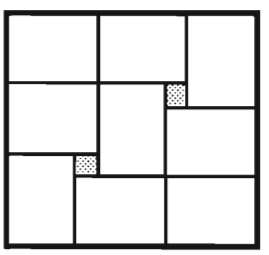
PACKING PROBLEMS ON A PC

by

ANDREW GEORGE DEIGHTON



Submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in the
Department of Computer Science,
University of Natal
1991

Durban
1991

ABSTRACT

Bin packing is a problem with many applications in various industries. This thesis addresses a specific instance of the this problem, known as the Container Packing problem. Special attention is paid to the Pallet Loading problem which is a restricted sub-problem of the general Container Packing problem.

Since the Bin Packing problem is NP-complete, it is customary to apply a heuristic measure in order to approximate solutions in a reasonable amount of computation time rather than to attempt to produce optimal results by applying some exact algorithm. Several heuristics are examined for the problems under consideration, and the results produced by each are shown and compared where relevant.

PREFACE

The work described in this thesis was carried out in the Department of Computer Science, University of Natal, Durban, from January 1989 to December 1991, under the supervision of Mrs. Jane J. Meyerowitz.

The studies represent original work by the author and have not been submitted in any form to another University. Where use was made of the work of others, it has been duly acknowledged in the text.

ACKNOWLEDGEMENTS

It is fitting that thanks be extended to everyone who has helped me in completing this thesis.

In particular, I would like to extend sincere thanks to Jane Meyerowitz, my supervisor, for the encouragement and advice that she has given me throughout.

I would also like to thank my fellow students and the staff of the Computer Science Department for their help and suggestions.

Special thanks are necessary to the people who proof read the thesis and helped to produce the final form, especially Huw Williams, Michael Deighton, Stuart Melville, Jane Meyerowitz, Dave Carson, Rosalind Kyle and Peter Deighton.

For her constant support and encouragement, I must also thank my fianceè, Rosalind Kyle.

Thanks are also extended to Ortrud Oellermann for the use of her computer on which the final draft has been prepared.

To everyone and anyone who helped in any way, thank you!

NOTATION

The following notation conventions are used throughout this thesis.

$|X|$: Absolute value, the difference between X and 0.

$\{X\}$: Ceiling function, the smallest integer greater than or equal to X.

$[X]$: Floor function, the largest integer less than or equal to X.

$|S|$: Size, the size (number of elements) of the set S.

CONTENTS

CHAPTER 1

INTRODUCTION

Almost everyone has, at some stage, come into contact with some form of packing problem. Whether it is packing food into a lunchbox, clothes into a suitcase, or even arranging furniture in a room, an optimal placement is necessary in order to make full use of the available space.

The problem of determining this optimal placement is known as the 'Packing Problem', and in its most general form, can be stated as:

> Given a fixed amount of some resource, and a number of weighted consumers of that resource, find an arrangement which maximises the total weight of the satisfied consumers
> .

Consider the example of arranging furniture in a room: the resource is the total available space within the room, defined by the room's dimensions, and the consumers of that resource are the individual pieces of furniture. The furniture is considered a consumer of the available space from the viewpoint that only one piece of furniture can occupy any space at any given time, thereby consuming it. The weights could depend on how dearly the furniture is prized. The weight of the family heirloom Chesterfield, for example, will probably be more than that of the chipboard magazine stand.

In this example, there are also a number of other constraints that must be added to the general problem: there must be enough space for people to be able to use the furniture; all the pieces must stand on the floor, not on top of each other; no furniture may be placed on its side; the Chesterfield must not be in direct sunlight; the chairs should not be in the fireplace; etc...

So, for any particular problem, the general statement may be modified to:

> Given a fixed amount of some resource, and a number of weighted consumers of that resource, find an arrangement whereby the total weight of the satisfied consumers is maximised, subject to any relative, problem dependant constraints.

This problem has a large number of industrial applications. It is found in various forms in a wide variety of different contexts and in various different dimensions; each instance of the problem substituting its own properties for the resource, consumers, weights and constraints.

Brown [Bro71] discusses a number of applications of the general packing problem. Industrial situations to which the problem is applied include such diverse fields as scheduling advertisements for television and radio, reducing the amount of wasted material (glass, paper, wood) in the form of unusable offcuts, allocating computer resources to a number of competing processes, and determining packing layouts for transporting goods to customers.

All these problems fit the general description of the packing problem. In the advertisement scheduling case, for example, the resource corresponds to the amount of time available for the advertisements to be broadcast; the consumers are the various advertisements; and their weights depend on the price paid for each. The additional restrictions include not showing the same advertisement twice in the same break, not showing two advertisements for the same type of product in the same break, and any other restrictions that the individual station requires. [Bro71]

Some applications of the packing problem have received more attention in the literature than others. The two most common are the 'Cutting Stock' and 'Trim Loss' problems. Both problems can be described in their 2-dimensional form in terms of the glass industry, although they are just as applicable to other industries and in other dimensions.

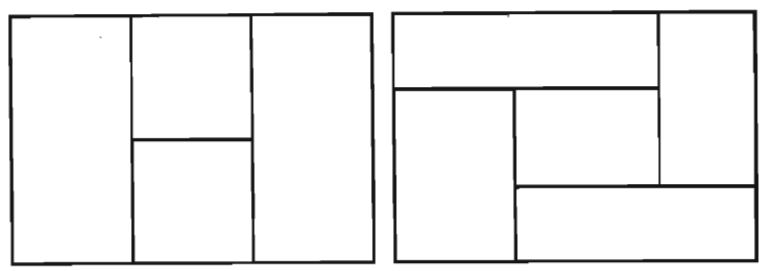
The cutting stock problem involves determining how to cut a given set of orders from a stock sheet. The dimensions of the required pieces and the stock sheet are known, and a cutting plan is required that will allow the maximum amount of the required order to be cut.

The trim loss problem, on the other hand, requires that the amount of unusable trim be minimised. Here it is usual that the orders be satisfied as soon as they are received, and an algorithm is needed that will determine which of the available stock sheets to use in order to reduce the amount of unusable offcuts remaining.

These problems have been the subject of a number of works. Roodman [Roo86] investigated a 1-dimensional problem of minimising waste for a steel manufacturing company, with the added constraint that the waste be kept to the minimum number of pieces. A $1\frac{1}{2}$-dimensional problem of this type in which the stock sheet has a finite fixed width but infinite length has also been addressed. The object is to cut all the required order items while keeping the length of the stock sheet used to a minimum. [Bak81, Cha87]

The 2-dimensional case has been investigated in considerable depth. Various authors have included their own restrictions: some require that the cut rectangles have their edges parallel to the edges of the stock sheet (orthogonal layouts) [Bak80] while others view this restriction as too limiting [Rin87]. Layouts are sometimes restricted to those which can be formed by applying only guillotine cuts [Bea85], and other times

not [Bea85a]. Guillotine cuts are cuts that extend from one side of the stock sheet to the opposite side, parallel to the other pair of sides. Figure 1.1 (a) shows an example of a layout formed by guillotine cuts, while figure 1.1 (b) shows a layout that cannot be formed by guillotine cuts.



(a) (b)

Figure 1.1: A layout that can be formed by a sequence of guillotine cuts (a), and one that cannot (b).

The definitive work in the field is generally considered to be that of Gilmore and Gomroy [Gil64], while recent approaches have attempted to use knowledge based systems [Dag90].

If one considers the available resource as the allowable volume into which a given cargo load can be placed, and the consumers of that resource to be the individual items making up that cargo, the general packing problem reverts to that of packing the maximum amount of the given cargo into the available space. This problem, which forms the focus of the current work, is particularly important for any industry that requires sending bulk products to customers.

> The efficient utilisation of transport and warehousing facilities is of prime importance in all industries and considerable effort is often employed in obtaining improvements. [Dow84a]

The problem in this form becomes

> Given the dimensions of some containing region, and a description of a cargo to be packed into that region, calculate a layout for the cargo within the region that results in maximum space utilisation.

This problem is known as the Container Packing Problem, as it is commonly applied to the case where the available space is in the form of a shipping container, and the cargo is a list of cartons that are to be shipped or stored within that container.

The container packing problem itself is still a general problem, encapsulating a large number of sub-problems. Each instance of the container packing problem has its own individual characteristics, depending on the nature of the cargo and the reason for the packing, making it unique and requiring a specific method to provide an efficient solution.

The main problem with all the packing problems discussed so far is that they belong to the class of problems known as NP-complete, [Cha87, Yao80, Dow84a, Dag90] and thus it is not expected that an exact algorithm will be developed to solve the problem in a feasible amount of computation time.

> Algorithms for pallet loading cannot, in a reasonable amount of computer time, generate optimality [Dow84a]

> The problem is NP-Complete. Mathematical programming techniques are generally inadequate for the solution of these problems due to the computational burden. Thus, instead of

> trying to reach an optimal layout pattern, most researchers have used heuristic solutions. [Dag90]

The heuristics that have been developed to solve a number of container packing problems are presented in chapters two and three of this work. Before examining them, however, it is necessary to identify the one sub-problem of the container packing problem that has received most attention. This problem, known as the 'Pallet Loading Problem' or 'Manufacturers Loading Problem' is of particular importance.

The pallet loading problem, which forms the focus of chapter two, restricts the possible cargos and the format of the layouts produced.

The cargo being packed is restricted to a set of rectangular cartons (boxes), all of which are necessarily of identical dimensions. These are packed into the available space in strict layers. As a result of these restrictions, the problem reduces to one of determining the maximum number of rectangles that can be packed into a larger containing rectangle [Dow87].

This reduced problem is still not as simple to solve as it may seem, as is evident from the results shown in chapter two.

It is self-evident that, for a number of reasons, it is advantageous to be able to solve the packing problem. A number of authors have quoted the obvious advantage of reduced transportation costs to be gained from shipping a maximum number of boxes in a single container [Ste84, Smi80]. Brown [Bro71] has also given the example of calculating an optimal number of distinct box sizes to use for shipping a given range of variable sized objects. Dowsland [Dow84a] has also shown how the entire packaging process can be integrated into a single computer package in order to minimize costs throughout the process, by

calculating the best box size to optimally fit onto a pallet, into a container, etc.

All these applications require that a rapid solution be available for the Pallet Loading problem, and thus this is a fair problem to tackle with the technology of modern computing power. But why restrict the solution process to personal computers (PC's)?

There are a number of reasons for this, including cost and availability. Personal computers are much cheaper than mainframes. This means that the solutions to the problem will be affordable to many more people and companies if they can be calculated on personal computers. Many organisations have access to personal computers, so if a package to solve this problem can be created for a PC, users will not have to purchase expensive new hardware.

There are a number of other advantages to using a micro-computer that have appeared in the literature. These include aspects such as interaction between the packing staff and the computer, which is not easily available on mainframe systems, and the improved graphic capabilities of the micro which allows the production of easy to understand picture-type output that can easily be interpreted by the packing staff who are generally not computer literate.

> The program package was initially developed for use on the company's IBM 360 computer system. ... was necessarily run in batch mode, and this proved to be a major disadvantage owing to the remoteness of the intended user in both a physical and organizational sense. In order to allow a higher degree of interaction between user and machine, the package was modified for operation on a Z80 micro-computer. [Bis82]

> The ability for non-computer staff to make direct use of interactive computer facilities therefore aids problem solution and also ... helps overcome the apparent threat of computer technology. [Dow84]

> The problem can only be effectively tackled if interaction is possible between the computer and those concerned with packaging and distribution. This can be achieved effectively by using modern interactive computer technology. [Dow84]

Armed with these reasons and justifications, research into various packing problems and their solutions was undertaken. The results of the research are presented in this thesis. Chapter two presents methods of solving the restricted pallet loading problem while chapter three addresses some of the more general container packing problems, and their solutions.

CHAPTER 2

THE PALLET LOADING PROBLEM

The pallet loading problem has received more attention than any other container packing problem in the open literature for a number of reasons: the restrictions it places on the cargo slightly reduce the complexity of the problem; the solutions can be applied without increasing labour costs; and it is a problem that is often encountered in transport situations.

It includes several restrictions on the cargo and the layouts produced, and thus is slightly easier to work with than any general packing problem. The main restriction it applies is that the boxes are all identical in dimension. This allows the other restrictions to be imposed, as they use this uniformity to reduce the complexity of the layouts produced.

It will be assumed for discussion purposes that the boxes have dimensions a×b×c, and the pallet on which they are to be packed has base dimensions X×Y, and a maximum allowable height for the boxes Z. Unless otherwise stated, all dimensions are given in millimeters, or are irrelevant.

The results produced are restricted to orthogonal, $2\frac{1}{2}$-dimensional layouts. That is, the edges of each of the boxes are kept parallel to the edges of the pallet, and the boxes are packed in strict layers, with the top of one layer forming a flat surface on which the next layer can be packed.

The results can also be interpreted easily as a result of the graphical output available, and so can be applied more quickly, resulting in reduced transportation costs without increased labour costs associated with the physical packing process.

The problem can be stated formally as

> Given the dimensions of a containing space (pallet), and the dimensions of a cargo unit (box), find the orthogonal, $2\frac{1}{2}$-dimensional layout that will result in the maximum number of such units being packed into the available space.

DeCani has shown that an orthogonal arrangement of boxes will not necessarily produce an optimal layout for all box and pallet combinations [Dec78]. This is done by considering packing a box of size 21×1 onto a 20×10 pallet. Since the length of the box is more than either of the dimensions of the pallet, no boxes can be packed by applying an orthogonal packing. However, a box can be packed if it is placed diagonally across the pallet as in figure 2.1.

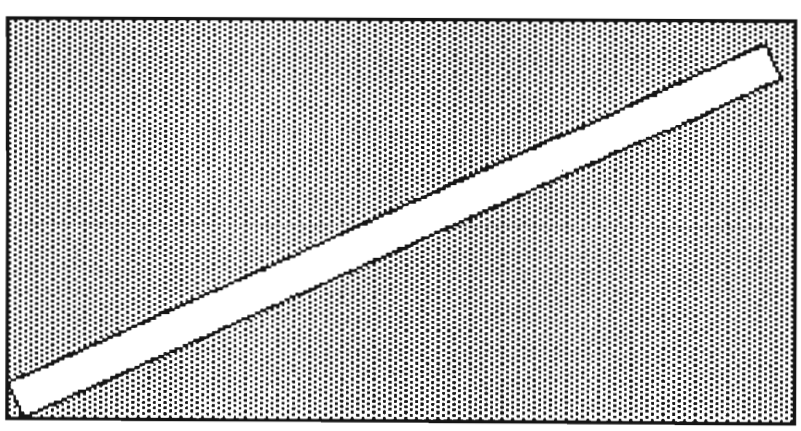


Figure 2.1: DeCani's case where orthogonal layouts are not optimal.

Despite this, for the pallet loading problem, a restriction is made to orthogonal layouts for a number of reasons.

> Non-orthogonal layouts need not be considered as the crushing force on the corners of the boxes would be intolerably high. [Smi80]

> Practical considerations usually mean that the boxes must be placed orthogonally and in layers in which the vertical orientation of the boxes is fixed [Dow87]

The layouts that are produced consist of a set of flat layers which are duplicated up the height of the pallet without exceeding the maximum height restrictions.

> For ease of packing on the shop-floor, all layouts must be in strict layers. [Smi80]

It is often the case that boxes must be kept upright when packed, especially if the products packed in the boxes can be damaged by being placed in some other orientation. If this is not the case for the particular product being packed, then the overall height utilisation can often be improved by defining three layers, corresponding to the three possible vertical orientations of the boxes being packed and using an optimal combination of these three layers.

Suppose that such a set of three layers has been calculated. Each layer in the set has a number of boxes in it (N_i) and a vertical dimension (D_i). The optimal combination of these layers contains T boxes, where T is given by using M_i of each layer, the M_i being calculated by maximising:

$$T = N_1M_1 + N_2M_2 + N_3M_3$$

subject to

$$D_1M_1 + D_2M_2 + D_3M_3 \leq Max\_Height,$$

$M_1, M_2, M_3 \geq 0$ and integers.

This maximisation can easily and exactly be carried out through the use of a complete search such as that given in algorithm 2.1.

