Forest Chemicals Review
www.forestchemicalsreview.com
ISSN: 1520-0191
July-August 2022 Page No. 2482-2491
Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022

A Dynamic Programming Algorithm Based Improved Strip Packing Algorithm for Single-size PCB Packing Problem

Pan Peng-fei[1], Han Yi[2,*], Lyu He-xin[3], Yan Yu-bao[1]
[1]College of Computer and Artificial Intelligence, Changzhou University, Changzhou, Jiangsu 213164, China.
[2]College of Jiayang, Zhejiang Shuren University, Hangzhou, Zhejiang 310008, China.
[3]College of Innovation and Entrepreneurship, Zhejiang Shuren University, Hangzhou, Zhejiang 310008, China.
*Corresponding Author Han Yi: 187267654@qq.com.

Abstract: **-** The two-dimensional rectangular material sheet packing problem has long been a tricky one in electric industry perplexing PCB manufacturers. In order to solve single-size rectangular PCB orthogonal packing problem, a dynamic programming method based improved strip packing algorithm (DISPA) was proposed. DISPA algorithm takes into consideration the wasted area generated when a strip is packed. If the fact occurs that the wasted areas of multiple strips meet the relaxed constraints, a dynamic programming procedure is adopted to calculate the potential packing number in the remaining sheet. And the strip with the largest total packing number will be adopted as the current operational decision. Before a strip is placed, an extreme point rule will be applied to position the strip and obtain other feasible extreme points. Hence, the follow-up process can run smoothly and effectively. According to the experimental results, DISPA algorithm has higher average utilization rate and packing number than dynamic programming algorithm, simplex algorithm and minimum-waste-area algorithm.

Keywords: *Dynamic programming, PCB, Strip, Simplex algorithm, Packing.*

I. INTRODUCTION

The thorny problem that PCB manufacturers face every day is considering how and where to place multiple PCB piece boards on a large fix-sized rectangular sheet board (material), so as to obtain the most number of piece boards or the highest overall utilization rate on the sheet board[1,2]. At present, customers often hold great demands for PCB piece boards with the same size. Therefore, enterprises often adopt large-scale production mode to ensure the efficiency of cutting, picking and other processes. The packing process is of the similar operational process to cutting stock problem on a two-dimensional plane[3]. Currently, most of the related researches design the objective functions and related constraints based on the number of PCB piece boards or the utilization rate of a large sheet board. Problem solving methods related to these mathematical models are dynamic programming method[4], branch and bound method[5], continual fraction method[6], branch and bound based continual fraction method[7], two-block method[8], three-block method[8], four-block method[9] and intelligent optimization algorithms[10].

Li[10] applies dynamic programming algorithm to obtain the layout plan on each randomly generated set board (which is smaller than a whole sheet board but much larger than the PCB piece board). Then a knapsack algorithm is taken to assemble these set boards into multiple segments, and afterwards knapsack algorithm again helps to choose best segments combinations horizontally and vertically. On the basis of

Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022

literature [10], Wang et al.[11] puts forward to a dynamic programming algorithm for single-size PCB packing problem. Compared with the three-block method in literature [8], their algorithm has a higher utilization rate while consumes more time. However their dynamic algorithm has a fatal defect since they fail to consider the situations where the remaining sheet board can only be packing with PCB piece board stays horizontally or vertically. In reference [12], another dynamic programming algorithm is proposed to remove some redundant calculations and to reduce the time complexity.

With the expansion of PCB markets, strip packing mode is becoming more and more helpful for improving production and cutting efficiency. In this paper, we take the maximum utilization rate on a fix-sized sheet board as the optimization goal and propose an improved dynamic programming based strip packing algorithm (DISPA) for single-size PCB packing problem. Our algorithm was mainly composed of extreme point selection, strip packing and dynamic programming decision-making procedures, providing a novel idea for this research field.

II. MATHEMATICAL MODEL of SINGLE-SIZE PCB PACKING PROBLEM

Assuming that a large sheet board is a rectangle with a size [L,W] and a number of rectangular PCB piece boards with the same size [l,w]. The goal is to place as many as possible the PCB piece boards on the sheet board so as to the whole utilization rate reaches the maximum. The constraints require that all the piece boards can't exceed the overall size of the sheet board during packing and no overlapping is allowed among piece boards [13].

