GENERATING PALLET LOADING PATTERNS: A SPECIAL CASE OF THE TWO-DIMENSIONAL CUTTING STOCK PROBLEM*

HAROLD J. STEUDEL†

A heuristic algorithm employing dynamic programming is presented for solving the two-dimensional cutting stock problem where all the small rectangles are of the same dimensions, but without the usual restriction that the cutting be done with “guillotine” cuts, i.e., cut which must be made in stages from one edge to the opposite edge of the large rectangle being cut. The objective of the algorithm is to determine a cutting or layout pattern for which the ratio of the unused area to the total area of the large rectangle tends to be small. To demonstrate the method, the common problem of establishing standardized loading patterns for rectangular items on pallets is examined in detail. The algorithm is described with a minimum of mathematics through the use of several pictorial displays and a simple example. The efficiency of the heuristic is then evaluated by comparing computer generated loading patterns for 182 different size items to the loading patterns recommended by the U.S. Navy, and shown to be 10.4 percent more efficient for 64 out of 182 cases when the number of items per layer were not identical. The algorithm is also shown to be an effective aid to management both in establishing standardized loading patterns and procedures, and in communicating these loading standards to production personnel via computer generated “shop paper.” This type of computer design flexibility and control is a valuable management tool not only for standardizing pallet arrangements, but for carton design and consolidation, warehouse design and layout, bin and shelf stocking, designing tapes for numerically controlled gas cutting machines, and numerous other industrial problems involved with the efficient layout of rectangular objects.

(DYNAMIC PROGRAMMING—APPLICATIONS; PRODUCTION/SCHEDULING—CUTTING STOCK; PRODUCTION/SCHEDULING—MATERIAL HANDLING)

1. Introduction

A common problem for consumer goods industries is establishing standardized procedures for loading finished packaged goods onto pallets for subsequent storage and distribution. These standards are usually distributed as a specification or methods sheet consisting of a sketch showing how to position the product on the pallet, plus a product description and loading instructions. Generating these sheets manually is largely routine, but nevertheless draws on the experience of an analyst to determine loading patterns which yield good utilization of the pallet. The task is laborious and often time consuming. Thus, using a computer to automatically generate loading specification sheets for pallet loading has considerable practical value to many companies.

The pallet loading problem can be viewed as a special case of the two-dimensional cutting stock problem where all the small rectangles are of identical dimensions. The task consists of partitioning a rectangular pallet of length \(L\) and width \(W\) into smaller rectangular areas of length \(l\) and width \(w\) so as to determine a loading pattern which tends to minimize the amount of unused pallet deckboard area. The problem is constrained by maximum load height and weight limitations.

Considerable work has been done on both the one-dimensional and the two-dimensional cutting stock problem. A recent review of approaches to and computational experience with one-dimensional cutting problems is given by Golden [3]. The two-dimensional problem also has been studied by several authors [2], [4], [6].

*Accepted by David G. Dannenbring; received July 5, 1977. This paper has been with the author 7 months for 3 revisions.
†Marquette University.
recently, Christofides and Whitlock [1] have presented an effective tree-search algorithm for solving two-dimensional cutting problems in which a constraint is imposed to limit the number of each size of small rectangle to be cut. The algorithm is also constrained to consider only cutting patterns made by guillotine type cuts, that is, straight cuts made in stages from one edge to the opposite edge of the object being cut, such as with a common paper cutter. These constraints are common to solutions currently available.

The requirements for solving pallet loading problems, however, differ in two ways. For one, items loaded and transported on any one pallet are generally identical in dimensional size and weight. Secondly, practical experience suggests that considerable advantage in terms of the utilization of the deckboard area often can be gained by employing loading patterns which could not be obtained with guillotine type cuts.

In this paper a heuristic algorithm based on dynamic programming is presented for solving two-dimensional cutting stock problems in which all the small rectangles are of the same dimensions and nonguillotine cuts are allowed. The algorithm has been coded using FORTRAN IV to generate pallet deckboard loading patterns which satisfy commonly accepted criteria of load stability. The effectiveness of the algorithm is evaluated by comparing 182 computer generated loading patterns with the loading patterns recommended by the U.S. Navy Supply Research and Development Facility. The results indicate that the algorithm is both effective and computationally efficient.