Algorithm 2.1

To calculate the best combination of layers.

```
proc Best_Combination (Max_Height, D1, D2, D3, N1, N2, N3)
    Max = 0
    for M1 = 0 .. Max_Height div D1 do
        for M2 = 0 .. Max_Height div D2 do
            M3 = (Max_Height - (M1*D1 + M2*D2)) div D3
            if M1*N1 + M2*N2 + M3*N3 > Max then
                Max =  M1*N1 + M2*N2 + M3*N3
                Record( M1, M2, M3 )
            endif
        enddo
    enddo
    Return (Max)
endproc
```

Since algorithm 2.1 finds the optimal combination of layers exactly, and can be applied to any set of three layers, the packing algorithms need only determine the layers to use and can thus be evaluated and compared on a basis of number of boxes per layer [Bis82].

In order to compare the available algorithms, several packing situations have been evaluated using each. Deighton [Dei91] reported results for testing 207711 different combinations of box dimensions over a pallet surface of 5885×2321mm (corresponding to the dimensions of a 'General Purpose Container' as defined in [Dei88]), and these results are quoted where relevant. Several other test cases have been quoted in the literature, and these have also been used for testing each of the algorithms discussed. In particular, the exhaustive set of box and pallet dimensions described by Dowsland [Dow87] has been used to evaluate each of the algorithms.

For demonstration purposes, a sample set of seven box and pallet combinations has been chosen. The results are summarised in table 2.1, and the resulting layouts are described in the relevant sections. The results are also duplicated in Appendix A for ease of reference.

Test	X	Y	a	b
a	38	38	7	3
b	20	20	7	2
c	20	15	7	4
d	20	15	7	3
e	14	11	4	3
f	14	13	4	3
g	22	16	5	3

Table 2.1 - Test box and pallet sizes.

The problem remains to determine the number of boxes that can be packed into a layer on the pallet and which layout to use in order to pack the boxes in the best way. Several methods have been devised to solve this problem and these form the focus of this chapter.

2.1 PALLETISING CHARTS

Since a rapid solution to the problem is desirable for a number of reasons, researchers have been attempting for a number of years to find a means of providing a viable solution in acceptable time. Early attempts at this involved the production of what are known as "Pallet Loading Charts". For any given pallet, a chart can be drawn up to show how to pack boxes of any size onto the pallet.

> Practical applications usually require quick reference to suitable layouts, for a range of boxes on a given pallet. A pallet chart is often used to provide this. [Dow84]

The chart assumes a constant pallet size, and depicts layouts for all boxes with sizes in a given range on that pallet. They thus define a look-up function which can be used to determine the packing layout to use. There is one such chart for each pallet size, and if a new pallet size is used, a new chart will need to be produced.

> These aids (Palletising Charts) can be of considerable benefit, but their usefulness is limited because each chart is only applicable for a given pallet size. [Dow84a]

An example of a palletising chart is shown in figure 2.2. This figure shows the chart for packing boxes of base dimensions between 240×140mm and 410×270mm onto a 1060×813mm pallet.

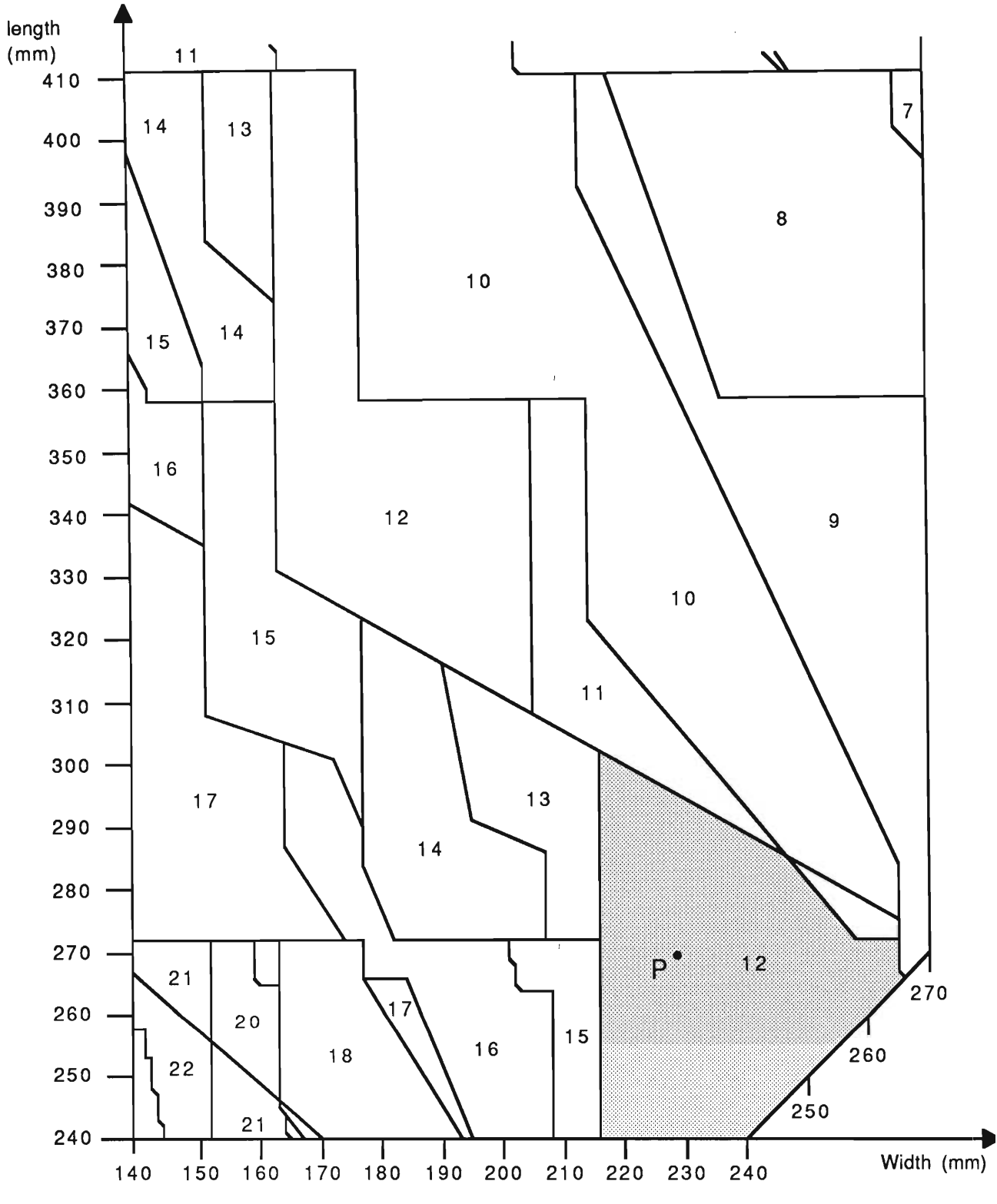


Figure 2.2: An example of part of a Pallet Loading Chart for a 1060×813 mm pallet (from Smi80)

The vertical axis of the chart represents the box length, while the horizontal axis corresponds to the box width. For packing a box of dimensions a×b within the range of the chart onto a pallet of the correct size, the maximum number of such boxes that can be packed onto the pallet is given in the region of the chart in which the point (a,b) lies. For example, a box of size 270×230mm can be checked with reference to the point P in figure 2.2. This box size, and any other box size that falls into the shaded region has a maximum packing number of 12. That is, 12 such boxes can be packed per layer on a pallet of the specified dimensions.

These charts are usually accompanied by a set of tables (diagrams) showing how to carry out the packing in order to pack the maximum number of boxes. For the same example as above, the required packing plan would be as in figure 2.3.

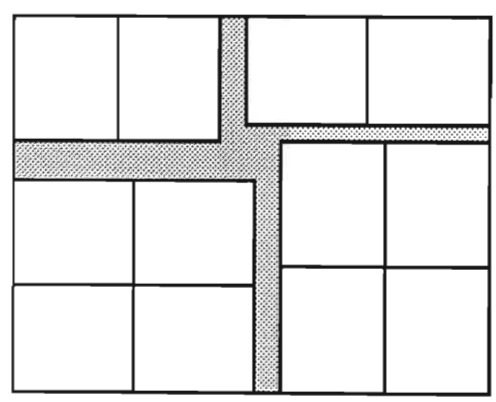


Figure 2.3: The packing configuration for 270×230mm boxes on a 1060×813mm pallet.

Despite the obvious advantages of such charts, they did not enjoy much coverage in the open literature. The reasons for this were the high cost of producing such charts, as well as their inability to handle varying pallet sizes.

The charts are expensive to set up, as the packing process must be carried out for each box size within the required range.

> Very little information is available on how these charts are set up. One method is to cover the plane with an even lattice of points spaced at a given interval and to generate a suitable layout or layouts at each point. The difficulty with this method is in choosing the interval to give a dense cover of the required range and yet require a feasible number of calculations. [Dow84]

> Pallet charts ... are usually derived using a systematic search procedure over the range of packed sizes required, applying some algorithm. [Dow84a]

These charts as described are useful if the pallet is of fixed size and one wishes to evaluate the effect of packing different sized boxes onto the pallet. Sometimes, however, the box dimensions are fixed, and the analysis required is dependent on allowing different amounts of overhang over the edges of the pallet. This requires a number of pallet charts to be generated, corresponding to the different dimensions of the total containing rectangle. Another approach that could be used, although no reference could be found for it in the literature, would be to generate a pallet loading chart for a fixed box size and a varying sized pallet.

Dowsland [Dow84], however, went one step further in this area. By observing that packing a box ($a \times b$) onto a pallet ($X \times Y$) is the same problem as packing a different box ($ra \times rb$) onto the pallet ($rX \times rY$) for any real number $r > 0$, it can be shown that any pallet loading problem of this form can be represented as a point in 3-dimensional space with the axes defined by

$$x = \frac{a}{X},$$

$$y = \frac{b}{X},$$

$$z = \frac{Y}{X}.$$

By applying this observation, Dowsland defined a method of setting up a three-dimensional pallet loading chart which would allow analysis of the effects of changing pallet or box size either individually or in combination.

The chart is made simpler by the observation that

> Because X has been pre-defined as the pallet length and a as the box length, only the region defined by
>
> $0 < z \leq 1$
>
> $0 < y \leq x$
>
> needs to be considered. [Dow84]

This chart, however useful it may be, requires a computer to access it, as it is not possible to depict it graphically. It did, however, form the basis of a method of producing rapid results [Dow85a].

More work on the production of pallet loading charts was not carried out, because of the shift in emphasis of the research since the introduction of cheap, easily available computer equipment.

> The advent of the micro computer and the subsequent improvements in its graphics capabilities, has meant that the emphasis has shifted from palletizing charts and mainframe bureau results to microcomputer packages, complete with a graphical representation of the most suitable layout. [Dow87a]

Thus the research has shifted from algorithms to determine these charts to algorithms that will run in feasible time limits and produce acceptable results. The algorithms that have resulted from this research are presented in the rest of this chapter.

2.2 AN UPPER BOUND

Since the problem of packing boxes onto a pallet is known to be NP-complete [Dow87], it is unlikely that an efficient algorithm to solve the problem exactly will be forthcoming. It is therefore necessary to define some way of approximating the optimal solution as closely as possible in a feasible amount of time.

Thus, in attempting to solve the pallet loading problem, it is customary to apply some heuristic measure.

The closeness to which the heuristic's result resembles the exact result can be evaluated by comparing the number of boxes each method packs. It is usually easy to determine the number of boxes packed by the heuristic methods, however it is not so easy to find the exact solution with which to compare these results, since if it were, there would be no need for the heuristic in the first place.

It is, therefore, useful to have an estimate of the maximum number of boxes that can be packed with which to compare the results obtained, as this will provide an estimate of the accuracy of the heuristic being applied. This estimate is provided in the form of an upper bound.

A simple rule to follow is that the total surface area of the boxes packed cannot be more than that of the pallet. Thus, if N boxes are packed, we have that

$$N * \text{Box Area} \leq \text{Pallet Area},$$

or

$$N \leq \frac{\text{Pallet Area}}{\text{Box Area}}.$$

This clearly gives an upper bound on the number of boxes that can be packed, but is not always obtainable.

Consider, for example, the case of packing boxes of size 2×2 onto a 5×5 pallet. The upper bound is then $N \leq \left[\frac{5*5}{2*2}\right] = \left[\frac{25}{4}\right] = 6$. Thus the upper bound on the number of boxes packed is 6. However, it is not possible to pack more than 4 boxes without overlapping the edge of the pallet. (figure 2.4)

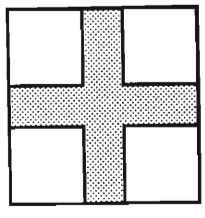


Figure 2.4: The best packing of a 2×2 box on a 5×5 pallet. The shaded regions denote unusable space.

An improvement to the direct area/area upper bound was suggested by Dowsland [Dow85]. This upper bound stems from the observation that if boxes are placed orthogonally on the pallet, then there is an equivalent packing layout in which the boxes have all been shifted as far as possible to the top-left corner of the pallet. (figure 2.5)

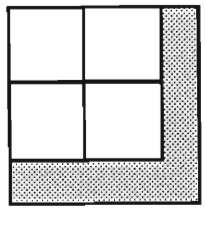


Figure 2.5: A packing layout equivalent to figure 2.4, in which the boxes have been shifted to the top-left of the pallet.

The position of each box in the layout can thus be expressed as an integer combination of box lengths and widths from the top left corner of the pallet. It then becomes clear that no box needs to extend to the right of the point defined by the maximum of such combinations. A similar argument can be applied to the other pallet dimension.

These two maximum lengths define a sub-pallet, known as a *perfect partition reduction* of the original pallet, which can be used to give an improved estimate of the maximum number of boxes that can be packed. By applying an algorithm such as the one shown in algorithm 2.2 , the perfect partition reduction of a pallet can be calculated.

The upper bound can thus be defined as

$$N \leq \frac{\text{Reduced Pallet Area}}{\text{Box Area}},$$

where the reduced pallet area is given as Reduced Length * Reduced Width.

```
Algorithm 2.2
    To evaluate a perfect partition reduction of a pallet.

proc Reduce_Length (MaxLength, BoxLength, BoxWidth)
    MaxSoFar = 0
    for X = 0 to MaxLength div BoxLength do
        Y = (MaxLength - X * BoxLength) div BoxWidth
        Total = (X * BoxLength) + (Y * BoxWidth)
        if Total > MaxSoFar then
            MaxSoFar = Total
        endif
    enddo
    Return( MaxSoFar )
endproc

proc Perfect_Partition (X, Y, a, b)
    Return((Reduce_Length(X, a, b)*Reduce_Length(Y, a, b)) mod (a*b)
endproc
```

This method can be applied to the example of a 2×2 box being laid on a 5×5 pallet. The reduction of the pallet to a perfect partition of the boxes gives a maximum of $\left\lceil \frac{4*4}{2*2} \right\rceil$ = 4 boxes per layer. This is the achievable maximum on the number of boxes that can be packed. (See figure 2.5)

The upper bound produced by the perfect partition reduction method, however, cannot always be attained. This can be shown by considering the following examples.

For packing a 5885×2321mm container floor with boxes of size 230×155mm the perfect partition reduction of the container floor is 5885×2320mm, given by (2*230mm + 35*155mm) × (2*230mm + 12*155mm), and thus the upper bound is 382 boxes per layer.

However, for a slightly smaller box (230×154mm) the reduced container surface is 5852×2310mm, which is given by (38*154mm) × (15*154mm), and so the upper bound produced is only 381.

If a layout of the larger box with 382 boxes per layer existed, then by replacing each box in that layout with a smaller box, one would obtain a layout of these smaller boxes containing more than the upper bound allows, which is clearly not possible. Thus there can be no such layer of the larger boxes, and hence the upper bound for the larger boxes cannot be reached.

A further improvement on this upper bound estimate was produced by Dowsland [Dow84] at about the same time as the perfect partition upper bound. This method depends on what are termed *equivalence classes* of problems.

It is intuitively obvious that packing a box of size 2×2 onto a 5×5 pallet is the same problem as packing a 20×20 box onto a 50×50 pallet with only a change of scale. It is perhaps less obvious that this is also the same as packing a 248×18 box onto a 53×48 pallet.

The possible layouts for a given box on a particular pallet depend not only on the absolute dimensions of the box and pallet, but also on their relative dimensions which determine the set of *efficient partitions* of the pallet on the box [Dow84].

A feasible partition of the pallet length (X) on the box length and width (a, b) is given as an ordered pair of non-negative integers (m, n) such that

$$m*a + n*b \leq X.$$

The partition is perfect if equality holds, and is efficient if it cannot be improved by increasing either m or n.

