MGPM Methods for Rectangular Single-Size Printed Circuit Board Orthogonal Packing Problems

Pengfei Pan, Yi Han, Hexin Lv, Guoyong Dai, and Zhiyong Jin

School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou, China
Zhejiang Shuren University, Hangzhou, China

Correspondence should be addressed to Yi Han; 187267654@qq.com

Received 23 February 2022; Revised 12 July 2022; Accepted 19 September 2022; Published 12 October 2022

Abstract

Printed circuit board (PCB) orthogonal packing problems derive from the massive circuit board production process in the electronic industry. In the early stage, it mainly relied on human experiences to make decisions on how to make the most of each sheet while meeting customers’ orders. Up till now, researchers in enterprises and academic circles are still trying to explore effective mathematical models and feasible optimization methods. In most cases, what PCB companies generally face is the orthogonal layout decision, which considers positioning each rectangle PCB piece (PCB-P) on a rectangle sheet board (panel) in an ideal way so that each panel has the minimal remains with the most PCB-Ps. In this paper, multi-round gradual packing methods (MGPMs) are proposed based on the idea of combinatorial optimization and dynamic programming. MGPMs include the depth priority-based method (MGPM-DP) and breadth priority-based branch-and-prune method (MGPM-BC). The former has the advantage of finding better solutions, while the latter has the advantage of consuming short computational time. Through extensive computational tests on real data from a PCB production enterprise in China, both MGPM-DP and MGPM-BC have achieved noticeable and satisfactory results compared with the simplex method, dynamic programming method, and a widely used industrial software (Yuanbo) in China.

1. Introduction

PCB is the core component of various types of electronic products and electrical equipment. From mobile phones to large aerospace equipment, almost all electronic products contain PCB. With the arrival of an informational society and the fast development of high-techs such as Industry 4.0, big data, cloud computing, and the Internet of things, PCB plays an even more crucial role as a solid foundation in the whole industrial chain. Since the industrial revolution, manufacturing enterprises have always been valuing the utilization rate and cost saving of raw materials.

In the PCB production industry, rectangular PCB piece (having a length (L or l) and a width (W or w)) packing problem (RP-PPP) has always been a tricky thing bothering enterprises. RP-PPP considers how to arrange a number of PCB-Ps on a whole sheet board (SB) in an ideal way to meet the production plans while improving the utilization rate of the SB, reducing production costs, and boosting PCB production efficiency [1]. Sometimes, a PCB-P is not rectangular. Enterprises normally capsule an irregular piece into a rectangle if this situation happens.

There are two types of rectangular PCB packing decisions here. One is single-size RP-PPP, and the other is multi-size RP-PPP. Single-size RP-PPP revolves around positioning PCB-Ps of the same size on a sheet board. Multisize RP-PPP can be transferred into multiple single-size RP-PPPs. Therefore, researches on models and algorithms of single-size RP-PPP are fundamental and especially essential [2].

PCB packing problem is highly similar to the 2D graphics layout (cutting-stock) problem. Therefore, the models and algorithms for 2D graphics layout problems could apply as references to a great extent. As is known to all, cutting or packing problems are NP-complete ones. Therefore, it is impossible to depict each detailed constraint
exactly. Normally approximate models are employed for this kind of problem for constraints such as a PCB piece should not overlap with others. Exact models are suitable for problems with explicit objectives and specific constraints. And both exact methods and approximate ones can be applied to solve problems with exact or approximate models. Only that, when an exact method is used for an approximate model, the solutions may not be optimal. Currently, exact methods such as linear programming method (simplex method), dynamic programming method, and branch-and-bound method are mainly and extensively adopted means for solving single-size cutting-stock problems (SCPs) [3].

The nesting idea runs through the cutting problem of one-dimensional tubular objects, the cutting problem of two-dimensional graphics, the packing problem of three-dimensional space, and even the packing problem under four-dimensional time constraints [4]. 2D cutting-stock problem is a special case of a 2D bin packing problem.