2. Model Formulation

The problem is to determine a cutting or "loading" pattern for which the ratio of the unused area to the total area of the large rectangle tends to be small. To solve this two dimensional problem, dynamic programming is first used to determine four optimum sets of length and/or width placements of the small rectangles along the inside edges of the large rectangle. The objective is to maximize the utilization of the perimeter of the large rectangle. In the second phase of the algorithm the optimum arrangement of rectangles along the perimeter is projected inward to fill in the center portion of the large rectangle so as to minimize the amount of unused area. Each of the two phases of the resulting heuristic will now be discussed.

Description of the Recursive Procedure

To determine the placement of small rectangles which maximizes the utilization of the perimeter of the large rectangle, the following recursion is defined:

\[ F_n(S_n) = \max \left[ X_n \cdot l + Y_n \cdot w + F_{n-1}(S_{n-1}) \right] \] (1)

subject to:

\[ X_n \cdot l + Y_n \cdot w \leq D_n, \quad n = 1, \ldots, 4, \] (2)

where

- \( F_n(S_n) \) = the maximum value of the sum of the length and width placements through stage (edge) \( n \) of the large rectangle with state variable \( S_n \) entering that stage.
- \( X_n \) = the number of small rectangles of length \( l \) placed lengthwise along edge \( n \).
- \( Y_n \) = the number of small rectangles of width \( w \) placed widthwise along edge \( n \).
- \( D_n \) = the dimensional size of edge \( n \) for the large rectangle (either length \( L \) or width \( W \)).
- \( S_n \) = the state variable which defines the initial conditions for edge \( n \). \( S_n \) has three possible values:
  - \( S_n = 1 \): \( X_n = 0, Y_n = 2 \),
  - \( S_n = 2 \): \( X_n = 2, Y_n = 0 \),
  - \( S_n = 3 \): \( X_n = 1, Y_n = 1 \).
To establish the necessary geometry relationships, a pattern layout is defined to consist of a set of from one to four of the following “blocks” of small rectangles where:

- $B_1$: the block of rectangles formed by $X_1$ and $Y_4$,
- $B_2$: the block of rectangles formed by $X_2$ and $Y_1$,
- $B_3$: the block of rectangles formed by $X_3$ and $Y_2$,
- $B_4$: the block of rectangles formed by $X_4$ and $Y_3$.

Figure 1 is an example layout pattern which shows the arbitrarily selected designation of edge $n$ (i.e., stage $n$) $n = 1, \ldots, 4$. Also shown are the four blocks of rectangles which define the convention for the relative placement of the small rectangles along edge $n$. For example, if $X_1 > 1$ (and hence $Y_4 > 1$), then block $B_1$ exists and its relative placement will be as shown at the intersection of edge 1 and edge 4. Likewise, if at edge 2 (stage 2) the optimal value of $X_2 > 1$ for some value of the state variable $S_2$, then block $B_2$ exists which defines the positions of both the $X_2$ rectangles and the $Y_1$ rectangles. This convention thus prohibits intermixing length and width placements along an edge. Furthermore, all rectangular items are assumed to be stacked on the bottom or largest area surface.

**Description of the Inward Projection Procedure**

The second phase of the model projects the arrangement of rectangles along the perimeter inward to fill in the center portion of the large rectangle. Two potential problems must be considered. For one, a condition of overlap or interference between the blocks could occur. An example was shown in Figure 1 where the inward projection of the perimeter layout results in a condition of interference between blocks $B_1$ and $B_3$ as indicated by the cross hatched area. The second problem is that the inward projection could result in a layout pattern which has a central hole larger than a small rectangle.

