An Efficient Ant Colony Optimization for Two-dimensional Bin Packing Problem with Defect Issue

Hao-Chun Lu, Yi-Siao Wan, and Yu-Ting Shueh
Department of Information Management, Fu Jen Catholic University, Taiwan.
076135@mail.fju.edu.tw

ABSTRACT
In various industries, manufacturers must cut rectangular items from stock sheets. In the actual setting, stock sheets inevitably contain some defects, and manufacturers have to avoid cutting the products which contain the defects (i.e., TFT–LCD industry, e-paper); therefore, the two-dimensional bin packing problems with defect (2DBPPWD) emerges. However, no effective method for solving 2DBPPWD in the current literature has been developed. To address the research gaps and industrial requirements, this study proposes an effective approach that integrates ant colony optimization with the flawless corner space placement algorithm to solve 2DBPPWD. Finally, we compare the proposed approach with the traditional approach (genetic algorithm integrated with bottom–left placement) to solve C21 (Hopper & Turton, 2001) and N13 (Burke et al., 2004) problems. Computational results demonstrate that our proposed approach can solve 2DBPPWD effectively and significantly outperform those of current approaches.

KEYWORDS
Ant colony optimization, two-dimensional bin packing problems, defect, flawless corner space placement algorithm.

1. INTRODUCTION
Bin packing problems are a classic optimal problem in operations research. These problems have a similar logical structure that can be considered as a set of rectangular items being arranged on a set of bins. It has been studied in various applications in varying industries, i.e., TFT-LCD substrates cutting problem[1], thin-film transistor liquid crystal panel cutting problem [2], steel coil cutting problem [3], steel bridge construction [4], crosscutting rectangular products from wood stocks [5], reel and sheet cutting at a paper mill [6], the relative algorithms [7], etc. These goals are producing items from the stock sheets in order to maximize material utilization and minimize wastage. This optimal problem is known as a two-dimensional bin packing problem (2DBPP) in literature [8], and there are many methods can solve 2DBPP well. For example, Jakobs [9] designed the well-known genetic algorithm (GA) integrated bottom-left placement (BLP) algorithm, Leung et al. [10] proposed a mixed simulated annealing (SA) integrated with GA for improving the executing time, Liu et al. [11] proposed particle swarm optimization (PSO) integrated bottom-left fill placement (BLFP) algorithm to solve 2DBPP, etc.

In fact, the stock sheets inevitably contain some defects and manufacturers have to avoid the defects for cutting the products in the real world. As a result, it arise the two-dimensional bin packing problems with defect (2DBPPWD) shown in Figure 1. The current methods in literates are qualified for solving the 2DBPP; however, there is no method has the ability to solve the 2DBPPWD effectively. For the research gaps and industrial requirements, this study proposes an effective approach that integrates ant colony optimization (ACO) with flawless corner space placement (FCSP) algorithm to solve the 2DBPPWD.
The remainder of this study is organized as follows. Section 2, classical placement algorithms are reviewing. Section 3 develops the FCSP algorithm for conquering the bin with defect issue. Section 4, we present how to integrate ACO with FCSP to solve the 2DBPPWD. In Section 5, various numerical experiments demonstrate the advantages of our proposed method. Section 6 contains some concluding remarks.

2. PLACEMENT ALGORITHM REVIEW

In this section, we use existing popular placement algorithms to pack items into the bins with defects to demonstrate the inappropriateness and inefficiencies of these algorithms. In applying the current popular placement algorithms, we randomly generate some defects on the bin and treat the defects as rectangular items fixed on the bin. Then, it is easy to understand how to place the items which never overlap with any other items existing on the bin.

2.1. Bottom Left Placement Algorithm

Bottom Left placement (BLP) algorithm is a well-known algorithm that takes as input a list of rectangular items and places each item in turn into bin. It firstly places the item in the top-right location and makes successive moves of sliding it as far down and left as possible. Figure 2 indicates the inefficiencies of BLP algorithm in the defective case. The black squares are representing the defects on the bin. However, BLP algorithm makes successive moves of sliding while the defects will block the sliding movements of item as downward or leftward.

At first, item A will be placed in the top right of bin, and it move downward until blocked by the defect 1; therefore, item A move leftward until blocked by the defect 2. Item A cannot move downward or leftward anymore now; then, item B will be placed sequentially. As showing in Figure 2, item B cannot be placed in this bin by BLP algorithm. Figure 3 shows the intuitive way of placing items into same bin shown in Figure 2. Base on the packing result by BLP algorithm, it is clearly show that BLP algorithm is inefficient while dealing with defective case.