The set of efficient partitions of a pallet length of 5 units over a box length and width of 2 units each is

$$E\{5,2,2\} = \{(0,2); (1,1); (2,0)\}.$$

This is the same as the sets of efficient partitions that can be produced by box dimensions (a,b) = (24, 18) and pallet dimensions between 48 and 53.

Dowsland showed that the upper bound on the number of boxes that can be packed depends on the set of efficient partitions of the pallet on the box (E{X,a,b} and E{Y,a,b}). It can be proved that the upper bound on the number of a×b boxes that can be packed onto a X×Y pallet is equal to the upper bound for all other problems in the same equivalence class. In particular, it is equal to the upper bound for packing a box of dimensions c×d on a V×W pallet where c, d, V and W are chosen to minimise the value of $\left\lceil \frac{VW}{cd} \right\rceil$ subject to

$$nc + md \leq V;$$

$$nc + (m+1)d > V, \quad \text{for all } (n,m) \text{ in } E\{X,a,b\};$$

$$\left(\left\lceil \frac{X}{a} \right\rceil + 1\right)c > V;$$

$$pc + qd \leq W;$$

$pc + (q+1)d > W, \quad$ for all (p,q) in E{Y,a,b};

$$\left(\left[\frac{Y}{a}\right]+1\right)c > W.$$

This is a more computationally expensive problem to solve than any of the earlier methods, but produces an upper bound that has been shown to be attainable in more than 78% of examples in a given range [Dow84].

Another upper bound estimate which can occasionally be applied is the one suggested by Barnes [Bar79]. This provides a method of calculating the number of a×b rectangles that can be fitted into a large X×Y rectangle.

The method defines the maximum number of 1×a rectangles that can be packed by the following:

Let

$$X, Y \geq a$$

and define m,n such that

$n = X \bmod a \qquad (0 \leq n \leq a)$

$m = Y \bmod a \qquad (0 \leq m \leq a).$

Then the amount of wasted space on the pallet is given as:

$$A = \begin{cases} nm \text{ if } n+m \leq a \\ (a-n)(a-m) \text{ else.} \end{cases}$$

Similarly, for a 1×b rectangle, the wasted space (B) can be calculated.

For a packed rectangle a×b, the amount of wasted space can now be calculated as R where R=R(X,Y) is the least integer satisfying:

$$R \geq \max(A,B);$$

$$A = R \bmod a;$$

$$B = R \bmod b.$$

The upper bound is then given as:

$$\text{Bound} = \left\lceil \frac{X*Y-R}{a*b} \right\rceil$$

This result holds for relatively prime a,b and sufficiently large X,Y. The problem with this method is in the restriction to *sufficiently large* X and Y.

As Dowsland is quoted in reference to Barnes's method,

> "sufficiently large" implies that the ratio of containing rectangle to contained rectangle is larger than that encountered in most pallet loading problems. [Dow87]

The final upper bound algorithm which has been applied to the tested cases for evaluating the packing heuristics is that suggested by Dowsland [Dow87]. This method uses a combination of Dowsland's earlier algorithm [Dow84] with the method of Barnes [Bar79]. Once the minimum feasible partition has been calculated, Barnes' algorithm is applied to the pallet resulting from the perfect partition reduction, producing the final upper bound.

The upper bound resulting from this combination of methods has been achieved in over 90% of the tested cases. Example bounds are given in table 2.2.

Test	X	Y	a	b	Bound
a	38	38	7	3	68
b	20	20	7	2	28
c	20	15	7	4	10
d	20	15	7	3	14
e	14	11	4	3	12
f	14	13	4	3	15
g	22	16	5	3	23

Table 2.2: Upper bounds for the test box and pallet sizes of table 2.1.

2.3 BASIC APPROACHES

Once an upper bound has been calculated, it remains to determine whether that upper bound can be attained, and if so, how. Since, as has been mentioned earlier, the problem is NP-complete, it is not always practical to try to find the optimal solution, and thus some heuristic measure will be applied.

The results obtained from these heuristic methods can be tested against the expected optimal result in order to determine whether or not they are acceptable.

What heuristic can be applied in evaluating a layout to use for packing a pallet with boxes? The most fundamental and obvious method is simply to pack all the boxes parallel and adjacent to each other starting in one corner of the pallet, as shown in figure 2.6.

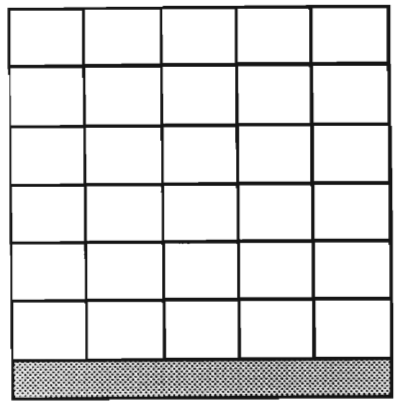


Figure 2.6: A basic packing of 200×150mm boxes on a 1000×1000mm pallet.

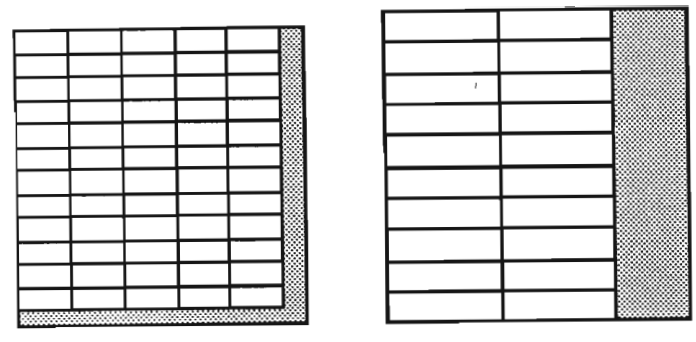
Since the boxes can be packed with either their length or width parallel to the length of the pallet, this results in the number of boxes packed being given by

$$N = \max\left(\left[\frac{X}{a}\right]\left[\frac{Y}{b}\right]; \left[\frac{X}{b}\right]\left[\frac{Y}{a}\right]\right).$$

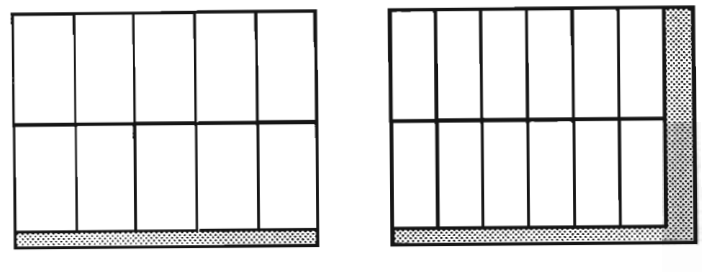
This will be referred to later as the basic method.

In the case of a 200×150mm and a 1000×1000mm pallet, the method packs 30 boxes per layer. The shaded area at the bottom of the pallet (figure 2.6) cannot hold any more boxes, as its width is only 100mm where 150mm is required to hold another row of boxes.

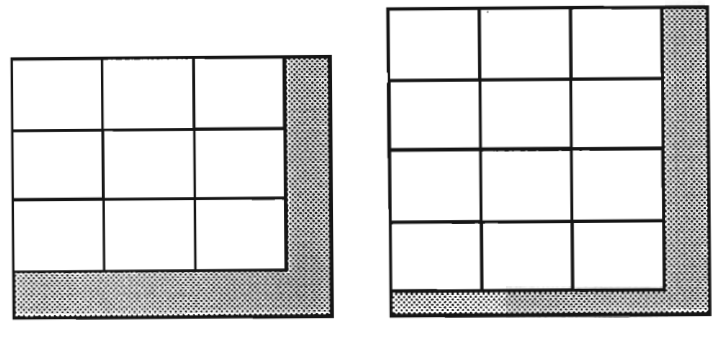
Deighton [Dei91] reports that this method reached upper bound in 23.5% of his 207711 tested cases for packing a large container with boxes. Of the 8565 cases defined by Dowsland [Dow87a], 20.57% attained the upper bound when tested with this basic approach. Example layouts produced are shown in figure 2.7.



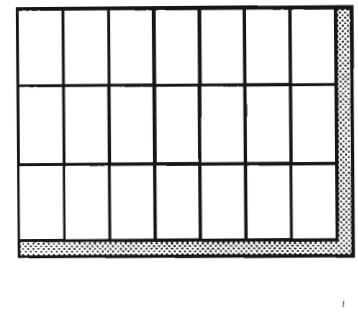
(a) 60 (b) 20



(c) 10* (d) 12



(e) 9 (f) 12



(g) 21

Figure 2.7: Tests of the basic packing layout. The numbers below each figure show the number of boxes packed, and a star indicates that the upper bound has been reached. The dimensions of the boxes and pallets used are as in table 2.1.

This clearly is not the best method to apply in all cases, but does have certain uses as a lower bound against which other methods can be compared. The method does, however, always provide an optimal solution for square boxes.

A point to notice about the method is that in the area where the boxes are packed, there is 100% space usage, and the only waste is in the shaded area (figure 2.6) along the edges of the pallet.

In some cases, it may be possible to improve the layout immediately by adding some boxes, rotated 90° to those already packed, along the edge. For example, consider packing a 5×5 pallet with 2×1 boxes. Using the basic method, the number of boxes that can be packed is 10 (figure 2.8(a)). However, by rotating 2 further boxes through 90° and fitting them along the edge of the pallet 12 boxes can be packed (figure 2.8(b)).

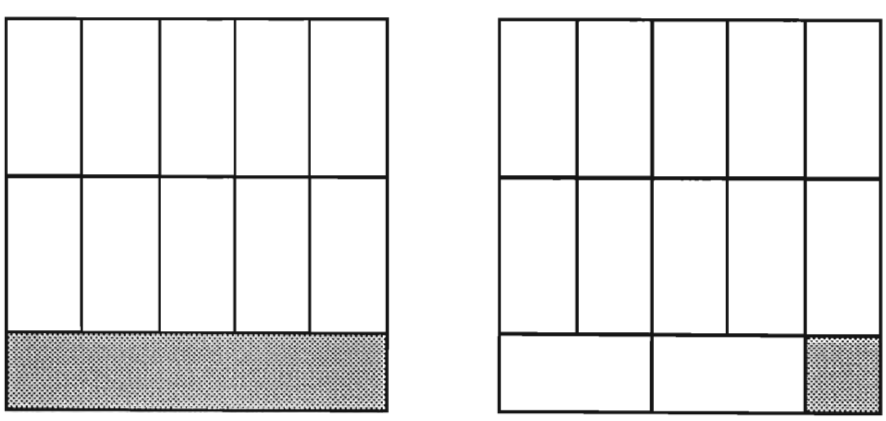


Figure 2.8: Packing 2×1 boxes on a 5×5 pallet. Basic layout (a) is improved by adding boxes along edge of pallet at 90° rotation.

This method thus defines the number of boxes that can be fitted onto the pallet as:

$$N = \max(A,B)$$

where

$$A = \left\lfloor \frac{X}{a} \right\rfloor \left\lfloor \frac{Y}{b} \right\rfloor + \left\lfloor \frac{X \bmod a}{b} \right\rfloor \left\lfloor \frac{Y}{a} \right\rfloor + \left\lfloor \frac{a*(X \operatorname{div} a)}{b} \right\rfloor \left\lfloor \frac{Y \bmod b}{a} \right\rfloor$$

and B is defined similarly by exchanging a and b throughout. The three areas corresponding to the terms in this formula are shown in figure 2.9.

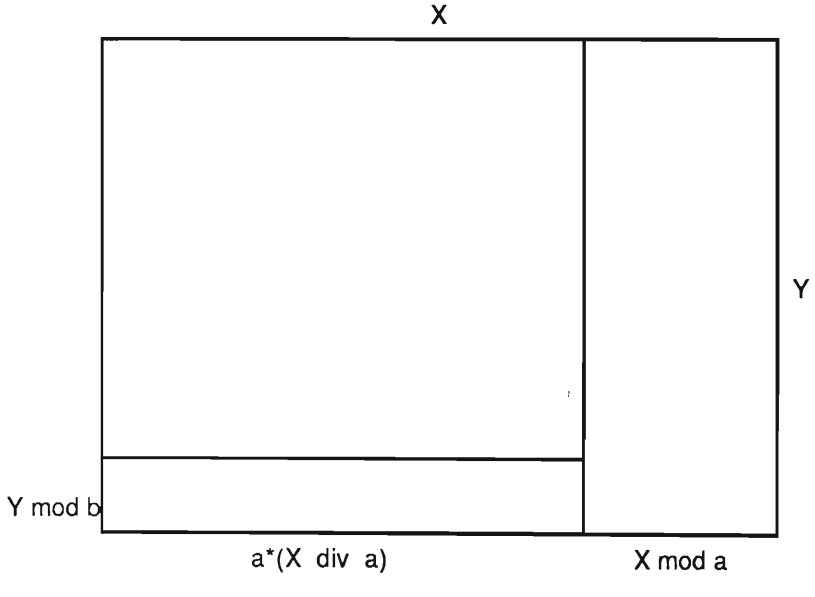


Figure 2.9: The three regions where boxes can possibly be packed on a pallet.

For reference later, this method will be referred to as the improved basic method.

Using this method, the results of the basic method were improved upon in 35.67% of Deighton's examples and 47.76% of Dowsland's. Examples are shown in figure 2.10.

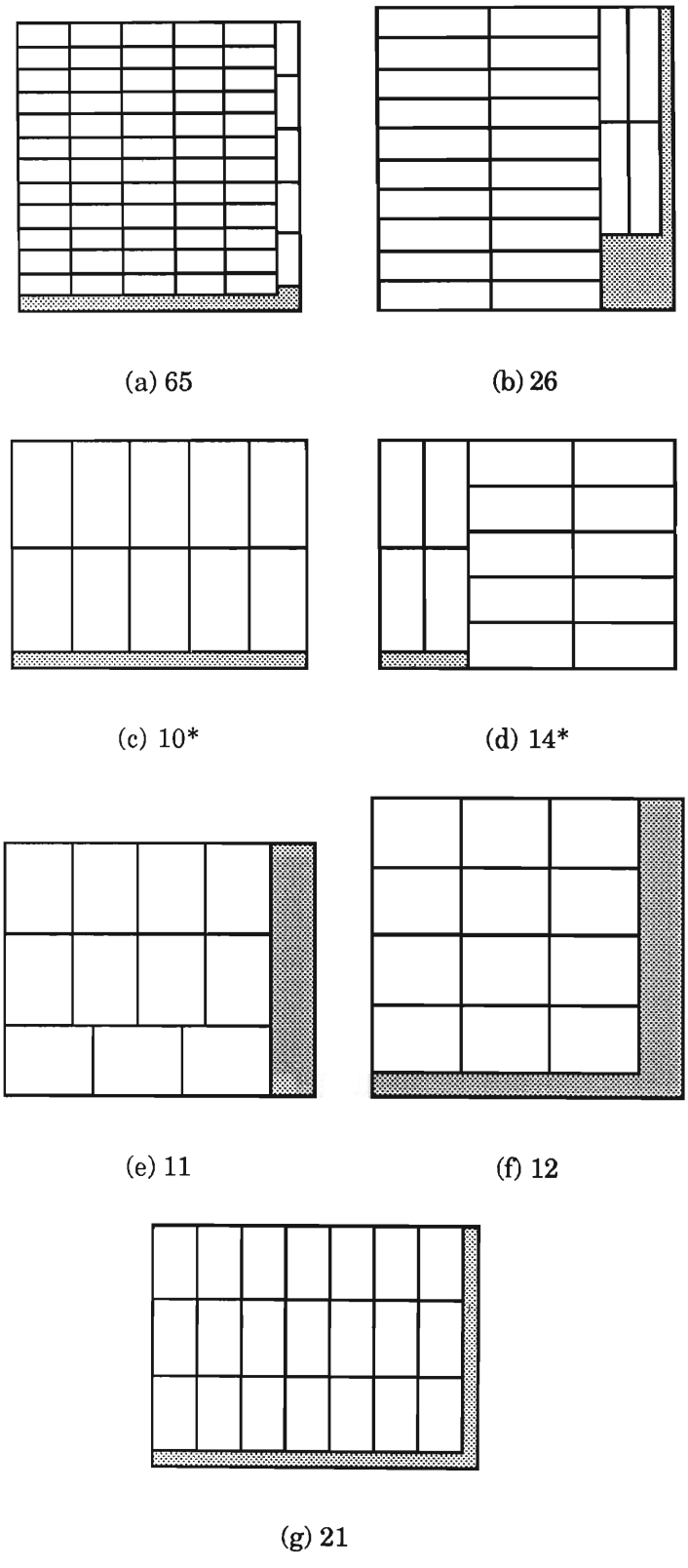


Figure 2.10: The results of applying the extended basic algorithm to the sample set of box and pallet dimensions.