The condition of interference is checked for by evaluating simple linear constraints. For example, interference between blocks $B_1$ and $B_3$ occurs only if both of the following conditions are true:

\[
(D_1 - X_1 \cdot l) < X_3 \cdot l \quad \text{and} \quad (D_4 - Y_4 \cdot w) < Y_2 \cdot w.
\]

Likewise, interference occurs between block $B_2$ and $B_4$ when,

\[
(D_1 - Y_1 \cdot w) < Y_3 \cdot w \quad \text{and} \quad (D_2 - X_2 \cdot l) < X_4 \cdot l.
\]

Interference conditions are relieved by treating blocks $B_1$ and $B_2$ as fixed in size. Blocks $B_3$ and $B_4$ are then modified by redefining values of $X_3$, $Y_2$, $X_4$ and $Y_3$ in order to satisfy the constraint which was initially in conflict. For the case shown in Figure 1, the interference condition would be relieved by reducing the value of $X_3$ by...
one. Accordingly, the value of \( Y_3 \) would be increased by one. The final layout pattern is shown in Figure 2. For this layout, the area of the resulting central hole is less than that of a small rectangle, and hence is insignificantly small. Furthermore, this layout pattern provides for 29 items per layer (95.2% deckboard utilization), whereas the best layout which could be obtained with guillotine type cuts would only yield 27 items per layer (88.6% deckboard utilization). The nested loading pattern also offers good stability.

In some cases, projection of the perimeter layout inward results in a central hole which is larger than a small rectangle. This type of situation is shown in Figure 3a. Although the resulting internal hole is quite large, it cannot be filled in efficiently by further projection. At this point, the model checks for multiple optimum solutions to the recursion, and returns to the beginning of Phase 2 if other solutions exist. If no other optimum solutions exist, blocks \( B_1 \) and \( B_2 \) are again held fixed, and blocks \( B_3 \) and \( B_4 \) are modified in size. In this case, \( B_4 \) is expanded and \( B_3 \) is reduced as shown by the resulting loading pattern in Figure 3b.

![Figure 3a. Layout Pattern with Large Central Hole.](image)

![Figure 3b. Final Layout Pattern after Hole Correction.](image)

3. An Example

Consider the problem of determining the pallet loading pattern where \( L = 48, W = 40, I = 15.5 \) and \( w = 9.5 \). At stage 1 (edge 1) the algorithm determines the following optimal values of \( X_1 \) and \( Y_1 \) for the three possible values of \( S_1 \) subject to the constraint

\[
15.5(X_1) + 9.5(Y_1) \leq 48.0
\]

**STAGE 1**

<table>
<thead>
<tr>
<th>State</th>
<th>( X_1^* )</th>
<th>( Y_1^* )</th>
<th>( F_1^*(S_1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>47.50</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>46.50</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>44.00</td>
</tr>
</tbody>
</table>

*Denotes optimum values

At Stage 2 (edge 2) with \( S_2 = 1 \), the optimum values of the decision variables are \( X_2 = 0, Y_2 = 4 \). The value of the objective function at this stage is 38.00 + 46.50 = 84.50. The additional 46.50 is the optimum value of the objective function at Stage 1 for the entering state variable \( S_1 = 2 \), which is a consequence of the decision of Stage 2. In a similar manner for \( S_2 = 2 \), the optimal decisions are \( X_2 = 2, Y_2 = 0 \) which yields the maximum value of \( F_2^*(2) = 78.50 \). These stage 2 decisions allow the state variable at stage 1 to assume values of either \( S_1 = 1 \) or \( S_1 = 3 \). The alternative \( S_1 = 1 \) is selected since this is the better choice. \( F_1^*(1) = 47.50 \) Similarly the maximum value
of the decision through stage 2 for entering state $S_2 = 3$ is $F_2^*(3) = 34.50 + 47.50 = 82.00$. The results are summarized below for stage 2.

<table>
<thead>
<tr>
<th>State</th>
<th>$X_2^*$</th>
<th>$Y_2^*$</th>
<th>$F_2^*(S_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>84.50</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>78.50</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>82.00</td>
</tr>
</tbody>
</table>

The solution for the problem through the third stage is obtained in a similar manner and shown below.

<table>
<thead>
<tr>
<th>State</th>
<th>$X_3^*$</th>
<th>$Y_3^*$</th>
<th>$F_3^*(S_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>126.00</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>131.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>128.50</td>
</tr>
</tbody>
</table>

The optimum decisions through the third stage yields $F_3^*(2) = 131.00$. This value was obtained under the state variable conditions $S_3 = 2$ and $S_1 = 2$ which in turn dictates the value of the state variable $S_4$. Thus the result can be obtained in one step.