The symbolic representation is shown in Table I and the mathematical model is as follows:

Table I Symbolic Representation

Symbels	Definition
L	Length of Sheet Board
W	Width of Sheet Board
S	Remaining Sheet Board
V	Wasted Area of a Strip
P	PCB piece board
U	Area of Sheet Board
l	Length of Piece Board
w	Width of Piece Board
s	Area of Piece Board
n	Maximum Number of Packed PCB Piece Boards
a	Utilization Rate of Sheet Board

Mathematical model:

$$MAX\ \mathrm{a} = \frac{\mathrm{n} \times \mathrm{s}}{U} \times 100\% \tag{1}$$

$$S.t.$$

$$\bigcap P_i = \varnothing \quad i = 1,2,3,\ldots,n \tag{2}$$

$$\bigcup P_i \subseteq U \quad i = 1,2,3,\ldots,n \tag{3}$$

Formula (1) indicates that the objective function is to maximize the utilization rate of sheet board; formula (2) shows that there cannot be overlapping among piece boards; formula (3) requires that multiple piece boards are not allowed to exceed the whole range of a sheet board [14].

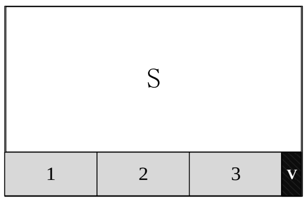
III. DISPA

DISPA combines strip packing idea with dynamic programming method. Firstly, the extreme point method is introduced to find out the starting position of the packing process. According to the minimum wasted area of four strips, the strips which satisfy the relaxed area constraint (half the area of a PCB piece board) are kept. If more than one strip is kept, the dynamic planning process is activated to calculate the potential packing number of PCB unit boards on the remaining sheet board. Add the number of piece boards contained in the strip and the potential number up and the strip with the maximum total number will be the current packing strip. If several strips are having the same total number, randomly choose one strip among them. And if the wasted areas of the four strips do not meet the relaxation area constraint, the strip with the smallest wasted area is selected as the current packing strip for the current step. After the strip packing process, extreme point method is used to find out the next starting position, and the strip selection process is repeated until the end.

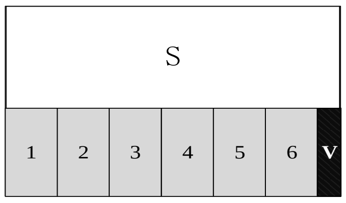
3.1 Strip Packing

Definition of stripes:

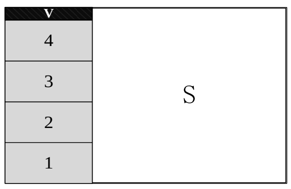
A belt of material with width w or l in horizontal or vertical direction[15]. Fig 1 shows 4 types of strips.



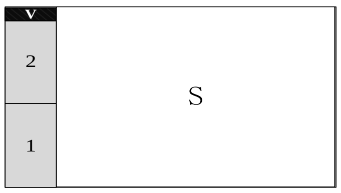
(a) horizontal strip with *w* as width



(b) horizontal strip with *l* as width



(c) vertical strip with l as width



(d) vertical strip with w as width

Fig 1. 4 Strips

Strip packing process:

1. determining the starting position of a strip with extreme point method;
2. comparing the wasted areas of 4 strips and keep the strips with wasted areas being less than half of the PCB piece board;
3. dynamic programming method is adopted to obtain the total number;
4. choose the best strip;
5. packing the best strip on the sheet board and generate extreme points for the afterwards procedure, goto step 1.

3.2 Improved Dynamic Planning Algorithm

Dynamic programming algorithm has advantages in solving multi-stage decision-making optimization problems, which can be decomposed into sub-problems with the same structure, and those sub-problems have certain correlations and non-aftereffects [16,17]. This paper modified the dynamic programming method in literature [10-12,15], in which the state transition equation is shown in formula (4) and the termination condition is shown in formula (5).

$$F(L,W) = \max\{F(L,W-l) + \lfloor L/w \rfloor, F(L-l,W) + \lfloor W/w \rfloor\} \tag{4}$$

$$F(L,W) = 0, \quad if\ L \le w or W \le w \tag{5}$$

When formulas (4) and (5) are used to pack strips, their algorithms only consider two strips as shown in Fig 1(b) and 1(c). In addition, since the termination condition is not further narrowed down, it is easy to capture wrong results. In view of the above problems, we consider 4 strips in the dynamic programming process, and add measures for situations where the remaining sheet board can only be horizontally or vertically packed with w-width strips.

