be.

IV. Evaluation

Analyzing the finished software, it can be concluded that we have obtained a software with a modern layered architecture and state-of-the-art technology, with functions that meet the objectives. But we know, most of these functionalities can be separated into independent microservice to achieve more flexibility and maintainability in the software, which is relevant even the enterprise is growing huge [12]. In terms of functionality, it fulfils the requirements and generates, as expected, patterns from the input data as you can see on Fig. 5, 6, 7 and 8, showing the information previously formulated. To evaluate the results of the pattern generation, our own algorithm has been compared with the two-phase guillotine cut using the equation (2). The results of the comparison are shown in Table II. and III. through the different test cases.

Table II
Test cases for Pane Division and 2-phase guillotine cut methods (Without using leftover materials (board size: 2600x2200))

	Test case	Pane Division	2-phase guillotine
1.	2600x2200	0	0
2.	2pcs 2600x2200	0	0
3.	2600x2200, 2200x2600	0	0
4.	2000x2000	0.30745	0.30745
5.	3000x3000	no pattern	no pattern
6.	2pcs 3000x3000	no pattern	no pattern
7.	3000x4000, 3000x4000	no pattern	no pattern
8.	4pcs 500x500, 8pcs 500x1000	0.02703	0.02703
9.	8pcs 500x500, 16pcs 500x1000	0.02149	0.02149
10.	8pcs 500x500, 16pcs 500x1000, 2600x200	0.00897	0.31039
11.	8pcs 500x500, 16db 500x1000, 2pcs 2600x200	0.24409	0.24409
12.	52pcs 500x500	0.10263	0.10263
13.	13pcs 500x1000, 7pcs 500x500	0.12971	0.12971
14.	4pcs 500x500, 8pcs 500x1000, 2600x200	0.15386	0.96233

Table III
Test cases for Pane Division and 2-phase guillotine cut methods (With using leftover material)

	Test case	Pane Division	2-phase guillotine
12.b	52pcs 500x500, using its waste	0.35853	0.34857
12.c	52pcs 500x500 using its waste from 12.c	0.21510	0.21259
13.b	13pcs 500x1000, 7pcs 500x500, using its waste	0.32246	0.60320
13.c	13pcs 500x1000, 7pcs 500x500 using its waste from 13.b	0.09667	0.09022