2.2. Bottom Left Fill Placement Algorithm

The Bottom Left Fill placement (BLFP) algorithm is a popular well-known algorithm that packs the set of items into the bin with another placing strategy. In the traditional 2DBPP, the performance of BLFP algorithm is superior to BLP algorithm. The BLFP maintains a list of location points in a bottom-left ordering to indicate where the empty areas may be placed items. When placing an item, the BLFP algorithm starts with the lowest and leftmost point for checking whether the item can be placed without overlapping any placed items. However, BLP algorithm makes successive moves of sliding while the defects will block the sliding movements of item as downward or leftward.
Hopper and Turton[12] found that BLFP algorithm outperforms BLP algorithm by up to 25%. Figure 4 shows the items placed into the bin with defects by BLFP algorithm. According to the placing rules of BLFP algorithm, it only considers the item can be placed with the insert point without overlapping any placed items, but it is no idea to treat how to place the item with justifying insert point without overlapping any defect. Because BLFP algorithm cannot deal with the defects on the bin, it clearly demonstrates that BLFP algorithm is inappropriate for dealing with 2DBPP with defects. Maybe BLFP algorithm can shift the item upward or rightward for avoiding the overlapping areas with defects; however, those shifting steps will generate many and complex strategies for considering the situations of many defects spreading on the bin and the follow-up related problems.

Combining BLFP with shifting strategy is not the original BLFP algorithm; therefore, it can be treated as a best-fit strategy and is complex enough to develop the specially algorithm, flawless corner space placement (FCSP) algorithm, for dealing with 2DBPPWD.

3. PROPOSES FLAWLESS CORNER SPACE PLACEMENT ALGORITHM

This section proposes an FCSP algorithm to deal with 2DBPPWD. When try to place a rectangular item into a bin, the FCSP algorithm only considers the available rectangular spaces which are tracked by a set of corner spaces (CSs). Each CS contains the bottom-left coordinate, width, length of unused rectangular space, and CS can be overlapped with other CS existing on the same bin. CSs are core components in FCSP algorithm. If FCSP algorithm chooses an appropriate CS for placing the rectangular item, it will split the remaining space of the placed CS into several new CSs that can contain other small rectangular items. FCSP algorithm also merges small CSs to form a larger CS if they can be merged. This merging step improves the utility of any unused space on the bin and reduces the number of fragmented CS generated by the splitting process of FCSP. The above process continues until no rectangular items needed be placed. The main processes of FCSP are shown in Figure 5 and describe as follows.

Figure 5. Processes of FCSP

(1) Choose: First, we search all CSs that can fit the target item and choose the fittest CS from CSs to be placed the target item for minimal waste and reducing unused fragmented spaces. For example, there are 3 CSs (C1, C2, and C3) that can fit the item A in Figure 6. After evaluating the choose process, FCSP algorithm will place item A in C1. If the placed area in C1 contains defects, FCSP algorithm will continuously move the item upward or rightward until item can avoid the defects.

(2) Split: After placing item into a CS, the remaining unused space in placed CS will be split into multiple new CSs. According to the location of item placing in CS, a placed CS can be split at most four CSs. Figure 7 describes one of the splitting situations.
Figure 7. One of the splitting situations

In Figure 7, the grey area denotes the placing item, Item A, and the white area is the remaining unused space in placed CS. FCSP algorithm considers all four edges of placed item to decide how to split the remaining unused space in placed CS.

Step 1: In Figure 8, if the top edge of Item A, a, is inside the placed CS, FCSP algorithm splits the topCS off from placed CS where topCS is type 1 CS and ◆ represents the CS of topCS.

Step 2: In Figure 9, if the right edge of Item A, d, is inside the placed CS, FCSP algorithm splits the rightCS off from placed CS where rightCS is type 2 CS and ▲ represents the CS of rightCS.

Step 3: Because the bottom edge of Item A, b, is not inside the placed CS, FCSP algorithm dose not splits any CS.

Step 4: Because the left edge of Item A, c, is not inside the placed CS, FCSP algorithm dose not splits any CS.

After considering all edges of Item A, Figure 10 demonstrates that the original placed CS is removed and the new CSs, topCS and rightCS, will be inserted to the set of CSs.

Adjust: There are two kinds of adjusting, that is, type 1 adjusting and type 2 adjusting. Type1 adjusting is horizontally extending for type1 CS, and type2 adjusting is vertically extending for type2 CS. Figure 11 shows the adjusting of type 1 for the two type1 CSs, C1 and C2. Because C2 completely contains C1, the size of C1 should be extended to the same as size of C2.