State transition equations of DISPA are shown in Formula (6-9):

$$F(L,W) = 0, \quad if\ L < w or W < w \tag{6}$$

$$\begin{aligned} F(L,W) = \max\{&F(L,W-w) + \lfloor L/l \rfloor, F(L,W-l) + \lfloor L/w \rfloor, \\ &F(L-w,W) + \lfloor W/l \rfloor, F(L-l,W) + \lfloor W/w \rfloor\} \quad if\ L \ge l\, and\ W \ge l \end{aligned} \tag{7}$$

$$\begin{aligned} F(L,W) = \max\{&F(L,W-w) + \lfloor L/l \rfloor, \\ &F(L-l,W) + \lfloor W/w \rfloor\} \quad if\ L \ge l\, or\ W < l \end{aligned} \tag{8}$$

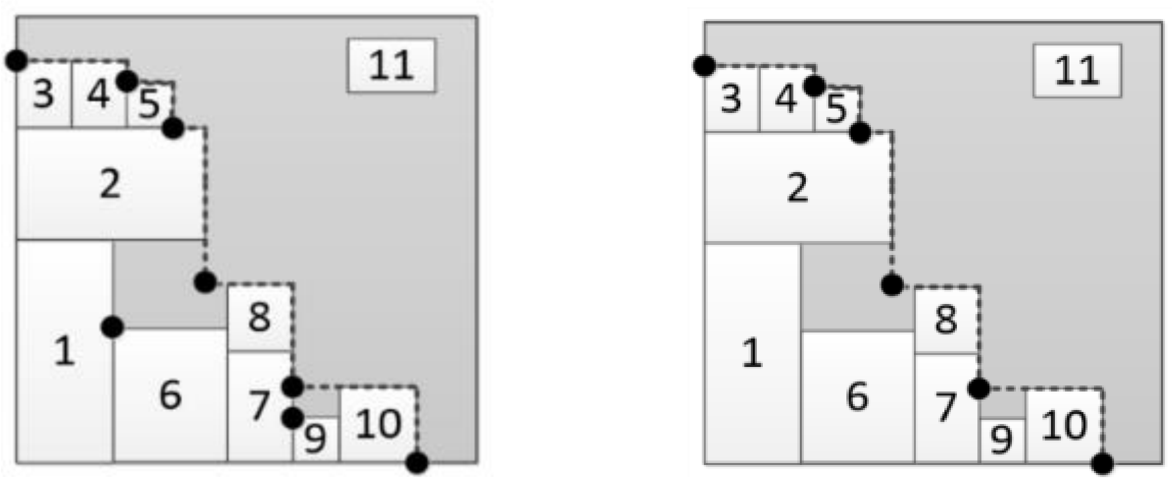
$$\begin{aligned} F(L,W) = \max\{&F(L-w,W) + \lfloor W/l \rfloor, \\ &F(L,W-l) + \lfloor L/w \rfloor\} \quad if\ L < l\, or\ W \ge l \end{aligned} \tag{9}$$

In DISPA algorithm, the relaxed wasted area constraint rule is adopted. Hence, more strips are selected so as to avoid the optimization searching process from falling into the local optimum as much as possible. Of course, in during the process of dynamic programming method, some calculation time will be consumed.

3.3 Extreme Point Method

The concept of extreme point was first put forward by Crainic et al. [18] when they were studying the three-dimensional bin-packing problem. According to the definition in literature [18], the extreme point is the intersection of three planes projected from the upper right corner point of an object with the coordinate axes X, Y and Z. After the extreme points are determined, only those feasible extreme points will be selected as the possible placement points for the coming objects.

Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022



(a) extreme points (b) corner points

Fig 2. Extreme Points and Corner Points

Literature [18] gives an example showing what the extreme points are in a two-dimensional bin-packing problem as shown in Fig 2(a) and mentions the concept of corner points as shown in Fig 2(b). According to Fig 2(b), we can infer that the corner point is the intersection point of the envelope edge lines of all the placed small rectangles [19]. Although the corner point method can simplify the process of determining the placement positions for the coming rectangles, it easily ignores the possible rectangle placement positions such as the intersection of rectangle 1 and rectangle 6 and the intersection of rectangle 7 and rectangle 9 in Fig 2(a).