<table>
<thead>
<tr>
<th>State</th>
<th>$X_4^*$</th>
<th>$Y_4^*$</th>
<th>$F_4^*(S_4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>169.00</td>
</tr>
</tbody>
</table>

The optimal solution can now be written for this four stage problem by tracing back from stage 4 to stage 1. The result at stage 4 determines that $S_3 = 2$ and $X_3 = 3$, $Y_3 = 0$. This leads to the second stage with $S_2 = 1$ the best choice for $S_2$ and yields $X_2 = 0$, $Y_2 = 4$. Finally at the first stage we get $X_1 = 3$, $Y_1 = 0$. In this case, blocks $B_1$ and $B_3$ are identical, no interference or central hold occurs, and the resulting layout pattern shown in Figure 4a results.

It is interesting to note the value of using the recursive relation expressed by equation (1). If the decisions were made independently at each stage, (i.e., without a recursion) suboptimum solutions result. At stage 1, in the above example, the best "independent" decisions would be obtained with $X_1 = 0$, $Y_1 = 5$, yielding a "edge 1" value of 47.50. If these decisions were fixed at this time, the best decisions possible at

![Figure 4a](image1.png) **FIGURE 4a.** Layout Pattern Obtained with Recursive.  
Where $L = 48$, $W = 40$, $l = 15.5$ and $w = 9.5$.  
![Figure 4b](image2.png) **FIGURE 4b.** Layout Pattern Obtained Without Recursive.
“edge 2” would be $X_2 = 1, Y_2 = 2$. Proceeding to make decisions independently would finally yield the layout pattern shown in Figure 4b. This pattern, however, is only 93.8% as efficient as that of Figure 4a.

4. Computational Results

To evaluate the performance of the algorithm, layout patterns for a 40 inch × 48 inch pallet were generated in 0.50 inch increments for items ranging in size from 5.00 inches to 14.00 inches in width, and from 7.00 inches to 15.00 inches in length. The

<table>
<thead>
<tr>
<th>Box Width (inches)</th>
<th>7.00</th>
<th>7.50</th>
<th>8.00</th>
<th>8.50</th>
<th>9.00</th>
<th>9.50</th>
<th>10.00</th>
<th>10.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00</td>
<td>52 (51)</td>
<td>48 (45)</td>
<td>48 (45)</td>
<td>45 (45)</td>
<td>40 (39)</td>
<td>40 (39)</td>
<td>36 (34)</td>
<td>36 (34)</td>
</tr>
<tr>
<td>5.50</td>
<td>50 (50)</td>
<td>44 (44)</td>
<td>44 (44)</td>
<td>38 (38)</td>
<td>38 (38)</td>
<td>34 (33)</td>
<td>34 (33)</td>
<td>32 (31)</td>
</tr>
<tr>
<td>6.00</td>
<td>46 (43)</td>
<td>40 (38)</td>
<td>40 (38)</td>
<td>37 (37)</td>
<td>34 (33)</td>
<td>33* (33)</td>
<td>30 (28)</td>
<td>30 (30)</td>
</tr>
<tr>
<td>6.50</td>
<td>37 (37)</td>
<td>37 (37)</td>
<td>32 (32)</td>
<td>31 (27)</td>
<td>30 (27)</td>
<td>28 (27)</td>
<td>26 (26)</td>
<td></td>
</tr>
<tr>
<td>7.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29 (26)</td>
<td>27 (26)</td>
<td>26 (26)</td>
<td>26 (26)</td>
</tr>
<tr>
<td>7.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26 (26)</td>
<td>26* (26)</td>
<td>26 (26)</td>
<td>25 (25)</td>
</tr>
<tr>
<td>8.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>26 (26)</td>
<td>26 (26)</td>
<td>24 (22)</td>
<td>22 (20)</td>
</tr>
<tr>
<td>8.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 (20)</td>
<td>20* (20)</td>
<td>20 (20)</td>
<td>19 (19)</td>
</tr>
<tr>
<td>9.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20 (20)</td>
<td>20 (20)</td>
<td>18 (18)</td>
<td></td>
</tr>
<tr>
<td>9.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18* (18)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In forming a UNIT LOAD, the over-all lateral dimensions shall not exceed
52 in. in length nor 43 in. in width.
resulting patterns were then compared in terms of the number of items per layer to the pallet patterns recommended by the U.S. Navy Supply Research and Development Facility as reported by Haynes [5]. These patterns will be referred to hence as the “standard.” Loading specifications and requirements relating to load stability were identical for the comparison. The results of the comparison for 182 different size items are shown in Table 1. For each item size, the number of items per layer obtained via the algorithm is given along with the number of items per layer (in parenthesis) specified using the “standard” pattern. Those sizes for which identical loading patterns were obtained from each method are further indicated by an asterisk (*). In some cases, a larger item is shown to have more pieces per layer than an adjacent smaller item. This apparent discrepancy results when a different pallet load pattern is recommended by the Navy specification sheet, and a larger amount of the potential overhang is utilized. Accordingly, the algorithm is run with the same amount of overhang which was more overhang than was assumed for loading the smaller item. Consequently, a larger number of items per layer resulted. (The amount of overhang stated in Table 1 was from the Navy specification sheet, and was used only as required to maintain consistency. In general, the maximum overhang should not exceed 50% of the smallest dimension of the item being stacked up to some upper limit.)