Figure 8. Split the topCS off

Figure 9. Split the rightCS off

Figure 10. All CSs

(4) Merge: After finishing all adjustings, there maybe exists some CSs which have the same coordinate, width, and height. In this process, FCSP algorithm only keeps one of the CSs which have the same properties.

4. INTEGRATING ACO WITH FCSP

In traditional ACO, ants represent traveller to visit each city one by one until all cities have been visited. In this study of 2DBPPWD, we use item ID represent the city and the reciprocal of item’s height as distance. Therefore, we will get a sequence of item ID when ant completely visited all cities. After the sequence of item ID placed by FCSP, FCSP algorithm can evaluate the fitness of the sequence for updating the pheromones on the trail.

5. NUMERICAL EXPERIMENTS

We use the C21 problems [13] and the N13 problems [14] as benchmark to compare the performances between ACO-FCSP and GA-BLP. Because the C21 and N13 do not have defects, we add some defects into bin in this experiment. The executing program is coded by C# with Visual Studio 2012, and all experiments were run on PC with Intel® Core™2 Duo E6320 Processor and 4GB RAM.

Both of GA-BLP and ACO-FCSP cannot obtain any result of N13 within 10 hours; therefore, N13 does not compare in this study. Table 1 demonstrates that the proposed...
ACO-FCSP can obtain better solution in every case except N13, it is clear to show that ACO-FCSP is more effective and efficient than GA-BLP.

Table 1. Experiment Result

<table>
<thead>
<tr>
<th>Problem</th>
<th>No of item</th>
<th>Executing time(s)</th>
<th>GA-BLP</th>
<th>ACO-FCSP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Best</td>
<td></td>
<td>Best</td>
</tr>
<tr>
<td>C1P1</td>
<td>16</td>
<td>32</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>C1P2</td>
<td>17</td>
<td>34</td>
<td>38</td>
<td>24</td>
</tr>
<tr>
<td>C1P3</td>
<td>16</td>
<td>32</td>
<td>57</td>
<td>23</td>
</tr>
<tr>
<td>C2P1</td>
<td>25</td>
<td>50</td>
<td>41</td>
<td>17</td>
</tr>
<tr>
<td>C2P2</td>
<td>25</td>
<td>50</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>C2P3</td>
<td>25</td>
<td>50</td>
<td>43</td>
<td>17</td>
</tr>
<tr>
<td>C3P1</td>
<td>28</td>
<td>56</td>
<td>52</td>
<td>33</td>
</tr>
<tr>
<td>C3P2</td>
<td>29</td>
<td>58</td>
<td>58</td>
<td>35</td>
</tr>
<tr>
<td>C3P3</td>
<td>28</td>
<td>56</td>
<td>39</td>
<td>33</td>
</tr>
<tr>
<td>C4P1</td>
<td>49</td>
<td>98</td>
<td>79</td>
<td>67</td>
</tr>
<tr>
<td>C4P2</td>
<td>49</td>
<td>98</td>
<td>83</td>
<td>67</td>
</tr>
<tr>
<td>C4P3</td>
<td>49</td>
<td>98</td>
<td>85</td>
<td>67</td>
</tr>
<tr>
<td>C5P1</td>
<td>73</td>
<td>146</td>
<td>159</td>
<td>99</td>
</tr>
<tr>
<td>C5P2</td>
<td>73</td>
<td>146</td>
<td>163</td>
<td>100</td>
</tr>
<tr>
<td>C5P3</td>
<td>73</td>
<td>146</td>
<td>158</td>
<td>100</td>
</tr>
<tr>
<td>C6P1</td>
<td>97</td>
<td>194</td>
<td>159</td>
<td>131</td>
</tr>
<tr>
<td>C6P2</td>
<td>97</td>
<td>194</td>
<td>170</td>
<td>127</td>
</tr>
<tr>
<td>C6P3</td>
<td>97</td>
<td>194</td>
<td>167</td>
<td>128</td>
</tr>
<tr>
<td>C7P1</td>
<td>196</td>
<td>392</td>
<td>316</td>
<td>276</td>
</tr>
<tr>
<td>C7P2</td>
<td>197</td>
<td>394</td>
<td>366</td>
<td>255</td>
</tr>
<tr>
<td>C7P3</td>
<td>176</td>
<td>352</td>
<td>327</td>
<td>263</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS
This study proposes an efficient ACO that incorporates a novel placement algorithm, FCSP, to solve 2DBPPWD effectively. The results of computational experiments demonstrate that our proposed approach can solve 2DBPPWD effectively and significantly outperforms the current approaches.

REFERENCES