Even if reference [18] presented what the extreme points looked like in the two-dimensional bin-packing problem, it did not explicitly depict how these extreme points were generated and how to recognize feasible extreme points. Therefore, we clearly clarified the generation method of extreme points and the elimination method for infeasible extreme points and integrated two methods into DISPA algorithm.

In extreme point method, we define the extreme points as the intersections of two rays shot out from the upper right corner of a rectangle with *X*-axis, *Y*-axis and other afore-generated rays. When extreme points are generated, a small square with a size (*w*,*w*). We test each extreme point with this small rectangle to see if the intersected area is $w \times w$. If not, we eliminate the current extreme point since it is not suitable for other rectangles.

3.4 Example of Extreme Point Method

Let's suppose there are four different piece boards and one sheet board. Take the sheet board as a large rectangle and the coordination of the bottom-left point is (0,0). During the running process of extreme point method, the leftist-and-bottommost point will be viewed as the current fittest rectangle placement point for each small rectangular piece board. Here, we use EP to represent the set of extreme points, XX to record rays paralleling with *Y*-axis and YY to capture rays paralleling with *X*-axis. The steps are like:

1. Sheet board size (150,100), piece board size [1=(40,60), 2=(75,35), 3=(70,30), 4=(30,25)];
2. EP=[(0,0)], XX=[0], YY=[0], and a small square ww with a size (25,25);
3. Assuming that the placement order of rectangles is 1-2-3-4, place the first rectangle as shown in Fig 3(a) with three new extreme points (0,60), (40,0) and (40,60) added to EP = [(0,0),(0,60), (40,0), (40,60)], and update XX=[0,40] and YY=[0,60];

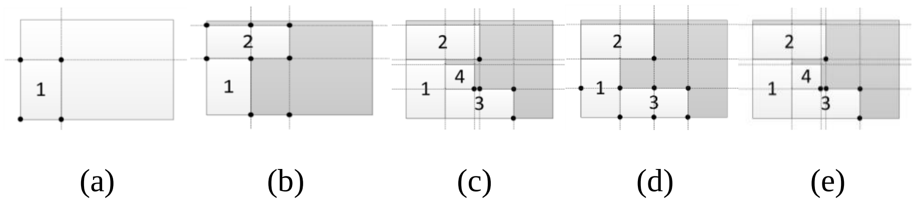


Fig 3. Extreme Point Method

4. Use square ww to eliminate the infeasible extreme point (0,0), because ww has an overlapping area with rectangle 1;
5. Sort EP into [(0,60),(40,0),(40,60)], and select the leftmost and lowest extreme point (0,60) for rectangle 2;
6. After rectangle 2 is placed at (0,60), the new extreme points are added into EP = [(0,60),(40,0),(40,60),(0,95),(40,95),(75,0),(75,60),(75,95)] and update XX=[0,40,75] and YY=[0,60, 95];
7. Use small square ww to eliminate infeasible extreme points (0,60), (40,60), (0,95), (40,95) and (75,95);
8. Sort EP into [(40,0),(75,0),(75,60)], and select the leftmost and lowest extreme point (40,0) for rectangle 3;
9. After rectangle 3 is placed at (40,0), new extreme points are added into EP = [(40,0), (75,0), (75,60), (0,30),(40,30),(75,30),(110,0),(110,30)], update XX=[0,40, 75,110] and YY=[0,30,60,95];
10. Use small square ww to eliminate infeasible extreme points (0,30), (40,0) and (75,0);
11. Order EP = [(40,30), (75,30), (75,60),(110,0),(110,30)] and select the leftmost and lowermost extreme point (40, 30) for rectangle 4;
12. After rectangle 4 is placed at (40,30), the new extreme points are added into EP=[(40,30), (75,30), (75,60),(110,0),(110,30),(0,55),(40,55),(70,0),(70,30),(70,55)], update XX=[0,40,70,75,110] and YY=[0, 30,55,60,95];
13. Use ww to eliminate infeasible extreme points (0,55), (40,30), (40,55), (70,0) and (70,55);
14. EP = [(70,30), (75,30), (75,60), (110,0), (110,30)], as shown in Fig 3(e).