2.4 A BOUNDARY OPTIMISING APPROACH

Steudel first published this heuristic in 1979 [Ste79] with an updated version appearing in 1984 [Ste84]. These algorithms consist of two phases, the first being a perimeter optimisation, and the second an inward projection.

> The objective is to maximise the utilisation of the perimeter of the large rectangle. In the second phase of the algorithm the optimum arrangement of the rectangles along the perimeter is projected inward to fill the centre portion of the large rectangle so as to minimise the amount of unused area. [Ste79]

2.4.1 The basic algorithm

The perimeter optimisation phase uses dynamic programming in order to evaluate the placement of boxes along the perimeter of the pallet so that the maximum length of the perimeter is used.

The formulation is as follows:

$$F_n(S_n) = \text{MAX} \{ X_n*a + Y_n*b + F_{n-1}(S_{n-1}) \}$$

subject to:

$$X_n*a + Y_n*b \leq D_n, \; n=1,2,3,4$$

where:

$F_n(S_n)$ = the maximum value of the sum of length and width placements on edge n of the large rectangle with state variable S_n entering that stage.

X_n = the number of small rectangles of length a placed lengthwise along edge n.

Y_n = the number of small rectangles of width b placed widthwise along edge n.

D_n = dimension of edge n (either X or Y).

S_n = the state variable that describes the initial state for edge n. S_n has 3 possible values:

$S_n = 1$: $X_n = 0$, $Y_n = 2$;

$S_n = 2$: $X_n = 2$, $Y_n = 0$;

$S_n = 3$: $X_n = 1$, $Y_n = 1$.

This geometrically divides the pallet surface into 4 blocks, defined by:

B_1: formed by X_1 and Y_4

B_2: formed by X_2 and Y_1

B_3: formed by X_3 and Y_2

B_4: formed by X_4 and Y_3,

as shown in figure 2.11. Obviously, if any one of the dimensions of these blocks is zero, the block will be empty.

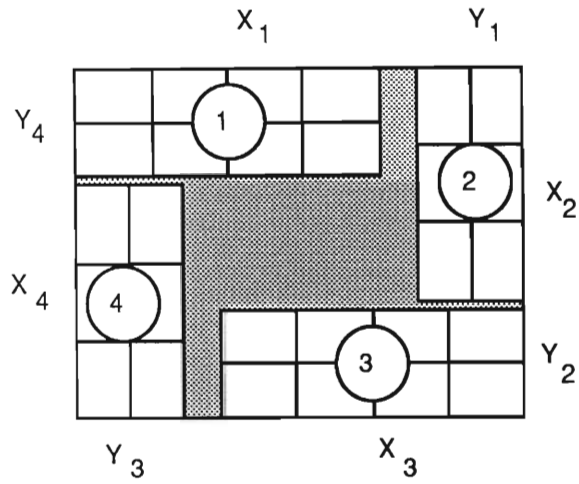


Figure 2.11: The four blocks defined by the perimeter optimisation phase.

The inward projection can take place once the perimeter optimisation has been completed. In this phase, the optimal perimeter placement is projected into the centre of the pallet by filling the four blocks. The theory behind the algorithm is that since the perimeter placement is optimal, this inward projection will result in an overall optimal layout.

This direct inward projection does not necessarily produce an optimal layout, or even a feasible layout because of two possible problems that must be catered for.

> Two potential problems must be considered. For one, a condition of overlap or interference between the blocks could occur. ... The second problem is that the inward projection could result in a layout pattern which has a central hole larger than a small rectangle (box). [Ste79]

Figure 2.11 shows a layout that is clearly not optimal due to the presence of a large 'hole' in the centre of the pallet, while figure 2.12

shows a layout which is not feasible due to the overlap between the centre boxes (shaded region).

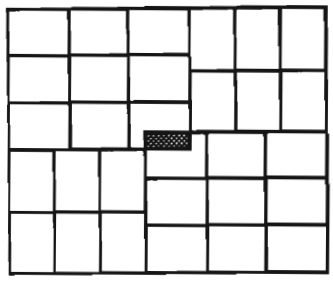


Figure 2.12: A layout showing overlap between blocks 1 and 3.

Overlap between blocks 1 and 3 occurs if:

$$(X_1 + X_3)a > X \text{ and } (Y_2 + Y_4)b > Y$$

while overlap between blocks 2 and 4 occurs if

$$(Y_1 + Y_3)b > X \text{ and } (X_2 + X_4)a > Y.$$

Steudel's method of handling this problem condition [Ste79] was to treat blocks 1 and 2 as fixed. Blocks 3 and 4 are then modified by changing the values of X_3, X_4, Y_2, and Y_3 in order to remove the overlap condition. In the case of figure 2.12, this is done by reducing X_3 by 1 and increasing Y_3 by 1 to obtain the layout in figure 2.13.

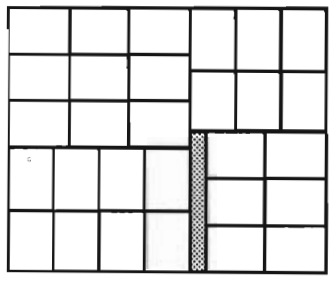


Figure 2.13: The final layout of figure 2.12 after removal of the overlap condition.

In cases where the perimeter optimisation results in a central hole, the suggested solution is to search for another perimeter arrangement of boxes that yields the same optimal perimeter usage. If one is found, then the inward projection phase starts again with this new optimal perimeter placement. If such an arrangement does not exist, then again blocks 1 and 2 are held fixed and blocks 3 and 4 are modified to fill the hole. The layout of figure 2.11 is modified to produce the final layout of figure 2.14.

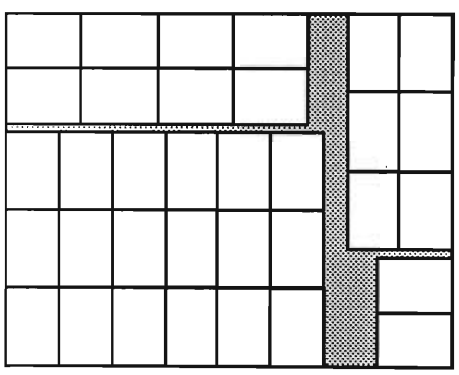


Figure 2.14: The layout resulting from removing the hole in figure 2.11.

This algorithm has also been tested by using the examples of Deighton and Dowsland. The results produced by the basic method were improved in 55.40% and 61.46% respectively. However, the basic method produced better results than this method in 24.16% of Deighton's tests and 13.96% of Dowsland's.

Example results are shown in figure 2.15.

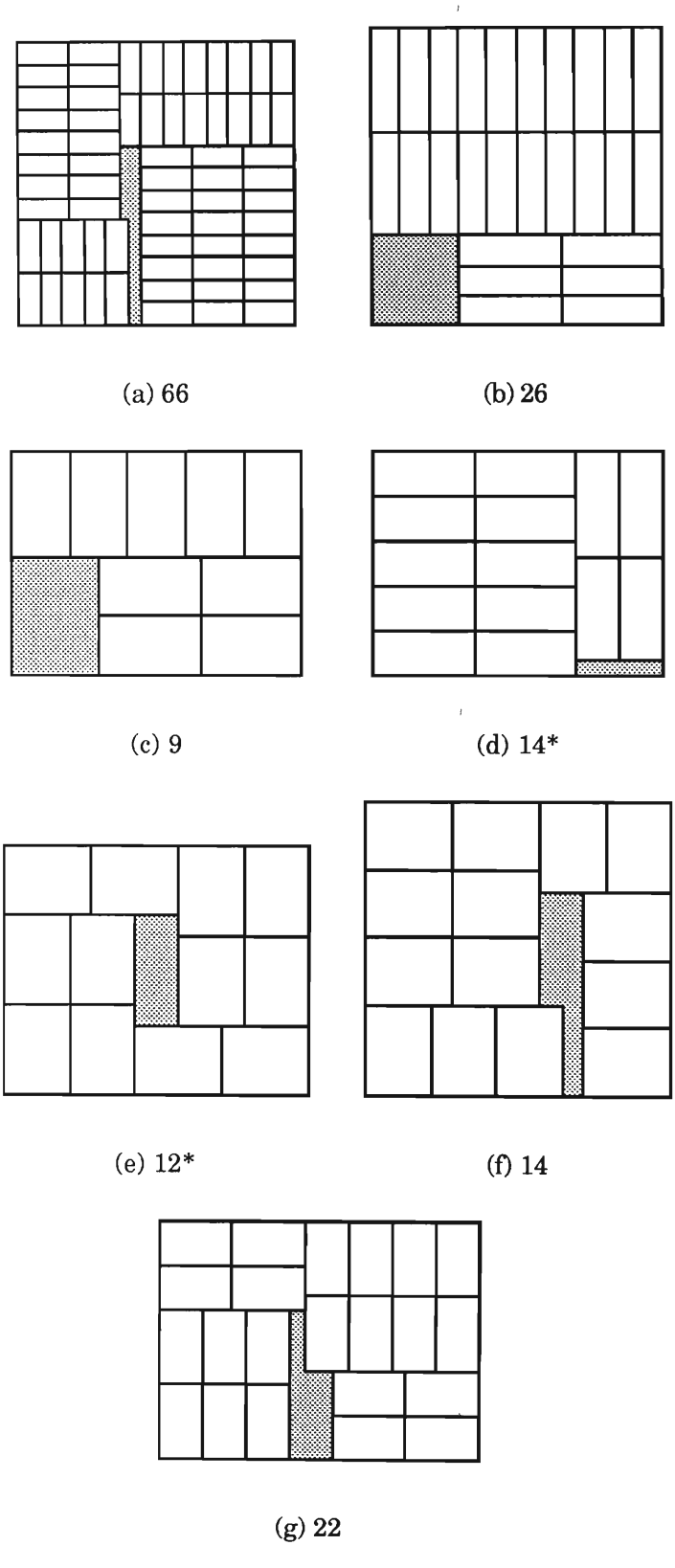


Figure 2.15: Example results produced by Steudel's 1979 algorithm.

Bischoff and Dowsland [Bis82] demonstrated with the example of packing a 48×40 inch pallet with 11×7 inch boxes that simple extensions to the algorithm could result in an improved layout. (See figure 2.16) Instead of holding blocks 1 and 2 fixed and modifying blocks 3 and 4, they held blocks 1 and 4 fixed and modified blocks 2 and 3 to produce a layout containing more boxes.

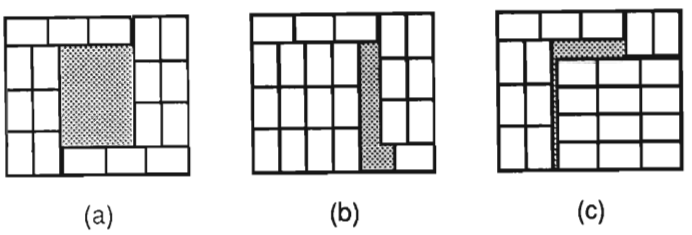


Figure 2.16: Bischoff and Dowsland's objection to Steudel's algorithm. Packing a 48×40 inch pallet with 11×7 inch boxes gives a central hole after perimeter optimisation (a). This is removed by modifying blocks 3 and 4 (b) resulting in 22 boxes per layer. Modifying blocks 2 and 3 gives a layout with 23 boxes per layer (c).

Steudel later updated his algorithm [Ste84] to include several modifications to the strategies for handling each of the problem conditions resulting from the inward projection phase.

2.4.2 Later Updates

In his 1984 paper, Steudel [Ste84] defined the perimeter optimisation phase of the algorithm in the same way as the earlier paper, but extended the inward projection phase of the algorithm to cater for the problems that had arisen in the method. An optional third phase was also added to the algorithm, in which an attempt is made to add boxes at a different vertical orientation once the final layout has been calculated.

In order to remove overlap between blocks, the following two strategies are evaluated, with the one giving the largest number of boxes per layer being selected.

Strategy 1. Treat blocks 1 and 2 as fixed and modify blocks 3 and 4 by changing the values of X_3 and Y_3 to remove the overlap.

Strategy 2. Treat blocks 1 and 2 as fixed. Eliminate either block 3 or 4, whichever is causing the overlap, and then redefine the two adjacent blocks to fill the resulting void. This is demonstrated in figure 2.17.

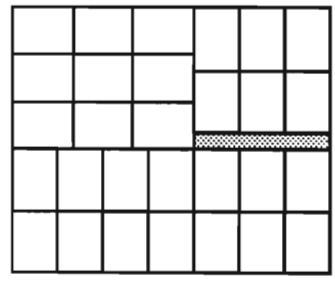


Figure 2.17: The result of applying strategy 2 to the layout of figure 2.12.

In [Ste84], Steudel defined a test for the central hole condition. This involves checking that all the regions are non-empty, and then checking the conditions

$|X_2a - Y_4b| \geq a$ (or b), and

$|X_1a - Y_3b| \geq b$ (or a).

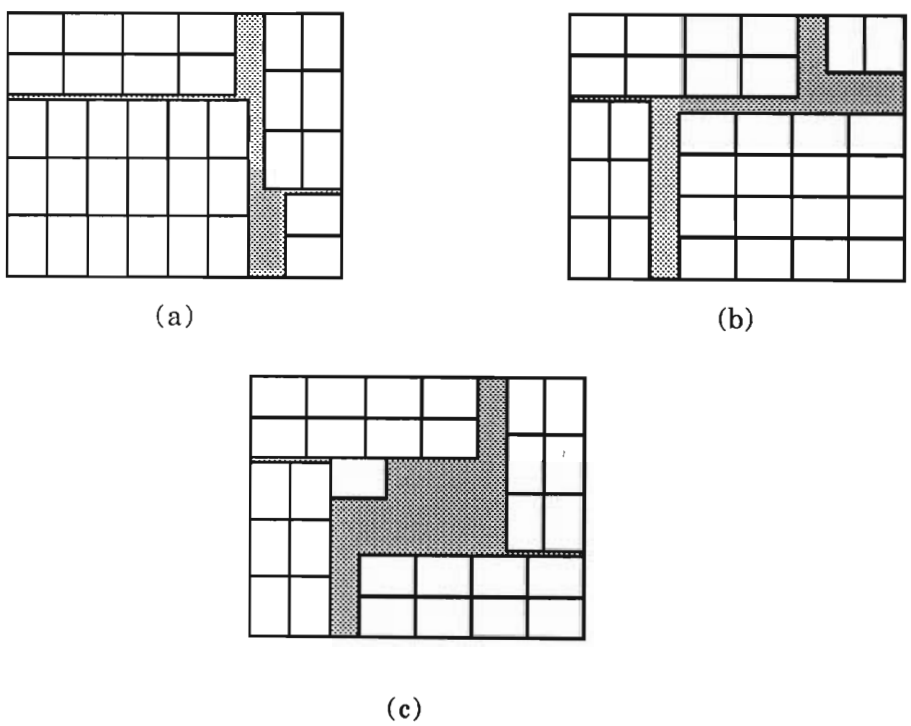
This problem is treated by evaluating the following strategies and selecting the one which results in the maximum number of boxes being packed per layer.

Strategy 3. Treat blocks 1 and 2 as fixed. Expand block 4 to fill the hole and reduce block 3 as required (figure 2.18(a)).

Strategy 4. Treat blocks 1 and 4 as fixed, expand block 3 to fill the hole and reduce block 2 as required (figure 2.18(b)).

Strategy 5. If the central hole is only large enough for one extra box, place one extra block in the hole, independently of the four block convention (figure 2.18(c)).