Agrawal [5] proposed a branch-and-bound algorithm (BBA) for single-size RP-PPP. The algorithm first constituted 2 strips based on the guillotine cut idea. Then 2-section, 3-section, 4-section, , , and \( n \)-section schemes were constructed, with each scheme being composed of 2 plans. 3-section schemes were derived from 2 2-section plans, and 4-section schemes came out from 2 3-section plans. And in a similar fashion, \( n \)-section schemes were produced by gradually splitting the bottom-right rectangular remains on the sheet board. This algorithm reduced search space and computing time to some extent. However, the optimality was not completely assured. Arslanov [6] proposed a continued fractions method, which could reduce the complexity of obtaining the total number of PCB pieces on the sheet board. This algorithm, which was based on mathematical theorems and lemmas, could greatly simplify the whole computational process. However, this algorithm could only explicitly point out the layout of PCB pieces on the bottom-left and up-right corners of the sheet board. And it did not guarantee optimality as well. Cui et al. [7] proposed a continuous fractions-based BBA. This algorithm adopted the continuous fractions method to obtain the upper bound. Then a branch and bound procedure were performed to produce the layout scheme gradually. As we have mentioned before, the continuous fractions method was based on the guillotine cut idea. Thus, the upper bound may not be an ideal one. He and Cui [8] designed a dynamic programming algorithm, which was based on two types of strips, as shown in Figures 1(b), and 1(d). Yang and Cui [9] performed research, which is based on literature [8], and considered cutting numbers for each scheme. However, the effect of cutting numbers was set too small to be a leading factor.

Li [10] developed a two-stage method based on the dynamic programming algorithm and a -D knapsack algorithm. Firstly, several workplates were constructed. Then, a dynamic programming algorithm was applied to determine the layouts of PCB pieces on each workplate. Afterwards, several segments, which were stacks composed of workplates with the same width as shown in Figure 1(d), were obtained by solving a 1D knapsack problem. Finally, segments were selected by solving another 1D knapsack problem and packed horizontally on the sheet board. Wang et al. [11] and Cui et al. [12, 13] solved multi-workplate and single-workplate PCB packing problems with dynamic programming algorithms and segment-based methods. References [7–13] are all features of the dynamic programming algorithm as described in the literature [9]. The possible common crises are when only two strips (actually, there are four types of strips, as shown in Figure 1) are considered and when the remaining sheet is satisfying \( w \leq L \leq l \) or \( w \leq W \leq l \). For one thing, their algorithms may lead to unwanted waste. For the other, errors will occur if no further control measures are performed. References [14–16] are strip and stage-based methods. First, in stages, segments should be decided. Then, in each segment, homogenous strips with the same orientation will be generated to fill the whole segment. In these methods, efforts should be cast on how many segments there will be. And strip-based ideas may lead to more wasted materials.

Beasley [17] proposed a tree-based exact algorithm for a single-size rectangle packing problem in 1985. This algorithm adopted a relaxation technique that captures satisfactory approximate optimal solutions. Litvinchev et al. [18] designed an approximate integer programming algorithm that could guarantee feasible satisfactory solutions as well. Reference [19] introduced strip-based algorithms or the simplex method for 2-dimensional packing problems. Those methods are mostly feasible solutions generating tools too. Only in the pallet packing section the book presented 3-, 5-, 7-, and 9-block methods for nesting packing problems, and the 9-block method exhibited some optimal solutions with diagonal patterns for several cases. However, for one thing, if the scale of a problem is large, the 9-block method would not always be the best one (11-, 13-, 15- or 17-block methods may be needed). For the other, when and how a diagonal pattern stands is still up in the air. In addition, there are many possible block types in each block-based method. Therefore, making sure which method and block types should be employed and what the placement sequence of all the blocks will be really time-consuming. For a manufacturing factory, satisfactory solutions with simpler packing patterns and acceptable time would be more plausible than optimal solutions with intricate separation lines and unbearable computational time.

In this paper, a step-by-step gradual nesting method based on the dynamic programming idea and a branch-and-prune strategy-based method is proposed from the perspective of making full use of the length and width of the sheet board. According to the experimental results, satisfactory and feasible results are obtained within acceptable computational time.

2. Mathematical Models for Constructing Feasible Solutions

Currently, there are not many mathematical models for RP-PPP since RP-PPP itself is difficult to depict explicitly. Normally, two models appear in most references. One is the
simplex method-based model, and the other is the dynamic programming algorithm-based model. Both algorithms are based on strips [9].