These results show that for identical restrictions, the proposed algorithm always generates a loading pattern which is at least as good as the “standard.” For items under 8.00 inches width, the algorithm often yields layout patterns with better utilization of the deckboard. With the larger items, the results are generally equivalent. This is due largely to the limited number of items, and accordingly loading pattern combinations, which are possible on the relatively small 40 inch × 48 inch pallet deckboard. For the results given in Table 1, however, the proposed algorithm exceeds the “standard” by an average improvement of 10.4 percent in deckboard utilization for the 64 out of 182 cases where the number of items per layer were not identical.

---

**P A L L E T L O A D I N G S P E C I F I C A T I O N S**

---

**TOP VIEW**

---

```
LOAD PRODUCT NUMBER 45678 AS SHOWN ABOVE WITH:
  29 BOXES PER LAYER
  7 LAYERS PER PALLET LOAD
  203 TOTAL BOXES PER PALLET LOAD
  1627.  LBS. TOTAL WEIGHT PER PALLET LOAD

BOX SIZE—LENGTH 8.50 IN., WIDTH 7.50 IN., HEIGHT 7.40 IN.
94.3 PERCENT DECKBOARD UTILIZATION
```

**FIGURE 5. Computer Generated Layout Pattern.**
Figure 5 is an example of computer generated shop paper which serves to communicate the loading standards to production personnel. In addition to the layout pattern, this sheet provides part identification, inventory status, and loading instructions. The level of control this type of standardization provides is especially valuable for multiplant companies.

The computer code for the algorithm is quite efficient. Various runs made on a Xerox Sigma 9 computer using the FLAG compiler showed that approximately 1.25 CPU seconds are required to solve any of the problems tested. The total memory requirements of the code is 14K words.

5. Concluding Remarks

The heuristic presented provides good solutions to two-dimensional cutting stock problems in which cuts other than guillotine type cuts are allowed. Potential applications are numerous since many industrial operations involve decisions regarding the method or pattern to use in partitioning a large rectangle into smaller rectangles of equal dimensions. In the case of establishing standardized pallet loading patterns, the computerized algorithm provides a practical way to generate shop paper for specifying the method for loading rectangular goods on pallets. The program also provides management with a viable means to evaluate and select the best size of pallet to accommodate the range of product sizes which are handled and stored.

The next step in this research would be to extend the heuristic to consider the case where all the small rectangles are not the same size. A recursion could be easily defined in terms of \( I_i, W_i, X_m, \) and \( Y_n \) for a known (and relatively small) number \( K \) of different small rectangle \( i = 1, \ldots, K \). Likewise state variable values could be defined for pairs of different size rectangles. The challenge in this extension would be in defining the geometry relationships for determining the relative placement of the items along an edge so as to minimize the frequency of internal holes in the layout pattern. This task does not seem impossible, however, and the resulting heuristic would have considerable application, such as in the area of plate cutting using gas torches where nonguillotine cuts are both applicable and advantageous.\(^1\)

\(^1\) The author wishes to thank the referees for their many useful comments and suggestions. Appreciation is also expressed to Dr. T. Heintz for his comments in the earlier stages of this work, and to Dr. T. R. Martin, former Dean of Business at Marquette University, for his support of this research.

References