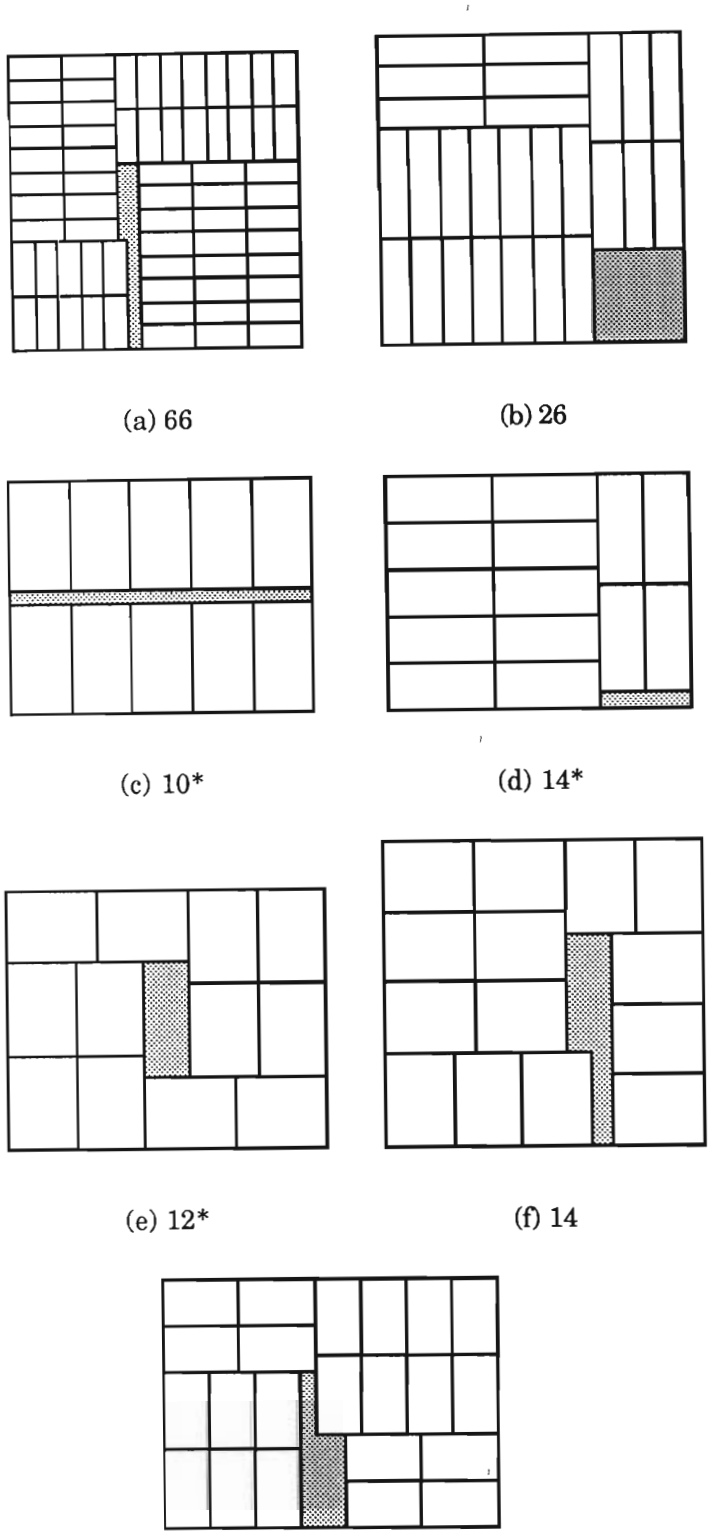
Once each of these strategies have been evaluated, ties are broken in favour of strategy 3 or 4 rather than strategy 5 wherever possible.



(a) (b)

(c)

Figure 2.18: The results of applying the three hole removing strategies to the layout of figure 2.11. Note that in (c), Steudel only allows for 1 box in the central region.



(a) 66 (b) 26

(c) 10* (d) 14*

(e) 12* (f) 14

(g) 22

Figure 2.19: The results of applying Steudel's extended method [Ste84] to the example set of box and pallet sizes.

This algorithm improved on the results of Steudel's earlier algorithm [Ste79] in 36.05% of Deighton's and 37.73% of Dowsland's examples. The improved basic method of section 2.2 produced better results than this method in 6.51% and 5.22% of the tests.

Even with the updated strategies, Steudel's algorithm does not always find an optimal layout. Consider the example of packing boxes of size 230×170mm onto a 1260×1000mm pallet. Perimeter optimisation results in the following values:

$X_1 : 4$ $Y_1 : 2$

$X_2 : 1$ $Y_2 : 1$

$X_3 : 4$ $Y_3 : 2$

$X_4 : 1$ $Y_4 : 1$

which gives a perimeter usage of 3320mm. Inward projection does not result in a central hole large enough for another box, nor does it result in any overlap, and so the number of boxes that can be packed is set at 12. (Figure 2.20 (a))

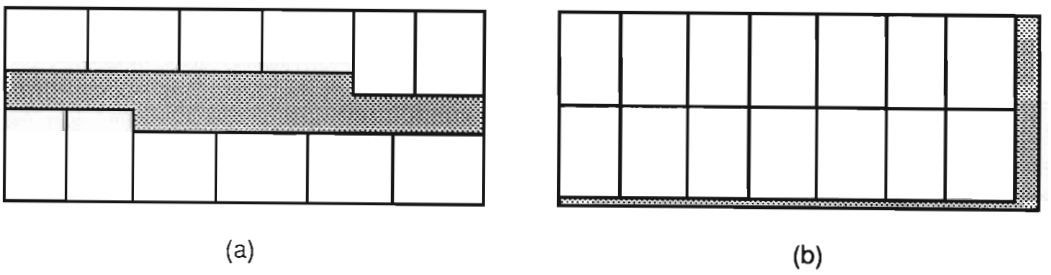


Figure 2.20: The best layout produced by the perimeter optimisation algorithm (a) is not as good as that produced by a sub-optimal perimeter usage (b).

However, a sub-optimal perimeter layout defined by

$X_1 : 7$ $Y_1 : 0$

$X_2 : 0$ $Y_2 : 2$

$X_3 : 7$ $Y_3 : 0$

$X_4 : 0$ $Y_4 : 2$

which only occupies 3300mm along the perimeter of the pallet can be used. This defines two large blocks that overlap in the centre. The overlap condition is be removed by redefining X_3 and Y_4 each to be 0 resulting in 14 boxes per layer (figure 2.20 (b)).

2.4.3 Further Analysis

Steudel's algorithm [Ste79, Ste84] makes the assumption that an optimal perimeter layout will result in an optimal overall layout when it is projected into the centre of the pallet. As has been shown above, this assumption is not always valid.

Sometimes, it may be necessary to use a sub-optimal perimeter layout in order to produce a better overall result. It does, however still make some sense to use combinations of box lengths and widths that best utilise the length of the pallet side being considered.

Analysis of Steudel's perimeter optimising phase reveals that the dynamic programming routine used selects the combination with the maximum length usage out of 81 possible combinations. These combinations correspond to the value of the state variable (S_n) of each edge of the pallet. Since there are three possible values which each state variable can assume, and each of the four state variables can

assume any of these three values independently of the value of any other state variable, there are 3^4 (81) possible combinations to consider.

Since there is such a limited number of possible edge placements to consider, and at most three strategies need to be examined for each of these, instead of selecting the optimal perimeter placement, one could select the one that provides the best overall layout.

The extended algorithm could then be defined as in Algorithm 2.3

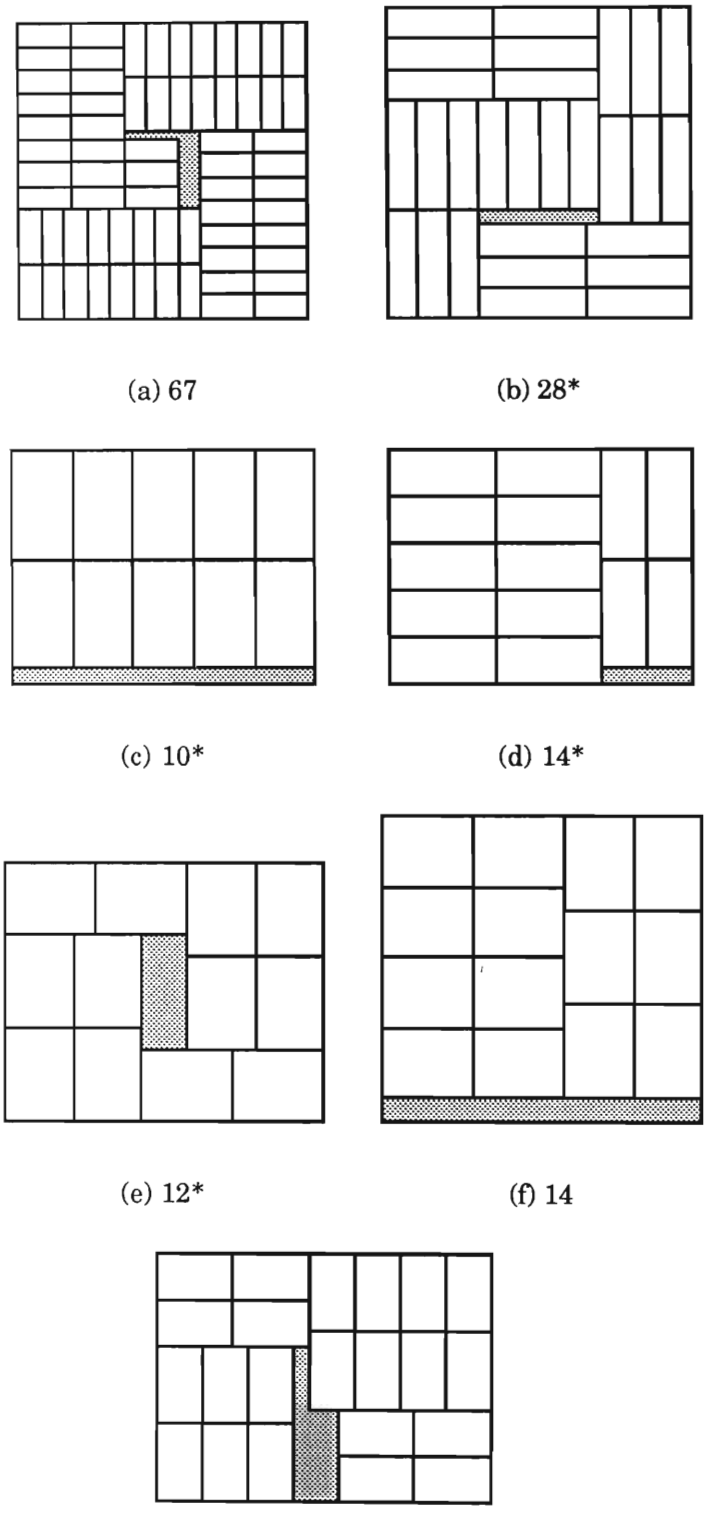
```
Algorithm 2.3
    Extended version of Steudel's algorithm.

proc Extend_Steudel
    for S1 = 1 .. 3 do
        for S2 = 1 .. 3 do
            for S3 = 1 .. 3 do
                for S4 = 1 .. 3 do
                    Set_Up_Perimeter(S1, S2, S3, S4)
                    Project_Inwards
                    Check_Against_Best_So_Far
                enddo
            enddo
        enddo
    enddo
endproc
```

The algorithm for projecting the perimeter arrangement into the centre of the pallet can also be extended slightly. If there exists a central hole, then the strategies can be evaluated as for the 1984 algorithm, except that strategy 5 can be modified to pack more than a single box into the central hole if this is possible.

This unpublished method (referred to later as the Improved Steudel method) produced results that are necessarily as good as those produced by either of Steudel's algorithms, as it contains the optimal perimeter layout as used by Steudel's algorithms as one of the cases that is tested.

It improved on the results of Steudel's updated algorithm [Ste79] in 16.20% of Deighton's tests, and 10.94% of Dowsland's, but still resulted in less boxes than the improved basic method in 1.73% and 1.06% of the test cases.

Examples of the layouts produced are shown in figure 2.21.



(a) 67 (b) 28*

(c) 10* (d) 14*

(e) 12* (f) 14

(g) 22

Figure 2.21: The results obtained by applying the new extended version of Steudel's heuristic to the example data set of table 2.1.

2.5 THE FIXED PATTERN APPROACH

Smith and DeCani published a method which was the forerunner of a group of similar heuristics. The basic algorithm is to select some general layout pattern, and then evaluate all the layouts that conform to that general pattern, selecting the combination with the best overall results as the one to apply.

2.5.1 Smith and DeCani's Heuristic

The Smith and DeCani heuristic [Smi80] examines all layouts that conform to the general four-region pattern. These regions are similar to the four regions of Steudel's algorithm (figure 2.11), but the size of each is calculated in a different way from that used by Steudel.

For each layer, Smith and DeCani's algorithm examines all the layouts of boxes that form a cyclic pattern of four regions of the type shown in figure 2.22, and selects as its solution the layout containing the largest number of boxes.

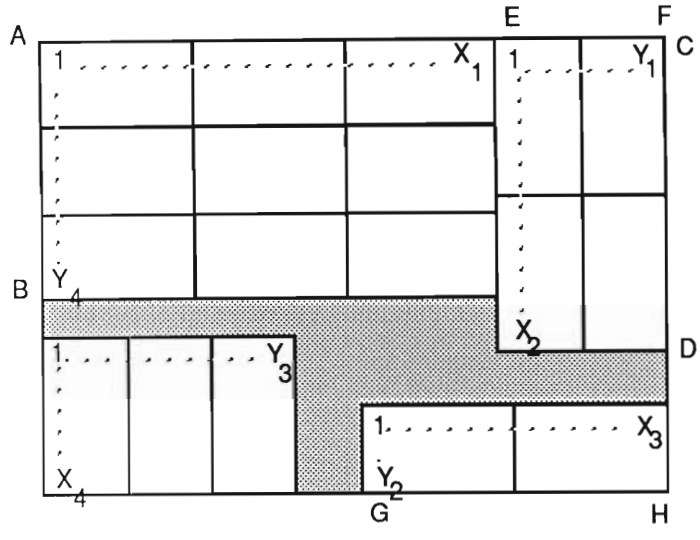


Figure 2.22: The layout pattern considered by Smith and DeCani's heuristic.

Assume that a box of size a×b is to be packed onto a pallet of size X×Y. Clearly, the maximum values for X_1 and Y_4 are given by

$$X_{1\,max} = \left\lceil \frac{X}{a} \right\rceil, \text{ and}$$

$$Y_{4\,max} = \left\lceil \frac{Y}{b} \right\rceil$$

The procedure considers each possible pair of values for X_1 and Y_4,

$$0 \le X_1 \le X_{1\,max}$$

and

$$0 \le Y_4 \le Y_{4\,max}$$

choosing the value of Y_1 and X_4 to be

$$Y_1 = \left\lceil \frac{X - X_1 a}{b} \right\rceil,$$

and

$$X_4 = \left[\frac{Y - Y_4 b}{a}\right]$$

in order to fill the top length and the left side of the pallet.

The smallest value that X_2 can assume in order to allow CD to extend below AB (figure 2.22) is given as

$$X_{2\ min} = \left\{\frac{Y_4 b}{a}\right\}$$

and the maximum value that it can assume is

$$X_{2\ max} = \left[\frac{Y}{a}\right].$$

Each of these values is selected in turn, forcing the value of Y_2 to be fixed at

$$Y_2 = \left[\frac{Y - X_2 a}{b}\right].$$

Similarly, X_3 can assume values between

$$\left\{\frac{Y_1 b}{a}\right\}$$

and

$$\left[\frac{X}{b}\right],$$

setting the value of Y_3 to

$$\left[\frac{X - X_3 a}{b}\right].$$

The allowable values of X_2 and X_3 are restricted by these minimum values in order to ensure that AB ≤ CD and EF ≤ GH (figure 2.22). This

makes the overlap situation which occurred in Steudel's algorithm impossible.

The number of boxes that can be packed using any combination of region sizes is simply

$$T = X_1Y_4 + X_2Y_1 + X_3Y_2 + X_4Y_3$$

In order to cycle through all the combinations, an algorithm such as that shown in algorithm 2.4 can be followed. This algorithm will cycle through all the combinations of region sizes, and return the maximum number of boxes that can be packed.

```
Algorithm 2.4
    Smith and DeCani's algorithm.

proc Smith (X, Y, a, b)
    Max = 0
    for X1 = 0 .. X div a do
        for Y4 = 0 .. Y div b do
            Y1 = (X-X1*a) div b
            for X2 = (Y4*b) div a .. Y div a do
                for X3 = (Y1*b) div a .. X div a do
                    Y2 = (Y-X2*a) div b
                    Y3 = (X-X3*a) div b
                    X4 = (Y-Y4*b) div a
                    if X1Y4 + X2Y1 + X3Y2 + X4Y3 > Max then
                        Max = X1Y4 + X2Y1 + X3Y2 + X4Y3
                        Record( X1, X2, X3, X4, Y1, Y2, Y3, Y4)
                    endif
                enddo
            enddo
        enddo
    enddo
    Return( Max )
endproc
```

Smith and DeCani also prescribe that the algorithm be run twice for each box size. The first run is made, then the box dimensions (a and b) are swapped, and the second run made. The best result of the two runs is then selected as the final.