2.1. Simplex Method Based Model. Suppose the rectangle sheet board is in a coordinate system. The coordinate of the bottom-left corner point is (0, 0). The X-axis overlaps with the length side of the sheet board and the Y-axis overlaps with the width side of the sheet board. The small PCB piece is has a length \( l \) and a width \( w \) as well.

\[
\begin{align*}
X\text{-axis:} \\
\text{Max } & C_{\text{XX}} = \left\lfloor \frac{W}{w} \right\rfloor x + \left\lfloor \frac{W}{l} \right\rfloor y \\
& l \cdot x + w \cdot y \leq L \\
& 0 \leq x \leq \left\lfloor \frac{L}{l} \right\rfloor \\
& 0 \leq y \leq \left\lfloor \frac{L}{w} \right\rfloor.
\end{align*}
\]

(1)

X-axis algorithm procedure:
Step 1: Establish the objective function \( C_{\text{X}} \) maximizing the number of PCB-Ps
Step 2: Set constraints
Step 3: Call the Linear Program Optimizer in Matlab2016
Step 4: According to \( x \) and \( y \), call the rectangle() function to draw the layout of strips on sheet board

Y-axis:
\[
\begin{align*}
\text{Max } & C_{\text{YY}} = \frac{L}{l} x_1 + \frac{L}{w} y_1 \\
w \cdot x_1 + l \cdot y_1 \leq W \\
0 \leq x_1 \leq \left\lfloor \frac{W}{w} \right\rfloor \\
0 \leq y_1 \leq \left\lfloor \frac{W}{l} \right\rfloor.
\end{align*}
\]

(2)

Y-axis algorithm procedure:
Step 1: Establish the objective function \( C_{\text{Y}} \) maximizing the number of PCB-Ps
Step 2: Set constraints

Step 3: Call the Linear Program Optimizer in Matlab2016
Step 4: According to \( x \) and \( y \), call the rectangle() function to draw the layout of strips on sheet board

At present, most commercial PCB packing software has embedded a simplex method. Since the simplex method only considers the best combination of strips packing horizontally or vertically, some potential optimal layouts may be missed out [8].

2.2. Dynamic Programming Method Based Model. The dynamic programming model, which is based on two types of strips, is listed below:

\[
F(L, W) = \max \left\{ F(L, W - l) + \left\lfloor \frac{L}{w} \right\rfloor, F(L - l, W) + \left\lfloor \frac{W}{w} \right\rfloor \right\}.
\]

(3)

\[
F(L, W) = 0.
\]

(4)

Dynamic programming algorithm performs in a divide-and-conquer way. First, suppose the sheet board is divided into two layouts, with each one having two parts. The first part is a strip, and the other part is the remaining board (remains). As long as the remains are not satisfying formula (4), the remains of each layout will be further divided into two layouts, with each one having two parts. As the dividing process goes on, the layout acquiring procedure extends like an upside-down tree. Through comparing each tree leave, the layout with the most PCB-Ps emerges as the best scheme.

However, it is taken as the terminating condition the dynamic programming algorithm will go wrong when \( w \leq L \leq l \) or \( w \leq W \leq l \). Therefore, we rectify formula (4) as formula (5):

\[
F(L, W) = 0, \min (L, W) < \min (l, w).
\]

(5)

The advantage of the dynamic recursive procedure is its simple dividing steps. But when the scale of problems increases, the amount of computation, memory usage, and time costs rise sharply as well.

3. Multi-Round Gradual Packing Methods

The strip-base simplex method and the recursive-based dynamic programming algorithm are 2 commonly applied heuristic algorithms for single-size PCB packing decisions. While in some cases, they are not guaranteeing optimal results. Assuming the sheet board is 3000 (mm) in length...
and 1500 (mm) in width and the piece board is 406 (mm) in length and 229 (mm) in width, the best packing result is shown in Figure 2. While the best scheme given by using commercial software (Yuanbo), which heavily relies on the strip-based simplex method or the dynamic programming algorithm, is shown in Figure 3. In Figure 2, the nesting packing layout contains 48 pieces, which is better than the 47-piece scheme in Figure 3.