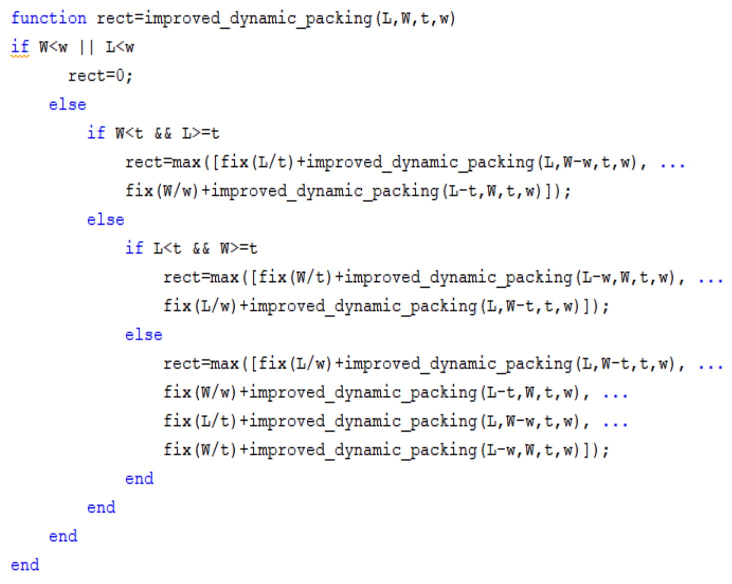
3.5 Executive Steps of DISPA

The executive steps of DISPA algorithm are as follows:

1. Set the coordinates of the leftmost and the lowest point of the sheet board as (0,0), with the length side L parallels to X-axis and the width side W parallels to Y-axis;
2. Initialize extreme point set EP=[(0,0)], XX=[(x=0)] and YY=[(y=0)];
3. Select the leftmost and bottommost extreme point from the EP, and check whether the extreme point is a feasible position for placing a strip;
4. Record the wasted areas of four strips;
5. Keep the strip with the wasted areas less than half of the piece board;
6. If there are multiple strips restored, the dynamic programming algorithm is adopted to calculate the total number;
7. Find out the strip with the largest total number and pack it on the sheet board. User extreme point method to produce possible extreme points;
8. Place the square ww at each extreme point in EP in turn to remove infeasible points.
9. Update the EP set, XX and YY;

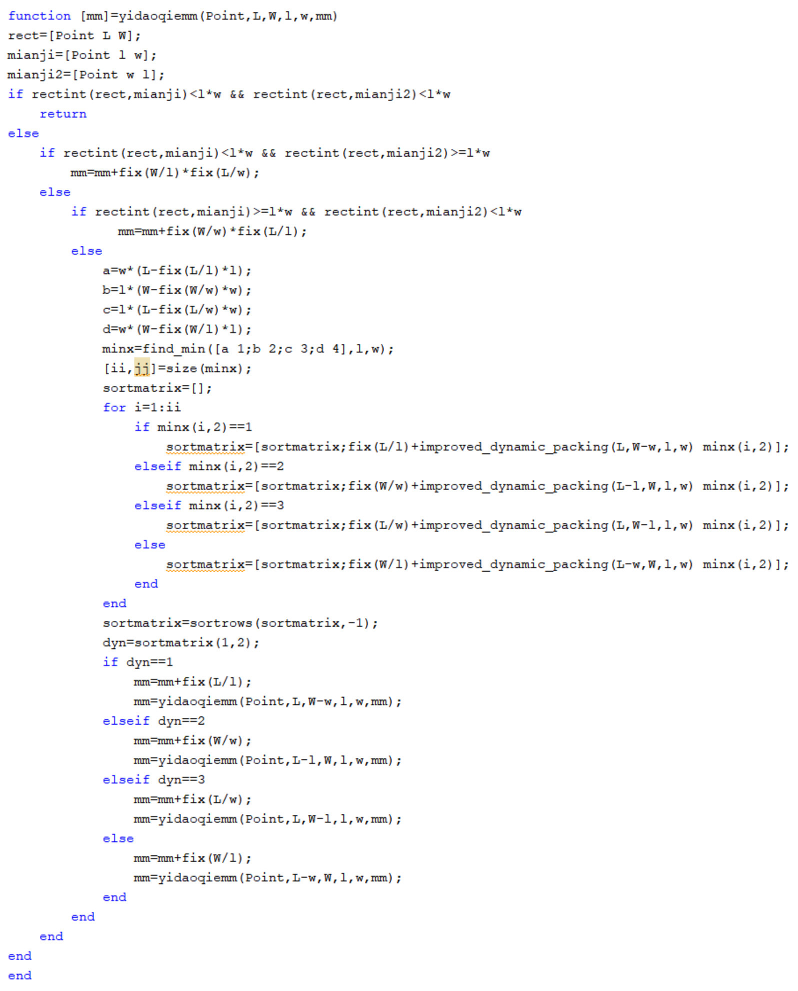
10. Repeat steps 2-9 until the remaining sheet board can no longer be packed.