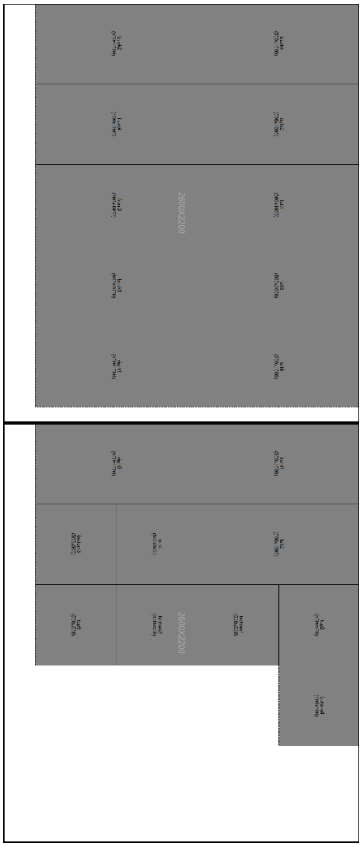


Figure 5 - Pane Division pattern from test case 13

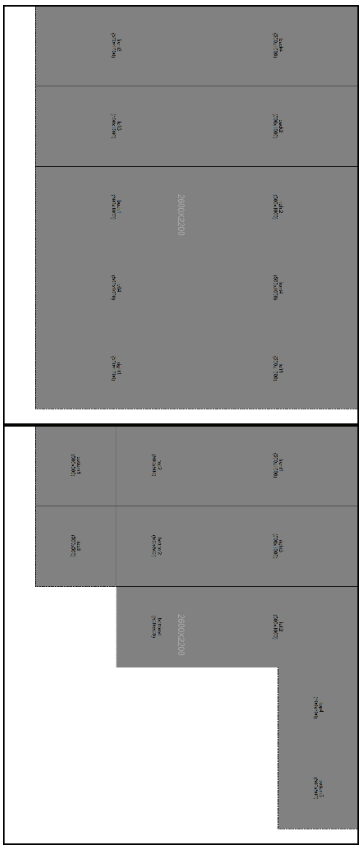


Figure 6 - 2-phase guillotine from test case 13

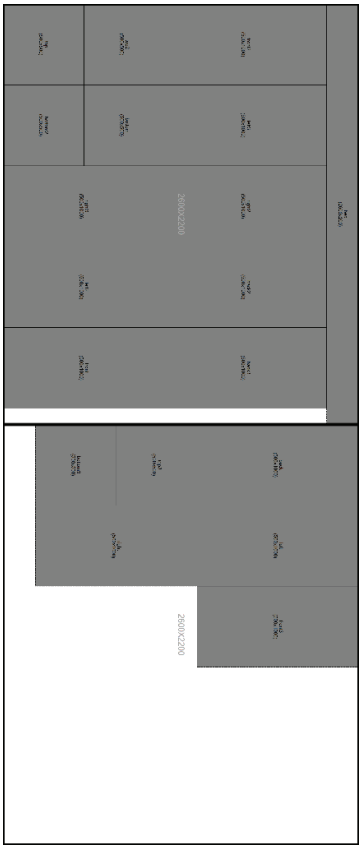


Figure 7 - Pane Division pattern from test case 14

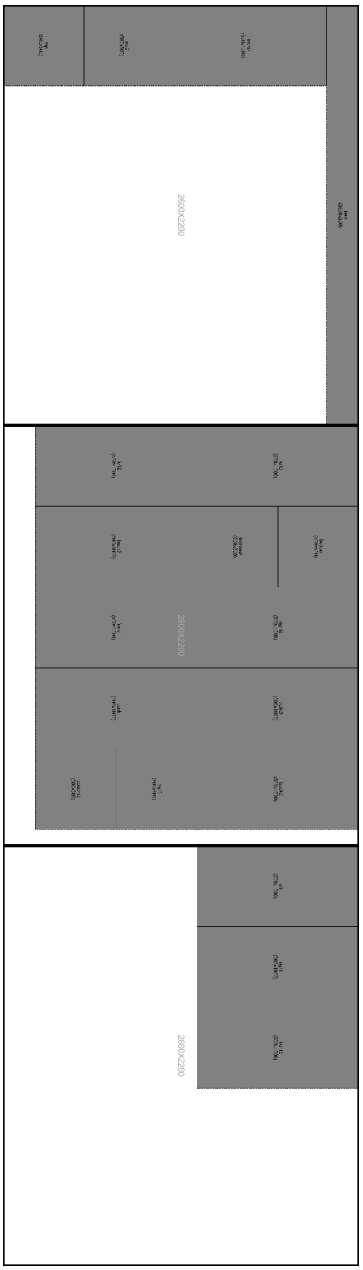


Figure 8 - 2-phase guillotine pattern from test case 14

By looking at the pattern and viewing the goodness of fit values, we can see that the two methods give similar results in most cases, the amount of material used is the same, and the difference in the layout is the position of the last few elements. However, there are cases 11, 12, 14 where the layout of the pattern is much more efficient using our method. It can be concluded that both methods provide an efficient solution in cases where several of the part types are identical, however, the Guillotine cut runs into a problem when a part with widely different parameters is added to the parts to be optimized. Our own algorithm handles this case better, since the algorithm of the other solution will only be effective if the parts in the second phase are of similar size to the initial part deposited in the first phase. In the cases using the remainder, we also get similar results. Overall, our algorithm is more effective without using residue, but no such conclusion can be drawn when using residue. Since we are using the residuals from the previous optimization, the plane splitting method uses less raw material in some cases, there will be less residual, so this may result in using the residual, more new raw material is needed,

which results in more residual. However, it should be noted that our own algorithm has a better layout, which can process the residual raw material tables with greater efficiency.