3.1. Prerequisite Conditions. Before the introduction of our proposed methods, there are some variables need to be listed as follows:

\[ L: \text{Length side of the sheet board} \]
\[ W: \text{Width side of the sheet board} \]
\[ l: \text{Length side of the small piece board} \]
\[ w: \text{Width side of the small piece board} \]
\[ a: \text{Number of type 4 strips (Figure 1(d)) on L;} \]
\[ b: \text{Number of type 3 strips (Figure 1(c)) on L;} \]
\[ c: \text{Number of type 1 strips (Figure 1(a)) on W;} \]
\[ d: \text{Number of type 2 strips (Figure 1(b)) on W;} \]
\[ CL: \text{Matrix composed of combinations on L satisfying certain constraints } \{a_1,b_1; a_2,b_2; a_3,b_3;\ldots\}; \]
\[ CW: \text{Matrix composed of combinations on W satisfying certain constraints } \{c_1,d_1; c_2,d_2;c_3,d_3;\ldots\}; \]
\[ clcw: \text{Row vector generated from a Cartesian product of } CL \times CW \{a,b,c,d\}; \]
\[ Msiz: \text{Matrix composed of clcw s the small piece board into another}\]
\[ RECT: \text{Matrix composed of rectangle pairs from } Msiz \}
\[ [a_1 \times l_1, c_1 \times w],[b_1 \times w_1,d_1 \times l_1];[a_2 \times l_2, c_2 \times w],[b_2 \times w_2,d_2 \times l_2];\ldots;\]

Inspired by the simplex method, a two-dimensional packing problem is reduced to two one-dimensional packing problems based on L and W, respectively. All enumerated constraint-satisfying combinations \((a,b)\) on L are denoted by \(CL\). And all enumerated combinations \((c,d)\) satisfying the constraints on W are denoted by \(CW\). And \(clcw=(a,b,c,d)\) is gained by doing the Cartesian product of \(CL\) and \(CW\). Then \(Msiz\) and \(RECT\) are obtained afterwards.

3.2. Nesting Packing Layout. In the nesting scheme shown in Figure 2, if the piece boards 1–12 and 13–24 are regarded as two new rectangular boards A and B, respectively, the nesting scheme will be of 4 rectangular, among which the lower-left corner and the upper-right corner are two identical rectangles, and the upper-left corner and the lower-right corner are two identical rectangles.

By observing Figure 2, it can be found that the length \(L\) and width \(W\) of the sheet board in the nesting scheme is almost fully utilized. And the best combinations of Strips on L and W are captured by an enumeration way. The best combination option on length \(L\) is \(4 \times l + 6 \times w\), and the best utilization mode on width \(W\) is \(3 \times w + 2 \times l\). Therefore, it can be considered to combine \(4 \times 3 \times l \times w\) piece boards into one large rectangle and \(6 \times 2 \times w \times l\) piece boards into another large rectangle. When a nesting layout comes into being, there is no room for a single-piece board on the remains in the middle.

The decisive factors of nesting packing layout are as follows:

1. Number constraint: only 2 different large rectangles are concerned
2. Length and Width constraints: \(L-w<(a \times l + b \times w) \leq L\) and \(W-w<(c \times w + d \times l) \leq W\) and \((a \times l \geq b \times w\) and \(c \times w \leq d \times l\) or \((a \times l \leq b \times w\) and \(c \times w \geq d \times l\)

3.3. Nonnesting Packing Layout. If the necessary conditions for the nesting layout are not met, the nesting packing mode cannot form. Thus, there will be some exceptions occur as follows:

\[ \begin{align*}
\text{case 1} & : a = 0 \quad \text{and} \quad b, c, d \neq 0 \\
\text{case 2} & : b = 0 \quad \text{and} \quad a, c, d \neq 0 \\
\text{case 3} & : c = 0 \quad \text{and} \quad a, b, d \neq 0 \\
\text{case 4} & : d = 0 \quad \text{and} \quad a, b, c \neq 0 \\
\text{case 5} & : a, d = 0 \quad \text{and} \quad b, c \neq 0 \\
\text{case 6} & : b, c = 0 \quad \text{and} \quad a, d \neq 0 \\
\text{case 7} & : a = 0 \quad \text{and} \quad b, d \neq 0 \\
\text{case 8} & : b, d = 0 \quad \text{and} \quad a, c \neq 0.
\end{align*} \]

In case 1, a \(b \times d\) rectangle is packed on the sheet board as shown in Figure 4. Cases 2–4 are similar to case 1. In case 5, a vertical strip with a size of \(l \times c \times w\) and a horizontal strip with a size of \(b \times w \times l\) are put on the sheet board to generate 2 initial schemes as shown in Figures 5(a) and 6(b). Case 6 is similar to Case 5. In Cases 7 and 8, the rectangles with a size of \(b \times d\) and \(a \times c\) can be directly taken as possible candidate schemes, as shown in Figures 6 and 7.