It is clearly easy to improve the code of algorithm 2.4. For example, lines 4 and 5 could be swapped, as the value assigned to Y_1 does not depend on the current value of Y_4, but only on X_1, and so the calculation can be taken out of the loop. The algorithm has, however, been left in this form for clarity.

This algorithm improved on the results obtained by the improved basic method in 53.35% of Deighton's tests and 51.55% of Dowsland's, and never gave results that were not as good. Examples of the layouts produced are given in figure 2.23.

Thus the Smith and DeCani algorithm does achieve fair results, showing that it is feasible to apply some heuristic measure to the pallet loading problem in order to reduce the time taken in determining which layout to use.

However, Smith and DeCani's algorithm does not achieve optimal results for all box and pallet combinations, and several improvements to it have been found.

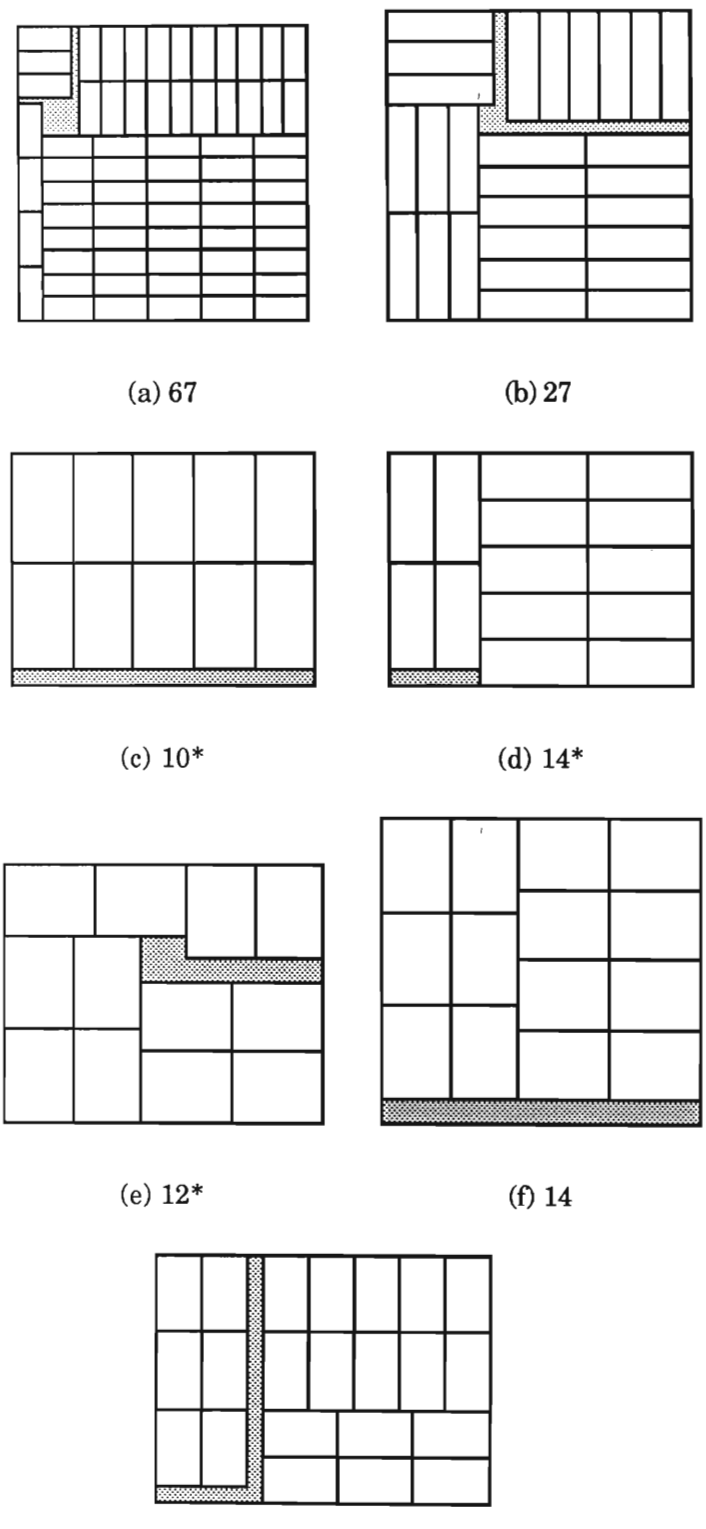


Figure 2.23: The result of applying Smith and DeCani's algorithm to the data of table 2.1.

2.5.2 Bischoff and Dowsland's Method

Bischoff and Dowsland [Bis82] produced a method similar to that of Smith and DeCani. The basic method is still the same, in that a set pattern is chosen, and

> layouts which form a pattern of this type are examined systematically [Bis82].

The two algorithms do however differ in a number of ways. The basic layout pattern obviously is different, with the inclusion of a fifth region into that of Smith and DeCani; and, a different method of cycling through the combinations is used.

The positions of the five regions in the layout pattern considered by this method and the relative orientations of the boxes within the regions are shown in figure 2.24.

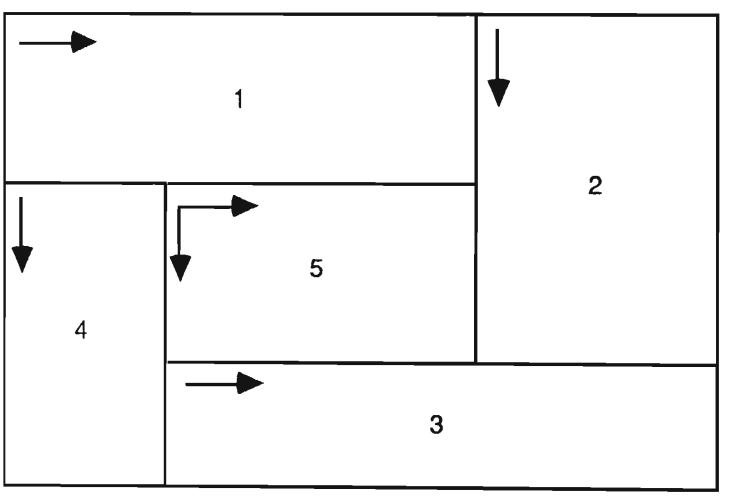


Figure 2.24: The relative positions and box orientations of the five regions of Bischoff and Dowsland's algorithm.

The orientation of the boxes in each of the four main regions is fixed (as indicated by the arrows in figure 2.24), while in the central region, either of the two possible orientations can be assumed so as to pack the maximum number of boxes into that region.

The sizes of the regions are allowed to cycle through all possible values. Unlike Smith and DeCani's algorithm, which included bounds that ensured feasible layouts, this process does not always produce a feasible layout. Infeasibility results from an overlap between diagonally opposite regions in the layout, and can easily be tested for by evaluating the same tests as were used by Steudel [Ste79]. If any overlap is encountered, the layout combination is excluded from further consideration by the algorithm.

In order that the fifth region is as large as possible, the four major regions must be positioned in the corners of the pallet. This results in the length and width of the fifth region being given as:

$$X' = X - (Y_1+Y_3)b; \text{ and}$$

$$Y' = Y - (Y_4+Y_2)b$$

if region 2 extends below region 1 as in figure 2.24, and

$$X' = X - (X_1+X_3)a; \text{ and}$$

$$Y' = Y - (X_4+X_2)a$$

when region 1 extends below region 2.

The number of boxes that can be fitted into this region using the basic method is therefore given as:

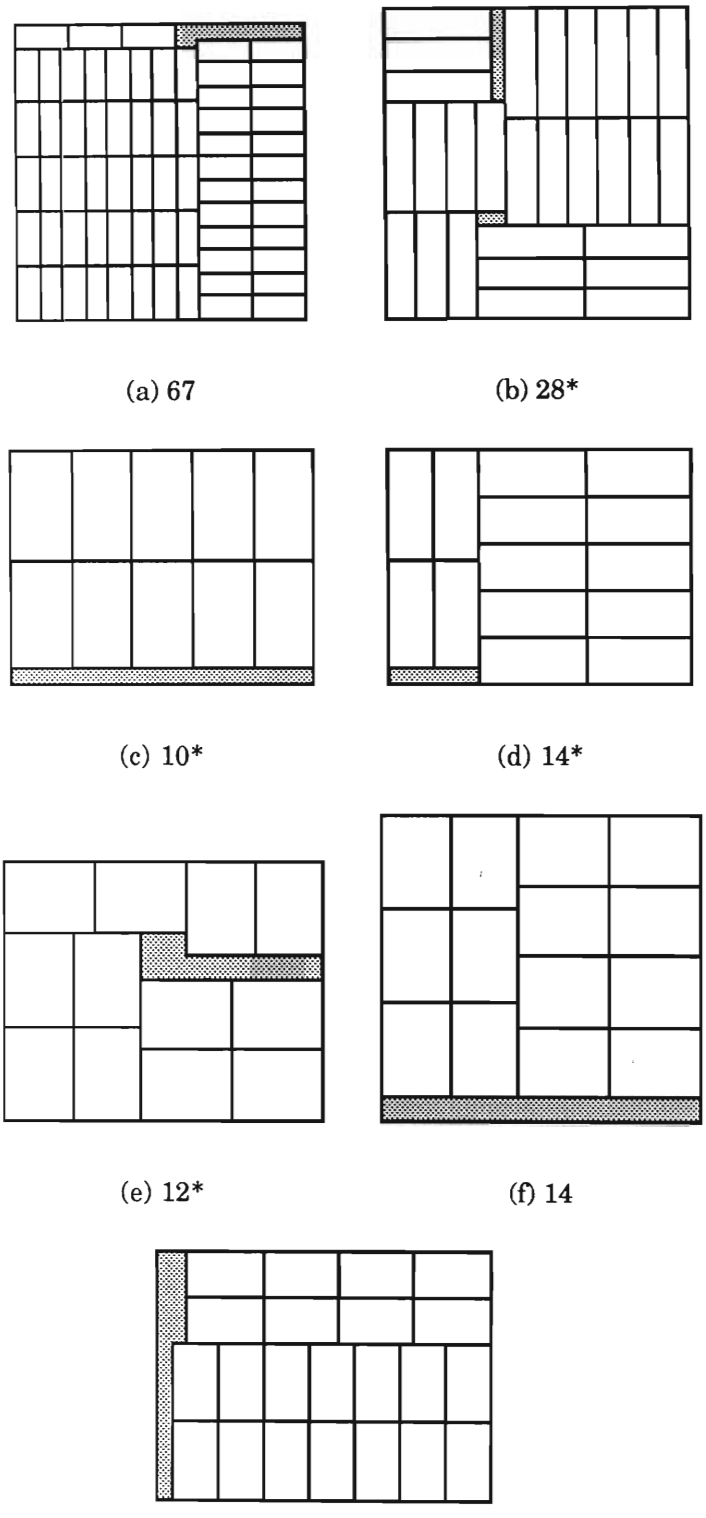
$$N = \max\left(\left[\frac{X'}{a}\right]\left[\frac{Y'}{b}\right]; \left[\frac{X'}{b}\right]\left[\frac{Y'}{a}\right]\right).$$

The final algorithm to use is given as Algorithm 2.5. Once again, the code has been left unoptimised so as to improve readability.

Algorithm 2.5

Bischoff and Dowsland's algorithm.

```
proc Bischoff (X, Y, a, b)
    MaxSoFar = 0
    for X1 = 0 .. X div a do
        for X2 = 0 .. Y div a do
            for X3 = 0 .. X div a do
                for X4 = 0 .. Y div a do
                    Y1 = (X-X1*a) div b
                    Y2 = (X-X2*a) div b
                    Y3 = (Y-X3*a) div b
                    Y4 = (Y-X4*a) div b
                    if No_Overlap then
                        if X2*a > Y4*b then
                            X' = X - (Y1+Y3)b
                            Y' = Y - (Y2+Y4)b
                        else
                            X' = X - (X1+X3)a
                            Y' = Y - (X2+X4)a
                        endif
                        ij = max((X' div a)*(Y' div b);
                                 (X' div b)*(Y' div a))
                        if ab + cd + ef + gh + ij > MaxSoFar then
                            MaxSoFar = ab + cd + ef + gh + ij
                            Record(a, b, c, d, e, f, g, h, i, j)
                        endif
                    endif
                enddo
            enddo
        enddo
    enddo
    Return( MaxSoFar )
endproc
```



(a) 67 (b) 28*

(c) 10* (d) 14*

(e) 12* (f) 14

(g) 22

Figure 2.25: The example results demonstrating Bischoff and Dowsland's five region algorithm.

This algorithm improved on the results produced by Smith and DeCani's algorithm in 2.99% and 2.55% of the tested cases, and never produced results that were not as good as Smith and DeCani's.

See figure 2.25 for examples of the layouts produced by Bischoff and Dowsland's algorithm.

2.5.3 Further Improvements

Although, as has been shown, the Bischoff and Dowsland algorithm improves on that of Smith and DeCani in some cases, it still does not provide the optimal solution for every box and pallet combination. Some improvements have been made to the algorithm, and these are presented in this section.

A detailed analysis of the various layouts considered by the Smith and DeCani algorithm shows that a large number of clearly sub-optimal layouts are considered. For example, when packing a 20×20 pallet with a 7×2 box, all the layouts of figure 2.26 were considered as possibilities. Every one of these layouts is clearly sub-optimal because of the presence of a large central hole (similar to that defined by Steudel [Ste79]), because one of the dimensions of some region is zero, or simply because it is a bad choice of dimensions.

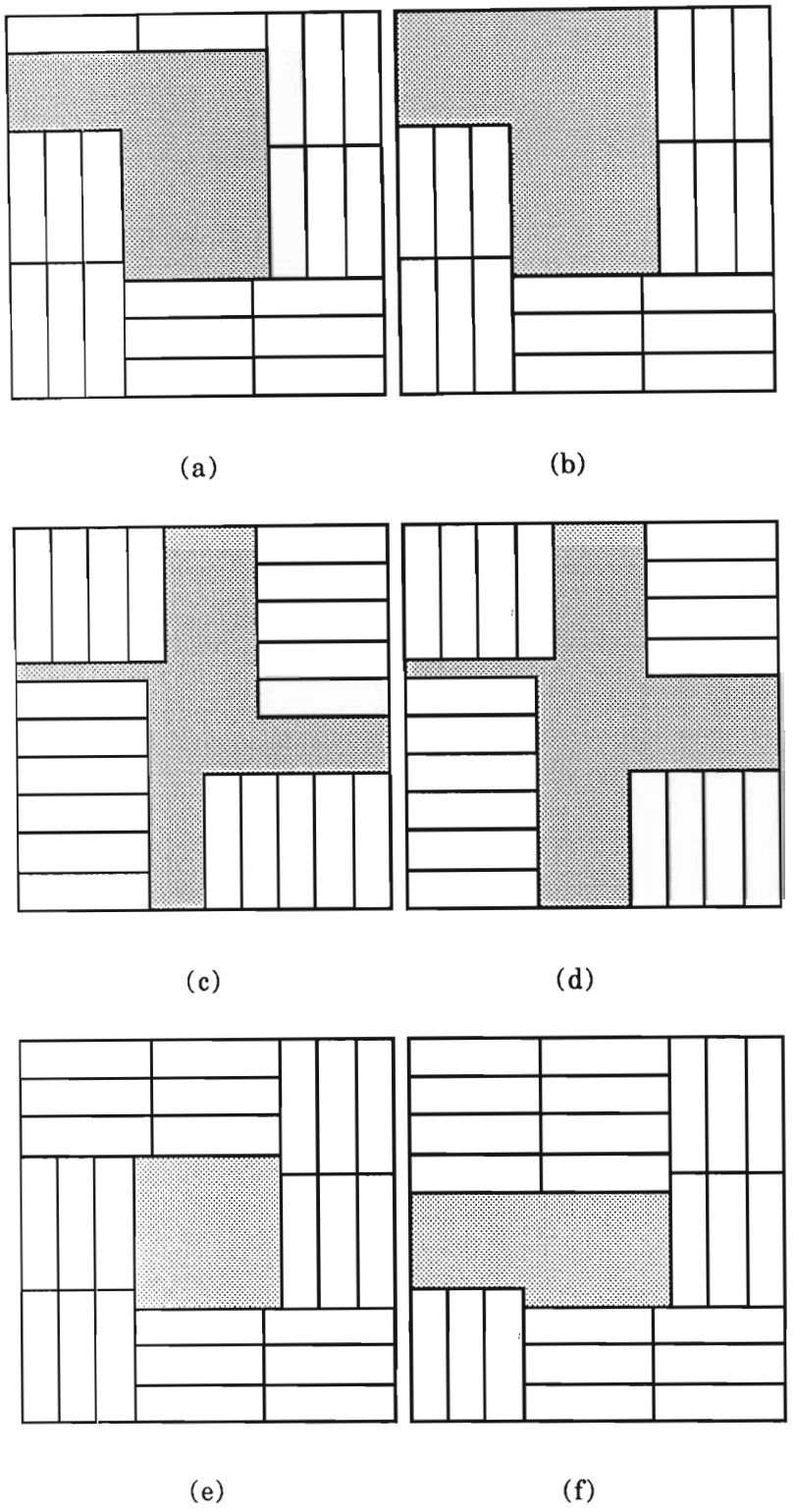


Figure 2.26: Several clearly sub-optimal layouts that are considered by Smith and DeCani's algorithm.