3.6 Some Codes of DISPA



```
function rect=improved_dynamic_packing(L,W,t,w)
if W<w || L<w
      rect=0;
    else
        if W<t && L>=t
            rect=max([fix(L/t)+improved_dynamic_packing(L,W-w,t,w), ...
            fix(W/w)+improved_dynamic_packing(L-t,W,t,w)]);
        else
            if L<t && W>=t
                rect=max([fix(W/t)+improved_dynamic_packing(L-w,W,t,w), ...
                fix(L/w)+improved_dynamic_packing(L,W-t,t,w)]);
            else
                rect=max([fix(L/w)+improved_dynamic_packing(L,W-t,t,w), ...
                fix(W/w)+improved_dynamic_packing(L-t,W,t,w), ...
                fix(L/t)+improved_dynamic_packing(L,W-w,t,w), ...
                fix(W/t)+improved_dynamic_packing(L-w,W,t,w)]);
            end
        end
    end
end
```

Fig 4. Codes of Improved Dynamic Programming Algorithm



```
function [mm]=yidaoqiemm(Point,L,W,l,w,mm)
rect=[Point L W];
mianji=[Point l w];
mianji2=[Point w l];
if rectint(rect,mianji)<l*w && rectint(rect,mianji2)<l*w
    return
else
    if rectint(rect,mianji)<l*w && rectint(rect,mianji2)>=l*w
        mm=mm+fix(W/l)*fix(L/w);
    else
        if rectint(rect,mianji)>=l*w && rectint(rect,mianji2)<l*w
            mm=mm+fix(W/w)*fix(L/l);
        else
            a=w*(L-fix(L/l)*l);
            b=l*(W-fix(W/w)*w);
            c=l*(L-fix(L/w)*w);
            d=w*(W-fix(W/l)*l);
            minx=find_min([a 1;b 2;c 3;d 4],l,w);
            [ii,jj]=size(minx);
            sortmatrix=[];
            for i=1:ii
                if minx(i,2)==1
                    sortmatrix=[sortmatrix;fix(L/l)+improved_dynamic_packing(L,W-w,l,w) minx(i,2)];
                elseif minx(i,2)==2
                    sortmatrix=[sortmatrix;fix(W/w)+improved_dynamic_packing(L-l,W,l,w) minx(i,2)];
                elseif minx(i,2)==3
                    sortmatrix=[sortmatrix;fix(L/w)+improved_dynamic_packing(L,W-l,l,w) minx(i,2)];
                else
                    sortmatrix=[sortmatrix;fix(W/l)+improved_dynamic_packing(L-w,W,l,w) minx(i,2)];
                end
            end
            sortmatrix=sortrows(sortmatrix,-1);
            dyn=sortmatrix(1,2);
            if dyn==1
                mm=mm+fix(L/l);
                mm=yidaoqiemm(Point,L,W-w,l,w,mm);
            elseif dyn==2
                mm=mm+fix(W/w);
                mm=yidaoqiemm(Point,L-l,W,l,w,mm);
            elseif dyn==3
                mm=mm+fix(L/w);
                mm=yidaoqiemm(Point,L,W-l,l,w,mm);
            else
                mm=mm+fix(W/l);
                mm=yidaoqiemm(Point,L-w,W,l,w,mm);
            end
        end
    end
end
end
```

Fig 5. Codes of DISPA

Forest Chemicals Review
www.forestchemicalsreview.com
ISSN: 1520-0191
July-August 2022 Page No. 2482-2491
Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022

IV. COMPUATIONAL EXPERIMENTS

DISPA is implemented with matlab 2016 on a Windows 10 operating system having 8G RAM and 1GHz CPU. The experimental data are selected from 150 pieces of data in an electric production enterprise. The data is divided into 5 sets (each data set contains 30 pieces of data). Simplex method, minimum area method, dynamic programming method and DISPA algorithm are compared in terms of computational time, utilization rate and packing number of PCB piece boards and the results are listed in Table II. We also compare DISPA algorithm with Yuanbo commercial packing software which is popular with circuit board manufacturing industry at present. The comparison results are shown in Table III and Fig 6.