If a, b, c, and d are all positive numbers and the nesting packing conditions are not met, Case 9 happens. In this case, there will be a big rectangle A and a small rectangle B. We first place A on the bottom-left corner of the sheet board. Then we place one B above A, and another B to the right of A as shown in Figures 8(a) and 8(b).

Afterwards, according to Figure 8(a), we first establish a vertical strip, and then a horizontal strip is created, as shown in Figure 9(a). According to Figure 8(b), we first establish a horizontal strip, and then a vertical strip is created, as shown in Figure 9(b). Then, the layouts in Figure 9 are two candidate schemes.

If the nesting packing layout is classified as Case 10, then each time we are searching for the best combinations on each side of the remains, one of those Cases will be matched. If the remaining sheet board is still very large, we need to continue the next round of packing procedure.

3.4. Dynamic Programming Algorithm Based MGPM (MGPM-DP). Dynamic programming algorithm is very
suitable and feasible for solving problems in a multi-round or divide-and-conquer way. The aforementioned dynamic programming algorithm was based on strips. Since each strip has some wasted parts, the final results may not be optimal. Here, we combined a dynamic programming algorithm with MGPM to produce a novel algorithm, namely, MGPM-DP. Formula (7) describes the state transition equation.

\[
F(L, W) = \begin{cases} 0 & \text{min}(L, W) \leq \text{min}(l, w) \\ \max[n_k + F(L_{ik}, W_{ik})] & k = 1, 2, \ldots, k. 
\end{cases}
\]

MGPM-DP is a rectangular-block-based algorithm rather than a strip-based dynamic programming algorithm. Here, an array of STRUCT structures is taken to store temporary unfinished schemes and record parent nodes in the last round, the number of packed pieces in the current round, rectangular blocks, and possible child nodes appearing in the next round. The classical dynamic algorithm relies on a time-consuming recursion mechanism and tries to shorten the computational time at the cost of memory space. Here, since the number of temporary schemes generated in each round of MGPM is unable to determine, it is easy to cause memory overflow or time limit exceedance when all tree nodes are recursively traversed. Therefore, we record cls for all the previously dealt with remains to avoid subsequently repeated combination acquiring process. Then the computational time of MGPM-DP may be further reduced.

Some elements in MGPM-DP are defined as follows:

**Plan:** temporary schemes generated by RECT

**Rnum:** ideal number of piece boards on the remains

**Num:** number of packed piece boards in the current round
Num: ideal total number of piece boards in the current scheme (Num = Num1 + Rnum)
Respace: size of the remains in the current scheme
Plancoll: set consists of schemes [Plan1; Plan2; Plan3; ...]
Mem: array for information storage

The executive mechanism of MGPM-DP goes as follows:

1. Enumerate all possible combinations of (a, b) on L to produce CL
2. Enumerate all possible combinations of (c, d) on W to produce CW
3. Perform Cartesian product of CL and CW to obtain clcw, Msize, and RECT
4. Traverse each Plan, to obtain Numli, Respacei, and Rnumi
5. Judge if Respacei can be further packed
6. Recursively apply MGPM-DP to Respacei until all final schemes are captured
7. Output the best scheme as the final solution

The flowchart of MGPM-DP is shown in Figure 10.

3.5. Branch-and-Cut-Based MGPM (MGPM-BC). Even if the dynamic programming algorithm has some advantages in the solution-seeking procedure, with the increase in problem scale, the whole procedure will be rather time-consuming. Hence, we combine MGPM with a branch-and-cut idea. First, all plans are generated according to decisive factor (2) of MGPM. Then, a variable WasteRate (formula (8)) is adopted to sort the plans in the current round. Next, remove the worst PruneRate% plans. Repeat the MGPM procedure until the best scheme is reached.

WasteRatei = \frac{L_i \times W_i - Respace_i - \sum_{k=1}^{n} \text{Sum}(rect_k)}{\sum_{k=1}^{n} \text{Sum}(rect_k)} \tag{8}

The flowchart of MGPM-BC is shown in Figure 11.