The first, and most obvious improvement to the method is to allow for an efficient means of filling the central region. For some of the

combinations of the region sizes around the pallet this fifth region can be quite large, and packing it as a single region may not be optimal.

Since filling the central region can be viewed as the same problem as packing a pallet, any of the pallet loading algorithms can be applied. In particular, the improved basic method of section 2.2 which uses a main region and allows rotated boxes along the edges has been used with some success [Dei91]. This results in a 7-region layout pattern as shown in figure 2.27.

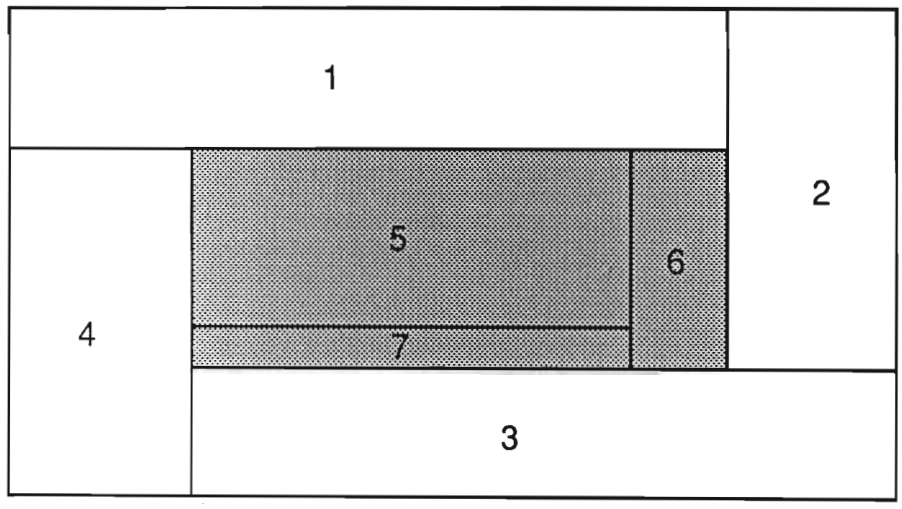


Figure 2.27: The format of the 7-region layout pattern being introduced. The three shaded regions correspond to using the extended basic method to fill the central hole in Smith and DeCani's layout

Recursively calling one of the heuristic measures has also been attempted, but this proved too time consuming for practical application. Even when the recursion is limited to only one level, several hours may be required to define a packing layout.

Another improvement has also been suggested [Dei91]. This involves an 11-region layout as shown in figure 2.28 that is a superset of Smith

and DeCani's 4-regions, Bischoff and Dowsland 5-regions and the new 7-region layout.

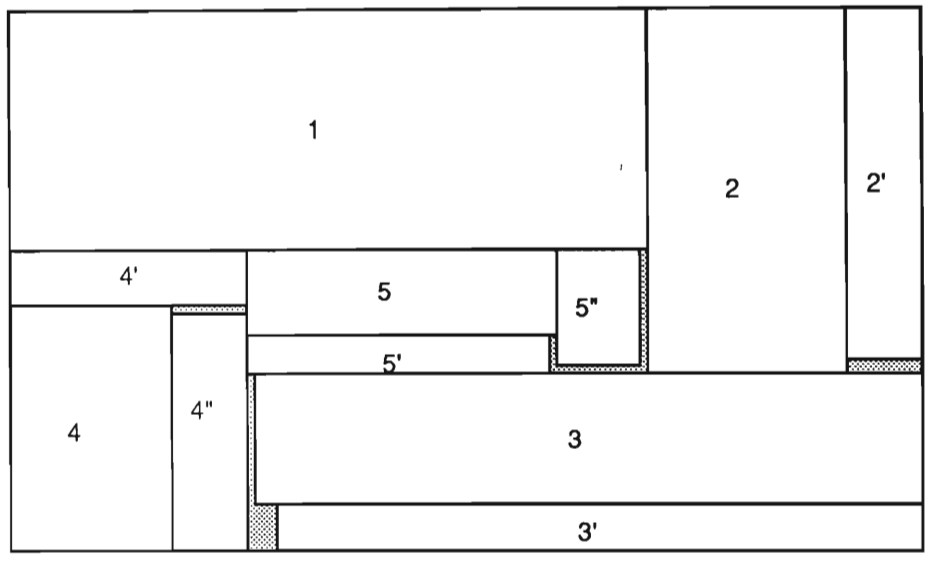


Figure 2.28: The relative positions of the 11 regions in Deighton's improved layout.

The algorithm allows the sizes of the four major regions (1, 2, 3 and 4) to vary, covering as much of the pallet as possible (as does the Smith and DeCani algorithm), while the dimensions of the other regions are defined to fill any remaining space around these regions, and thus are fixed.

The motivation for this extension results from a detailed analysis of Smith and DeCani's algorithm. Consider for example the configuration shown in figure 2.26(d). There is space for several further regions of boxes, as shown by the cross-hatched boxes in figure 2.29.

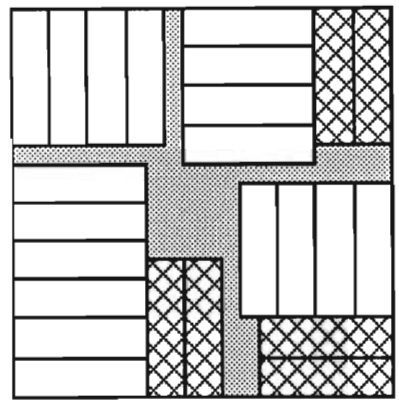


Figure 2.29: The layout resulting from filling the empty space in figure 2.26 (d).

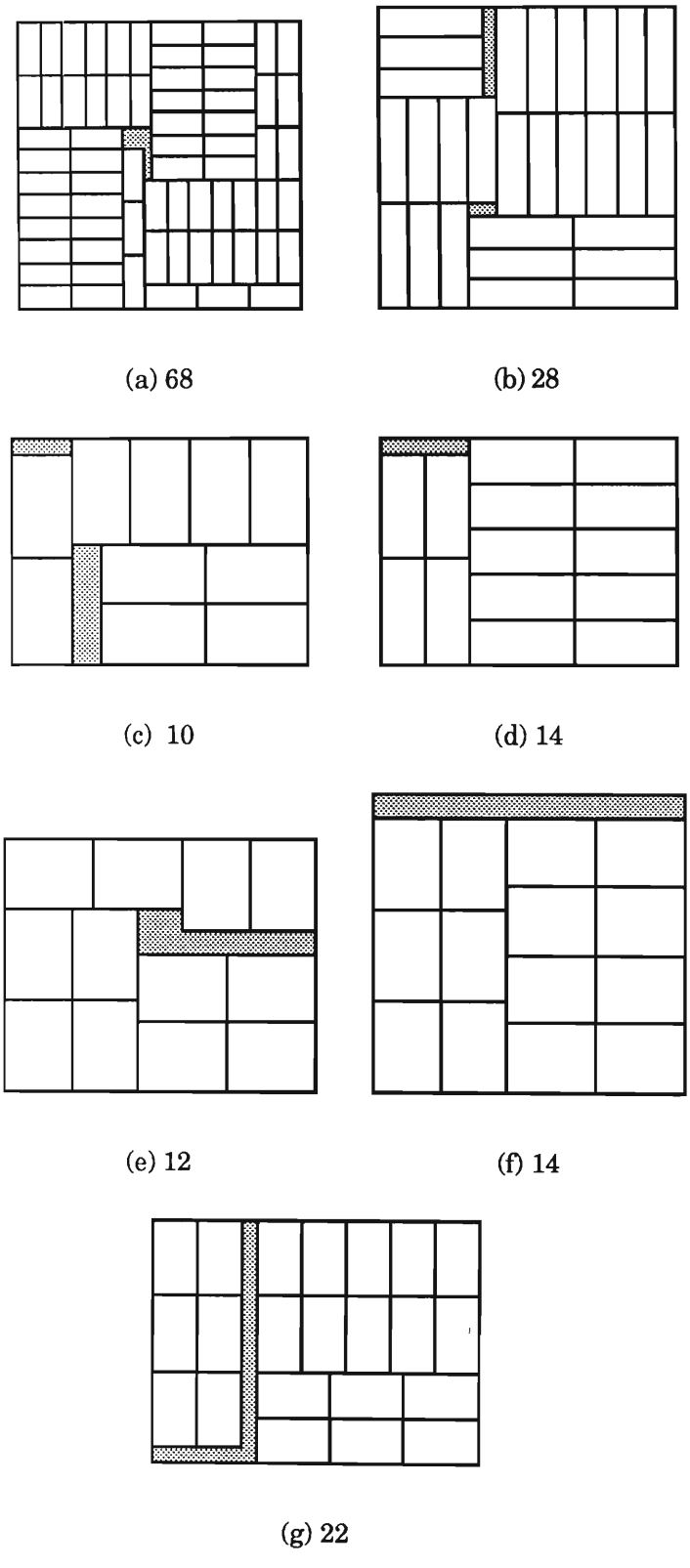
The algorithm follows the approach of Smith and DeCani in that two passes are made, each with a different orientation being used for positioning boxes in each region. The execution time of the algorithm can be reduced by noting that when the greater of the box's dimensions is parallel to the length of the container in region 1, region 2' will necessarily be empty, as will regions 3' and 4". Similarly, when the orientation of boxes in region 1 is rotated, region 4' will be empty. These restrictions can be coded into the algorithm to reduce the number of calculations made within the loops.

The algorithm can be stated as in algorithm 2.6.

Algorithm 2.6

Deighton's new 11-region algorithm

```
proc 11_Regions( X, Y, a, b)
    Max = 0
    for X1 = 0 .. X div a do
        for Y1 = 0 .. Y div b do
            for X2 = {(Y1*b)/a} .. Y div a do
                Y2 = (X-X1*a) div b
                for X3 = {(Y2*b)/a} .. X div a do
                    Y3 = (Y-X2*a) div b
                    X4 = (Y-Y1*b) div a
                    Y4 = (X-X3*a) div b
                    X2' = (X-X1*a-Y2*b) div a
                    Y2' = (X2*a) div b
                    X3' = (Y-X2*a-Y3*b) div a
                    Y3' = (X3*a) div b
                    X4' = (X-X3*a) div a
                    Y4' = (Y-Y1*b-X4*a) div b
                    X4" = (X-X3*a-Y4*b) div a
                    Y4" = (X4*a) div b
                    X5 = (X-Y2*b-X2'*a-Max(Y4*b+X4"*a,X4'*a)) div a
                    Y5 = (Y-Y1*b-Y3*b-X3'*a) div b
                    X5" = (Y-Y1*b-Y3*b-X3'*a) div a
                    Y5" = (X-Y2*b-X2'*a-Max(Y4*b+X4"*a,X4'*a)- X5*a) div b
                    Y5' = (X5*a) div b
                    X5' = (Y-Y1*b-Y5*b-Y3*b-X3'*a) div a
                    T = X1*Y1+X2*Y2+X3*Y3+X4*Y4+X5*Y5+X2'*Y2'+X3'*Y3'+X4'
                        *Y4'+X5'*Y5'+X4"*Y4"+X5"*Y5"
                    if T > Max then
                        Max = T
                    endif
                enddo
            enddo
        enddo
    enddo
    Return (Max)
endproc
```



(a) 68 (b) 28

(c) 10 (d) 14

(e) 12 (f) 14

(g) 22

Figure 2.30: Example results of the new 11-region algorithm.

This method has resulted in fair success. Deighton reports an improvement in 0.56% of his 207711 tested cases over the method of Bischoff and Dowsland, resulting in the upper bound being reached in 85.8% of the tests. The upper bound was also reached in 94.34% of Dowsland's test cases. Figure 2.30 shows examples of the layouts produced.

2.6 AN EXACT ALGORITHM

As has been noted earlier, and is commonly stated in the literature, the pallet loading problem is NP-complete. A consequence of this is that it is unlikely that an efficient algorithm will be found that will solve the problem exactly.

Despite this, Dowsland has attempted to develop an algorithm that will generate optimal results for any box and pallet combination within a given range. The basis of the algorithm is a graph theoretic formulation of the problem [Dow87].

The formulation of the problem is as follows:

> Given the problem of packing a set of a by b boxes ($a \geq b$) onto an X by Y pallet ($X \geq Y$) we define a 'pallet loading' graph G_{XYab} to be the graph whose vertices represent the set of possible box positions on the pallet such that two vertices are adjacent if the box positions they represent overlap. [Dow87]

In order to produce a solution to the pallet loading problem, a maximum subset of vertices must be found which are mutually non-adjacent. That is, a maximum stable set must be found in the graph.

The problem of finding maximum stable sets in a graph is known to be NP-hard [Lou82], and so there has been no saving in modelling the pallet loading problem in this way.

Dowsland [Dow87] shows that the pallet loading graph is a large graph, having

$$|V| = (X+1-a)(Y+1-b) + (X+1-b)(Y+1-a)$$

vertices. For example, packing of a 400×300mm box onto a 1000×1000mm pallet produces a graph of 842602 vertices - this for a layout that has an upper bound of 8 boxes as the maximum that can be packed. An example of the pallet loading graph for packing a 5×5 pallet with 3×2 boxes is given in figure 2.31. Clearly, it is not feasible to attempt to solve problems of this magnitude. It is therefore necessary, first, to reduce the size of the graph.

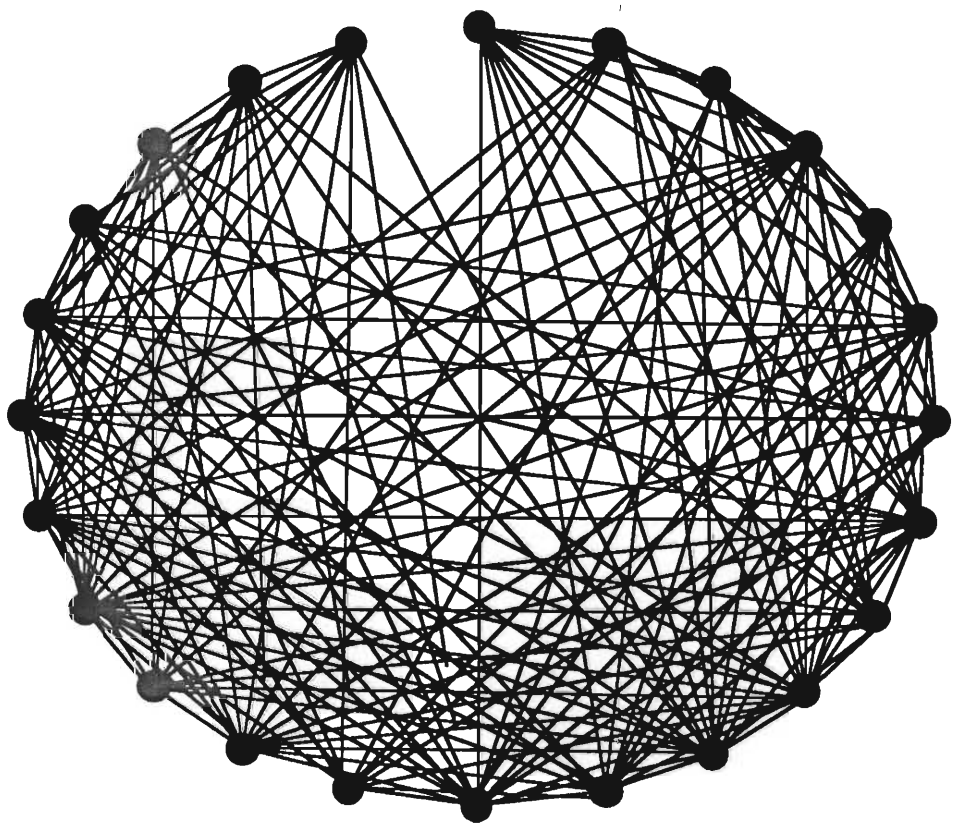


Figure 2.31: The graph used to evaluate the optimal packing of 3×2 boxes on a 5×5 pallet. ($G_{5,5,3,2}$)

2.6.1 Reducing The Problem Size

Two methods for reducing the size of the graph have been discussed by Dowsland, who suggests that they be used in tandem in order to achieve the best results.