By examining the pattern, you can see that it meets the requirements and download it in PDF format, ready to print. All the boards used can be printed on a single A4 sheet without resizing, and there is no problem in using multiple boards because they are numbered in sequence. And by adjusting the thickness of the saw blade, the distance between the parts is exactly the same as the loss caused by the cut.

Compared to the software reviewed in the chapter on related work, our solution has more features than most of the software reviewed, but none of them are as deep or sophisticated. The pattern generating component is on a par with the rest of the software. Our software is a transition in terms of features and functionality between PaneCutter and Optimik4, as the former's pattern generation was the most rudimentary and the latter has more features.

As for further improvements and limitations, the missing reading from .CSV or other input source files should be mentioned.

V. CONCLUSIONS

In this paper, a carpentry software was designed and implemented, in the scope of which We examined the furniture cutting industry, the materials and technologies used in it, and compared existing applications on the market. On the basis of these results, a set of requirements for the software to be designed and implemented was described. Various heuristic and deterministic optimization algorithms were discussed, one of which, as well as an algorithm of our own invention, was implemented in the software. A software has been implemented based on the above-mentioned research. The completed software was then tested. By means of several test cases, the goodness of the pattern generation function using leftover raw materials was investigated. Finally, the capabilities and limitations of the finished software were compared with the tested software.

ACKNOWLEDGEMENTS

The authors would like to thank the Hungarian National Talent Program (NTP-HHTDK-22) for its valuable support.

REFERENCES

[1] P. Bris, J. Hyza, M. Sedlacek, E. Kramna: Use of Quality Management to Optimize Foundry Industry Processes, Acta Polytechnica Hungarica, 2021, pp. 213-232

[2] M. Schlosser, “Carpentry industry and saw machine information,” October, 2019. [Online]. Available: https://faipar.hu/hirek/gep-es-szerszam/9203/fueggoleges-vagy-vizszintes [Accessed April 23, 2023].

[3] BerényiSoft Kft., “BIPPO software website,” July 15, 2022 [Online], Available: https://berenyisoft.com/tag/butorlap-szabaszati-optimalizalo-program/ [Accessed April 23, 2023].

[4] Bendeguz LTD, “PaneCutter software website,” [Online] Available: http://www.panecutter.eu/index.htm [Accessed April 23, 2023].

[5] Amorf-szoft Bt., “AMORF software website,” 2013, [Online] Available: http://amorfszoft.hu/amorf_b.html [Accessed April 23, 2023].

[6] R. Korytár, “Optimik 4 software website,” [Online] Available: https://www.optimik.com/description.html [Accessed April 23, 2023].

[7] R. Korytár, “Optimik 4 cutting patterns,” [Online] Available: https://www.optimik.com/plan.html [Accessed April 23, 2023].

[8] P. C. Gilmore, R. E. Gomory (1965). Multi-Stage Cutting Stock Problems of Two or More Dimensions. Operations Research. 13. 10.1287/opre.13.1.94.

[9] F. Dusberger, G. R. Raidl: Solving the 3-Staged 2-Dimensional Cutting Stock Problem by Dynamic Programming and Variable Neighborhood Search, Electronic Notes in Discrete Mathematics, Volume 47, 2015, pp. 133-140, ISSN 1571-0653,

[10] M. Chen (2008). A Greedy Algorithm with Forward-Looking Strategy, Greedy Algorithms, Witold Bednorz (Ed.), ISBN: 978-953-7619-27-5,

[11] E. Hopper, B.C.H Turton: An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research, Volume 128, Issue 1, 2001, pp. 34-57, ISSN 0377-2217,

[12] I. Poloskei: Enterprise-Level Migration to Micro Frontends in a Multi-Vendor Environment, Acta Polytechnica Hungarica, 2021, pp. 7-25