4. Computational Practices

In this part, the MGPM-DP, the MGPM-BC, the simplex method, and the strip-based dynamic programming algorithms are coded in MATLAB2016 on a desktop computer with a 16 G RAM, an Intel i5 2.4 G CPU, a 64-bit Windows 10 OS, and a 500 G HD. Here, PruneRate is 0.2.
4.1. Exploration of Computational Performances of 4 Algorithms. Here, 1200 pieces of data from an electronic manufacturing company in Pan’an Zhejiang province, China, are selected and merged into 6 datasets. Each data set has 200 pieces of data. The computational results are listed in Table 1. We adopted AN to represent an average number, AUR to denote average utilization rate, ACT to depict average computational time, and DS as the abbreviation of data sets.

It can be clearly seen in Table 1 that the average utilization rate and the average number of piece boards achieved by MGPM-DP are superior to the other 3 algorithms. Simplex Method shows an overwhelming advantage in computational time. The performance of MGPM-BC is only a little bit inferior to that of MGPM-DP in terms of AUR an AN. As for ACT, MGPM-BC does display its superiority over MGPM-DP. Although the average utilization rate of the strip-based dynamic programming algorithm is acceptable, the time consumption is substantial and not quite satisfactory.

Figure 12 shows a statistics chart of 1200 pieces of data falling into different sections of utilization rates. From Figure 12, we can find out the superiority of MGPM-DP over the other 3 algorithms as well.

4.2. Size-Fixed Sheet Board with Nonfixed Piece Boards. Here, we check the utilization rate distributions for 5 common sheet boards (1230 mm × 1030 mm, 1224 mm × 1104 mm, 1224 mm × 1054 mm, 1220 mm × 1100 mm, and 1220 mm × 1100 mm), aiming to look for ideal piece boards for each sheet board in that company. The 3D distribution charts for each sheet board are presented in Figures 13–17. According to Figures 13–17, we can learn that piece boards in each figure with convex peaks (utilization rates closest to 1) are ideal ones for that sheet board.

4.3. Size-Fixed Piece Board with Nonfixed Sheet Boards. In the real production setting, enterprises sometimes randomly set some workplates. They first pack piece boards on the workplates. Then they try to find the best packing layout of workplates on the sheet board. Here, 5 common piece board sizes (62.5 mm × 23 mm, 37 mm × 25 mm, 27 mm × 27 mm, 238.5 mm × 75 mm, 315 mm × 209.6 mm) are selected to explore the utilization rate distribution on sheet boards with \( L \) [1220 mm, 2060 mm] and \( W \) [1030 mm, 1230 mm]. The results are shown in Figures 18–22.
Material size (initially large plate size), unit plate size input

Generate CL

Generate CW

Find if the best plan in Mem

No

Generate rectangular pairs MRECT

Case1/Generate plan1

Case2/Generate plan2

Case3/Generate plan3

Judge whether the remaining area can be spliced

Yes

Update each panel plan

Select the best Plan, Insert to Mem

Yes

the best plan output

No

End

Figure 10: Flowchart of MGPM-DP.
Material size (initially large plate size), unit plate size input

Generate CL

Generate CW

Generate rectangular pairs MRECT

Case1/Generate plan1

Case2/Generate plan2

Case3/Generate plan3

Prune nodes

Case1

Case2

Judge whether the remaining area can be spliced

Yes

No

Update each panel plan output

End

Figure 11: Flowchart of MGPM-BC.
4.4. Comparison of MGPM-DP with Yuanbo Commercial Packing Software. Currently, there are many commercial packing software applications widely run in circuit board manufacturing companies. Yuanbo Software, which is apparently superior to other commercial software, is used as a comparison benchmark here. 5 experiments are performed.
Figure 14: Utilization rate distribution chart for $1224 \times 1104$ sheet board.

Figure 15: Utilization rate distribution chart for $1224 \times 1054$ sheet board.

Figure 16: Utilization rate distribution chart for $1220 \times 1050$ sheet board.
Figure 17: Utilization rate distribution chart for 1220 × 1100 sheet board.

Figure 18: Utilization rate distribution chart for 62.5 × 23 piece board.

Figure 19: Utilization rate distribution chart for 37 × 25 piece board.
Figure 20: Utilization rate distribution chart for $27 \times 27$ piece board.