We can clearly see from Table II that the average utilization rates and average packing numbers of DISPA algorithm for five data sets are higher than those of the other three algorithms. In terms of calculation time, simplex method and minimum area method have superior advantages.

According to the comparative analysis in Table III and the packing results in Fig 6, DISPA has superior or equal results to those of Yuanbo commercial packing software.

Table II Computational Results of 4 Algorithms

data	Simplex Method			Minimum-wasted-area			Dynamic Programming			DISPA		
	AG.U	AG.N	AG.T	AG.U	AG.N	AG.T	AG.U	AG.N	AG.T	AG.U	AG.N	AG.T
1	91.54%	27.87	1	91.4%	27.8	**0.0003**	91.07%	27.73	2.1	**92.36%**	**28.13**	0.1238
2	93.18%	32.37	1	92.91%	32.27	**0.0003**	91.66%	31.9	4.7	**93.35%**	**32.43**	0.3717
3	92.34%	29.43	1	92.28%	29.43	**0.0003**	90.47%	28.8	1.42	**93.06%**	**29.67**	0.1223
4	92.02%	33.3	1	92.41%	33.5	**0.0003**	91.92%	33.3	2.77	**93.24%**	**33.77**	0.4146
5	92.24%	28.27	1	92.54%	28.3	**0.0003**	91.9%	28.1	0.9185	**93.45%**	**28.57**	0.1054

Note: AG.U represents average utilization rate(%), AG.N represents average packing number, AG.T denotes average computational time (s)

Table III Comparative results of DISPA with Yuanbo Software

Data	*L*(mm)	*W*(mm)	*l*(mm)	*W*(mm)	DISPA	Yuanbo
1	1230	1030	255	155	**96.71%**	87.35%
2	1230	1030	306.8	166.5	**96.77%**	**96.77%**
3	1230	1030	259.25	160	**91.68%**	88.40%
4	1230	1030	300	127	**96.23%**	**96.23%**
5	1240	1040	194	133	**94.03%**	90.03%
6	1230	1030	265	169	**95.45%**	**95.45%**
7	1240	1040	250	111	**96.83%**	94.68%
8	1240	1040	240	175	**94.45%**	**94.45%**
9	1240	1040	220	126	**96.73%**	94.58%
10	2060	1230	254	210	**92.63%**	**92.63%**

Forest Chemicals Review
www.forestchemicalsreview.com
ISSN: 1520-0191
July-August 2022 Page No. 2482-2491
Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022