Figure 21: Utilization rate distribution chart for $238.5 \times 75$ piece board.

Figure 22: Utilization rate distribution chart for $315 \times 209.6$ piece board.
Figure 23: Comparison result with piece board: 158 mm × 89 mm and sheet board: 1245 mm × 1040 mm. (a) MGPM-DP. (b) Yuanbo commercial software.

Figure 24: Comparison result with piece board: 98.5 mm × 70 mm and sheet board: 1230 mm × 1030 mm. (a) MGPM-DP. (b) Yuanbo commercial software.

Figure 25: Comparison result with piece board: 313.5 mm × 31.6 mm and sheet board: 1220 mm × 1050 mm. (a) MGPM-DP. (b) Yuanbo commercial software.
to further clarify the performance of MGPM-DP as shown in Figures 23–27. Therefore, according to Figures 23–27, MGPM-DP’s performance is clearly better than that of Yuanbo Commercial Software.

5. Conclusion

In this paper, we reduce the two-dimensional PCB board packing problem into two one-dimensional problems focusing on the best combinations on two sides of a rectangle sheet board. Then, we try to obtain the best solution by applying multiple rounds of nesting packing mode. Experimental results show that the average utilization rates of MGPM methods are closer to the theoretical optimal value than the simplex method and strip-based dynamic programming algorithm. In addition, it is of great significance for enterprises to select and set the sizes of workplates by carrying out research on size-fixed sheet boards and piece boards, respectively. Moreover, comparisons between the MGPM-DP and the Yuanbo commercial packing software further verify the advantages and feasibility of the MGPM. Our follow-up work will aim at developing circuit packing software eagerly needed by electronic enterprises engaging in the PCB manufacturing industry.

Abbreviations

PCB: Printed circuit board
PCB-P(s): PCB pieces
MGPM(s): Multi-round gradual packing method(s)
MGPM-DP: Depth priority-based MGPM
MGPM-BC: MGPM
RP-PPP(s): Rectangular PCB pieces packing problem(s)
SCP(s): Single-size cutting-stock problem(s)
BBA: Branch-and-bound algorithm.
Appendix

The pseudo code of MGPM-DP is as follows:

```
input Pnl,Pic
output plan
begin
    function [plan = Splice( Pnl,Pic )]
    flag←IfSpell(Pnl,Pic) % Judge if the remains is packable
    if flag==false
        return; % No space is available
    CL←CreatCL
    CW←CreatCW % Combinations generation
    Msize←CreatMsize(CL,CW)
    FindMem(clcw,Mem) % search existing packing layout for each clcw
    RECT←CreatRECT
    Plancoll = []
    for each RECT
        if % Case 10
            Remains←CreateSpace
            Plan←CreatNode
            Splice (Remains,Pic )
            Plancoll = [Plancoll;Plan];
        if % Case 1
            Remains←CreateSpace
            Plan←CreatNode
            Splice(Remains,Pic )
            Plancoll = [Plancoll;Plan]
        ....
        end
    end
    Plancoll = PruneNodes(plancoll) % Cut branches according to WasteRate
    for each plancoll
        ChildPlan←Splice(Remains,Pic ) % Remaining schemes enter into the next round
    Plan←ChooseBestPlan(plancoll) % Output the best scheme
end
```

The pseudo code of MGPM-BC is as follows:

```
input Pnl,Pic
output plan
begin
    function [plan = Splice( Pnl,Pic )]
    flag←IfSpell(Pnl,Pic)
    if flag == false
        return;
    CL←CreatCL,
    CW←CreatCW,
    RECT←CreatRECT,
    Plancoll = []
for each RECT
    if % Case 10
        Remains←CreateSpace
        WasteRate←CreateWasteRate
        Plan←CreatNode
        Plancoll = [plancoll;plan];
    if % Case 1
        Remains←CreateSpace
        WasteRate←CreateWasteRate
        Plan←CreatNode
        Plancoll = [plancoll;plan];
    ....
end
plancoll←PruneNodes(plancoll) % Cut branches according to WasteRate
for each plancoll
    ChildPlan←Splice(Remains,Pic ) % Remaining schemes enter into the next round
end
Plan←ChooseBestPlan(plancoll) % Output the best scheme
end
```

Data Availability

All the data are available by contacting the corresponding author through email: 187267654@qq.com.

Ethical Approval

Yes.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors’ Contributions

PFP coded the MGPM algorithms in this paper. YH is responsible for writing this paper. The GYD and HXL summarized the literature. ZYJ coded Simplex Method and dynamic programming algorithm.

Acknowledgments

The authors are grateful to all the anonymous reviewers for their suggestions and comments. Talent Project of Zhejiang Shuren University (KXJ1420601).

References

[1] Q. Zhang, Design and Implementation of PCB/CAM Secondary Development System Based on Genesis Software, University of Electronic Science and Technology, Chengdu, China, 2015.
[2] N. Lin and H. Lin, "Research and development of printed circuit boards packing software," *Printed Circuit Information*, pp. 557–562, 2009.

[3] H. Li and C. Li, "Summary of research on layout of rectangular blanks of the same size," *Software Guide*, vol. 11, no. 3, pp. 127–129, 2012.

[4] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[5] P. Agrawal, "Minimising trim loss in cutting rectangular blanks of a single size from A rectangular sheet using orthogonal guillotine cuts," *European Journal of Operational Research*, vol. 64, no. 3, pp. 410–422, 1993.

[6] M. Arslanov, "Continued fractions in optimal cutting of A rectangular sheet into equal small rectangles," *European Journal of Operational Research*, vol. 125, no. 2, pp. 239–248, 2000.

[7] Y. Cui, C. Zhang, and Y. Zhao, "Branch-and-Bound algorithm based on continuous fraction for rectangular blank layout of the same size," *Journal Of Computer-Aided Design & Computer Graphics*, vol. 16, no. 2, pp. 252–256, 2004.

[8] D. He and Y. Cui, "Dynamic recursive shearing algorithm for optimal layout of rectangular blanks of the same size," *Computer Engineering and Application*, vol. 41, no. 25, pp. 220–222, 2004.

[9] S. Yang and Y. Cui, "Layout algorithm of rectangular blanks with the same size," *Journal of Guilin University of Technology*, vol. 32, no. 4, pp. 628–630, 2012.

[10] D. Li, *Research on Packing Algorithm of the Same Size Integrated Circuit Board*, Guangxi University, Nanning, China, 2016.

[11] R. Wang, Y. Cui, Y. Cui, and T. Wang, "Accurate packing algorithm for PCB multiple-size workplate packing problem," *Forging & Stamping Technology*, vol. 44, pp. 17–23, 2019.

[12] Y. Cui, F. Li, and Q. Chen, "Dynamic programming algorithm for single workplate packing problem of integrated circuit board," *Forging & Stamping Technology*, vol. 44, pp. 23–26+67, 2019.

[13] Y. Cui, "Dynamic programming algorithms for the optimal cutting of equal rectangles," *Applied Mathematical Modelling*, vol. 29, no. 11, pp. 1040–1053, 2005.

[14] Y. Cui and Z. Liu, "T-shape homogenous block patterns for the two-dimensional cutting problem," *Journal of Global Optimization*, vol. 41, no. 2, pp. 267–281, 2008.

[15] Y. Cui and R. Zhou, "Generating Optimal cutting Patterns for Rectangular blanks of a single size," *Journal of the Operational Research Society*, vol. 53, no. 12, pp. 1338–1346, 2002.

[16] R. Pessoa, "Contributions to the Two-Dimensional Guillotine Cutting Stock Problem," 2018, https://repositorio.ufc.br/bitstream/riufc/34556/3/2017_dis_rlpessoa.pdf.

[17] J. E. Beasley, "An exact two-dimensional non-guillotine cutting tree search procedure," *Operations Research*, vol. 33, no. 1, pp. 49–64, 1985.

[18] I. Litvinchev, L. Infante, and L. Ozuna, "Approximate packing: integer programming models, valid inequalities and nesting," in *Optimized Packings With Applications*, G. Fasano and J. D. Pinter, Eds., vol. 105, pp. 187–205, Springer, Cham, Switzerland, 2015.

[19] G. Scheithauer, "Introduction to cutting and packing optimization," in *Introduction to Cutting and Packing Optimization: Problems, Modelling Approaches, Solution Methods*, G. Scheithauer, Ed., Springer, Cham, Switzerland, 2018.