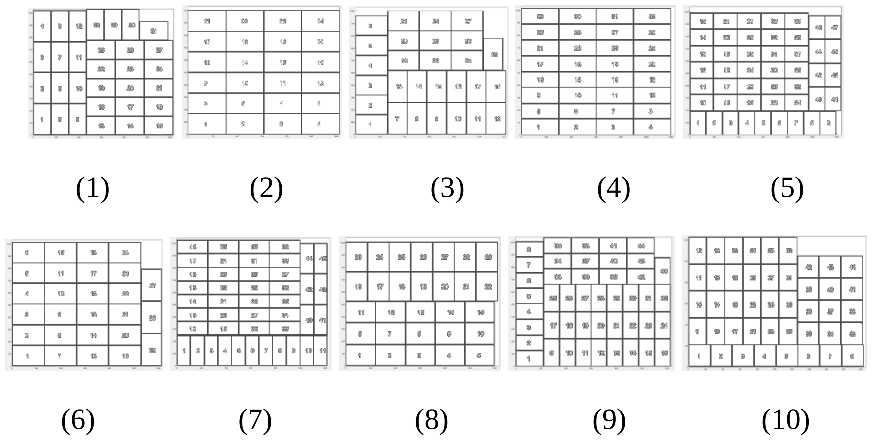


Fig 6. Packing Results of DISPA for 10 Cases

V. CONCLUSION

In this paper, an improved strip packing algorithm is designed to solve the two-dimensional rectangular single-size PCB packing problem. The relaxation idea is introduced into DISPA algorithm when considering the wasted area of each strip to increases the probability of catching the optimal solutions. Through dynamic programming method, the number of potential packing number in the remaining sheet board can be obtained to guide the packing procedure of strips in each step. Through the test with 150 data from an electronic factory in Pan 'an, Zhejiang province, we verified that our DISPA is feasible and effective for single-size PCB packing problem.

VI. FOUNDATION

This paper is granted by Talent Project of Zhejiang Shuren University (1420601).

REFERENCES

[1] Zhang Q (2015) Design and implementation of PCB/CAM secondary development system based on Genesis software. University of Electronic Science and Technology of China, Chengdu

[2] Lin N, Lin H (2009) Research and development of printed circuit boards. Printed Circuit Information (S1): 557-562

[3] Li H, Li C (2012) Review of the same size rectangular blank layout research. Software Guide 11(3): 127-129

[4] He D, Cui Y (2004) Dynamic recursive shearing algorithm for optimal layout of rectangular blanks of the same size. Computer Engineering and Application (25): 220-222

[5] Arslanov M (2000) Continued fractions in optimal cutting of a rectangular sheet into equal small rectangles. European Journal of Operational Research 125(2): 239-248

[6] Agrawal P (1993) Minimising trim loss in cutting rectangular blanks of a single size from a rectangular sheet using orthogonal guillotine cuts. European Journal of Operational Research 64(3): 410-422.

[7] Cui Y, Zhang C, Zhao Y (2004) A continued fraction branch and bound algorithm for rectangular blank layout with the same size. Journal of Computer Aided Design and Graphics 16(2): 252-256

[8] Mhand H (2001) Exact Algorithms for Large-Scale Unconstrained Two and Three Staged Cutting Problems. Computational Optimization and Applications 18(1): 63-88

[9] Pan W, Fan Z, Huang M (2022) A four-block layout algorithm for unconstrained two dimensional plate guillotine cutting problem of rectangular items. Control and Decision 37(5): 1211-1219

Forest Chemicals Review
www.forestchemicalsreview.com
ISSN: 1520-0191
July-August 2022 Page No. 2482-2491
Article History: Received: 10 April 2022 Revised: 18 May 2022 Accepted: 30 June 2022 Publication: 25 July 2022

[10] Li D (2016) Research on cutting algorithm of the same size integrated circuit board. Guangxi University, Nanning
[11] Wang R, Cui Y, Cui Y, et al (2019) Accurate algorithm for single blanking of PCB multi-working board size. Forging Technology 44(3): 17-23
[12] Cui Y, Li F, Chen Q (2019) Dynamic programming algorithm for single blanking of integrated circuit boards. Forging Technology 44(9): 23-26+67
[13] Zhang Y, He Y (2015) Research on PCB combination and design [J]. Guangdong Science and Technology, 24(12): 43-44
[14] Kan H, Hu X (2020) Cutting stock algorithm of integrated circuit board based on genetic algorithm. Computer and Network 46(7): 58-60
[15] Yang S, Cui Y (2012) Layout algorithm of rectangular blanks with the same size. Journal of Guilin University of Technology 32(4): 628-630
[16] Rao F, Rao Y, Li W (2015) A new algorithm for solving PCB packing. Manufacturing Automation 37(10): 26-28
[17] Qin G, Qiu G, Wang K, Huang X (2022) Blanking algorithm of single rectangular piece coil based on multi-level layout. Forging Technology 47(2): 73-77
[18] Crainic T, Perboli G, Tadei R (2008) Extreme Point-Based Heuristics for Three-Dimensional Bin Packing. INFORMS Journal on Computing 20(3): 368-384
[19] Martello S, Pisinger D, Vigo D (2000) The Three-Dimensional Bin Packing Problem. Operations Research 48(2): 256-267
[20] Chen Y, Ning X, Hu X, Wang K (2021) T-shaped cutting algorithm for circular parts based on the best breakpoint. Journal of Chongqing Normal University (Natural Science Edition) 38(6): 56-62