The first reduction method uses an earlier result due to Dowsland [Dow84]. This states that the solution to a packing problem depends not only on the absolute dimensions of the pallet and box being packed, but on the integer combinations of box lengths and widths which can be fitted into the pallet dimensions. Using this result so called equivalence classes of problems can be defined such that each problem in a class has the same optimal layout as every other problem in the same class. If one can find this optimal layout for any one box and pallet combination, the same layout can be applied to every other problem set in the class. In particular, the problem that gives the graph with the least number of vertices can be used to produce a solution.

The above example of a 400×300 box and 1000×1000 pallet is equivalent to a 4×3 box and a 10×10 pallet. The graph for this reduced problem contains only 112 vertices, clearly a substantial saving over the 842602 for the original problem, but still a fairly unmanageable size.

The second reduction method follows from the observation made in another of Dowsland's earlier papers. [Dow85] The ability to reduce any layout to a perfect partition layout as discussed earlier implies that any box can be shifted so that it is placed at a point that is an integral combination of box dimensions from the left and top of the pallet. There is, therefore, no need to define vertices in the pallet loading graph which do not correspond to points that can be expressed in such a way.

Applying this restriction to the same example as before results in a graph that contains only 40 vertices. This graph can then be used to determine the layout for the original packing problem. Similarly, the graph $G_{5\,5\,3\,2}$ of figure 2.31 is reduced to that of figure 2.32, which is much more manageable.

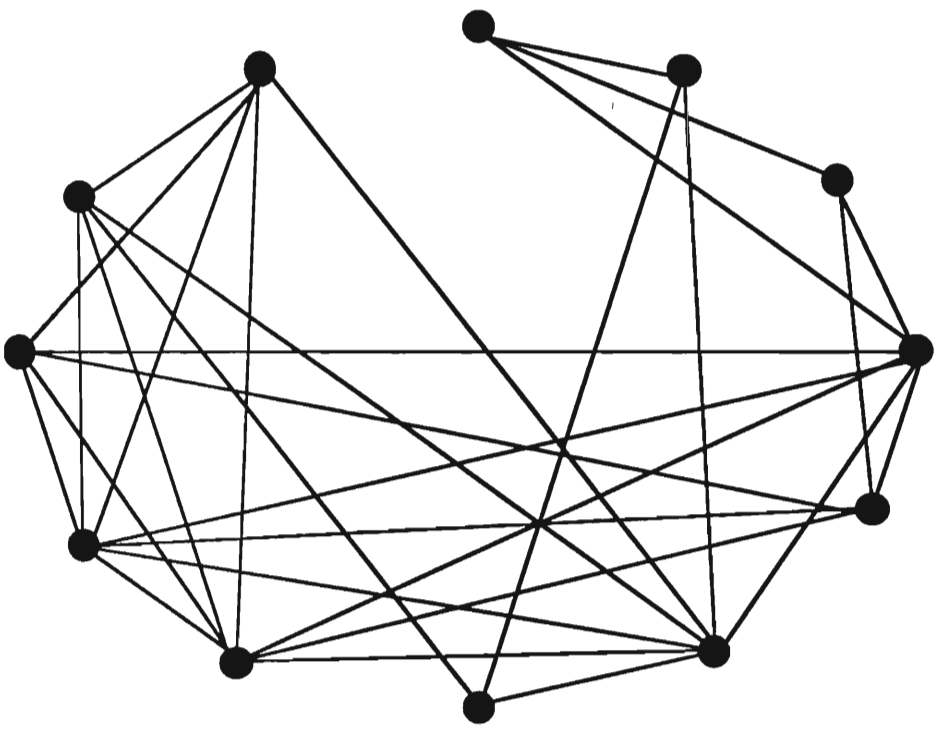


Figure 2.32: The reduced form of the graph for packing 3×2 boxes onto a 5×5 pallet.

Once the graph has been formed, it still remains to find the maximum independent set. Dowsland suggests the method of Loukakis and Tsouros [Lou82] although she points out that any graph theoretic algorithm to solve the problem can be used equally successfully. (See Appendix B for a description of the Loukakis and Tsouros algorithm.)

The method has been implemented and tested with several packing problems. The graphs used must necessarily be small in order for the computation time required to be reasonable. A graph of 40 nodes took around 5 minutes to solve. One of 220 nodes ran for 3 days on an IBM compatible PC without producing any relevant output. Thus the algorithm in this form seems to be of little practical use.

2.6.2 Introducing The Problem

The problem of the vast computation time required to solve the pallet loading problem had been tackled by Dowsland and published two years before the paper describing the algorithm [Dow85a]. This paper describes the same algorithm as the paper discussed above [Dow87], but places different amounts of emphasis on various parts of the algorithm.

Dowsland [Dow85a] states that none of the graphs tested directly with the Loukakis and Tsouros algorithm produced results within reasonable time constraints, and thus further restrictions need to be placed on the algorithm.

The first modification to the algorithm at this stage is to introduce the upper bound estimate that was discussed in section 2.2.

> If we make the assumption that the bound is always correct then it can be added to the basic search tree, both as a bounding condition to avoid searching branches which can never yield a set of the desired size, and to stop execution once a set of this size has been reached. [Dow85a]

Of a limited sample set of problems, Dowsland reported that around 77% were solved within the required time constraints [Dow85a]. This still leaves too many problems which are not solved within the specified time for the algorithm to be generally acceptable, and so more information from the physical problem needs to be incorporated into the algorithm in an attempt to tighten the bounds on the search.

At this stage, the upper bound is again used (this time indirectly) in order to reduce the amount of testing required.

The upper bound can be used to determine the maximum amount of wasted space in the final packing.

Wasted_Space = X*Y - a*b*Upper_Bound

Since the search procedure of Loukakis and Tsouros [Lou82] works by building up partial layouts, to which nodes representing boxes are added, it is possible to determine the amount of space in the partial layout which can never be used by adding another box to that layout. Whenever this wasted space becomes larger than the total wasted space in an optimal layout, the algorithm can backtrack.

By adding this restriction to the earlier algorithm, a new algorithm was developed which

> makes maximum use of this idea by defining an ordering on the vertices of G_{XYab} which means that wasted areas can be recognised early in the tree. [Dow85a]

Using this restricted search algorithm, 97 out of 100 tests performed by Dowsland were solved within the specified time [Dow84], excluding the time required for setting up the graph.

2.6.3 Saving Time

There are still the problem 3% of cases for which more than the acceptable limit on computation time was required. In yet another paper [Dow87a], Dowsland described a computer package that can guarantee an optimal layout within five minutes on a PC.

It has been shown that an exact solution to the problem can be found for every case (e.g. by using the exact algorithm described by Dowsland [Dow85a]) but the execution time required may well exceed the time constraints that would make it feasible.

In order to solve this problem, Dowsland has suggested a computer package that will combine an algorithmic analysis of the solvable problems with a database of previously calculated results for the problems which require more than the specified limit on computation time [Dow87a].

The approach suggested is to attempt to solve all the possible problems within the given time limit. Those problems which require more than the specified amount of time can be solved on a mainframe and their results stored in a database. When a user requests a layout for a particular box on a given pallet, the program first searches the database. If the results for the given problem are contained in the database, they can be presented to the user immediately. If they are not in the database, they can be calculated (within reasonable time constraints) and then presented.

The problem with creating such a database is that all possible problems must be attempted. Since there are obviously an infinite set of box and pallet sizes that could be used, this is not feasible. In order to overcome this, Dowsland returned to the concept of equivalence classes of packing problems. It is obviously clear that only one combination from each equivalence class needs to be tested, as every equivalent combination has the same optimal solution. A further restriction on Dowsland's test set is that only common box and pallet dimensions are tested. This restricts the box and pallet ratios to:

$$1 \leq \frac{a}{b} \leq 4$$

$$1 \leq \frac{X}{Y} \leq 2$$

$$1 \leq \frac{(X*Y)}{(a*b)} \leq 51$$

As long as a manageable finite set of box and pallet dimensions can be generated, the pallet loading problem will be solved. Dowsland gives a description of a manner of determining the box and pallet sizes which cover this range, resulting in 8565 unique combinations. The method is as follows

Since any packing problem (X', Y', a', b') is contained in a region of the three-dimensional pallet chart [Dow84] bounded by the constraints:

$$na + mb \leq X$$

$$na + (m+1)b > X, \forall n, m \text{ s.t. } X-b < na + mb \leq X$$

$$a(n_{max}+1) > X,$$

together with a similar set of constraints for Y, every region has at least one edge of the form:

$$X = n_x a + m_x b$$

$$Y = j_y a + k_y b$$

which moves between regions when

$$na + mb = n_x a + m_x b$$

or

$$ja + kb = j_y a + k_y b$$

which occurs when the box ratio $\frac{a}{b}$ satisfies

$$\frac{a}{b} = \frac{m-m_x}{n_x-n} = \frac{m_x-m}{n-n_x}$$

or

$$\frac{a}{b} = \frac{k-k_y}{j_y-j} = \frac{k_y-k}{j-j_y}$$

Since the ratios between X and Y, a and b and XY and ab are within confined ranges, the values in the above equation can be given by

$$m_x \leq \frac{X}{b}$$

$$n_x \leq \frac{X}{a}$$

$$k_y \leq \frac{Y}{b}$$

$$j_y \leq \frac{Y}{a}$$

$$m \leq \frac{X}{b} + 1$$

$$n \leq \frac{X}{a} + 1$$

$$k \leq \frac{X}{b} + 1$$

$$j \leq \frac{X}{a} + 1$$

As none of these values may be negative, they define a finite set of box ratios (a/b) at which an edge moves between regions. This is determined by examining the maximum values that a and b may assume.

The maximum value for a (a_{max}) occurs when

$$m_x = \left[\frac{X}{b}\right] + 1$$

and

$$m = 0$$

then

$$a_{max} = \text{Max} \left[\frac{X}{b}\right] + 1$$

$$\leq \text{Max} \sqrt{(XY/ab)(X/Y)(a/b)} + 1$$

$$= \text{Max} \sqrt{51*2*4} + 1$$

$$= \sqrt{408} + 1$$

$$= 21$$

Similarly, b_{max} can be shown to equal 11. Thus for the range of problems considered by Dowsland, the set of box ratios at which an edge moves between regions is given by $\frac{c}{d}$ such that

$c, d > 0$, integers

$c \leq 21$

$d \leq 11$

$$1 \leq \frac{c}{d} \leq 4$$

and

c, d mutually prime.

This process results in 79 values. By adding the two end-point ratios (1/1 and 1/4) and taking one point between each consecutive pair of ratios, 80 box ratios are found, which collectively cut every region in the 3 dimensional pallet chart at least once.

A covering set of problems can thus be calculated by considering all the pallets of the form

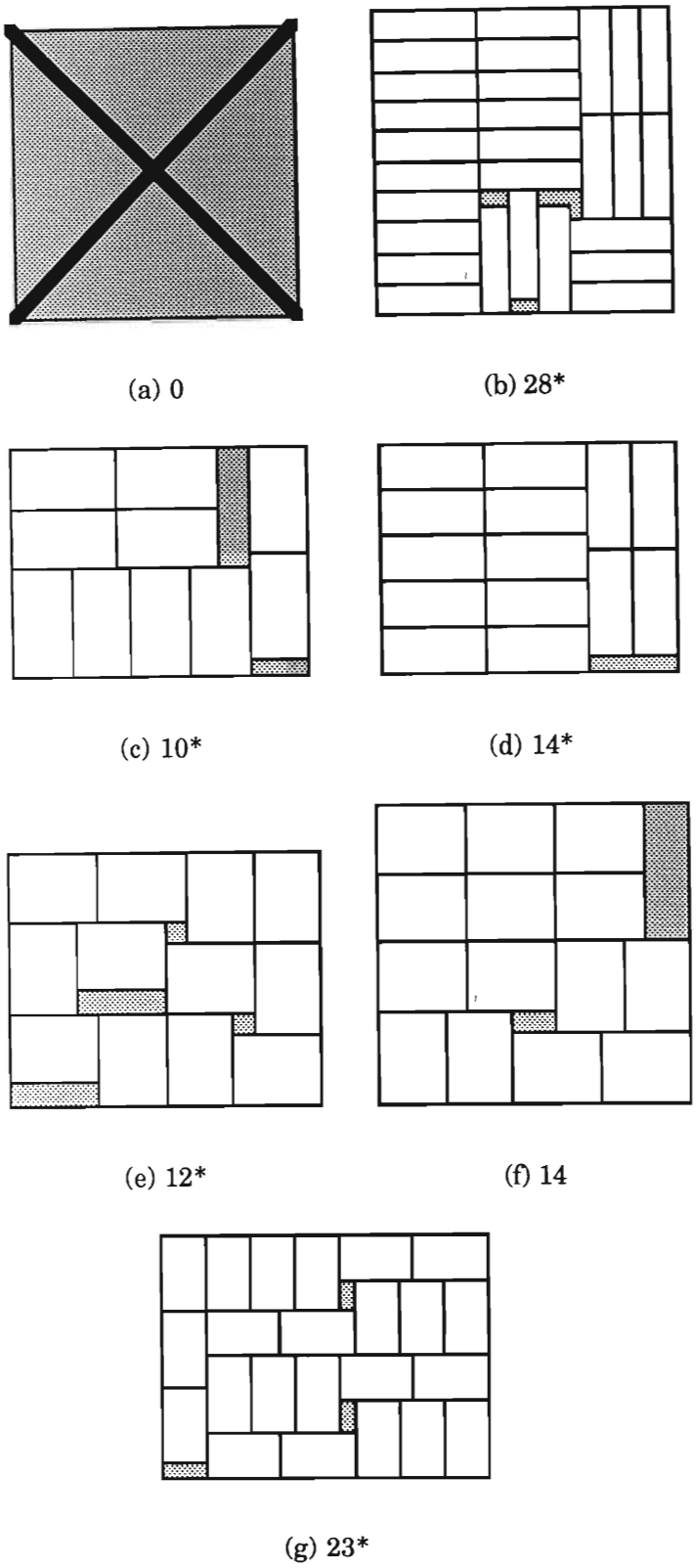
$$X = nc' + md'$$

and

$$Y = kc' + jd'$$

which fall into the required range. These can then be evaluated, with duplicates from the same equivalence class being removed, and a total of 8565 box and pallet combinations resulting.

Each of these can then be packed using the exact algorithm, and any that take more than the limit on the amount of time to produce a result can be stored to the database for later use.



(a) 0

(b) 28*

(c) 10*

(d) 14*

(e) 12*

(f) 14

(g) 23*

Figure 2.33: The optimal solutions to the packing examples as determined by Dowsland's combined algorithms. Note that the result was not determined for case (a).

Dowsland's final algorithm is applied to find the optimum packing of any box on a pallet where the box and pallet ratios satisfy

$$1 \leq \frac{a}{b} \leq 4$$

$$1 \leq \frac{X}{Y} \leq 2$$

$$1 \leq \frac{XY}{ab} \leq 51.$$

Since this does not cover a large portion of the test data used by Deighton, the algorithm was not tested with these examples. The example test from table 2.1 (a) was also not used, as the area ratio falls outside that required by Dowsland's algorithm. (This test resulted in a reduced graph containing 1034 nodes).

The test data derived from Dowsland's papers was, however, tested, and the the upper bound was attained in 98.03% of the cases. Figure 2.33 gives examples of layouts produced.

2.7 CONCLUDING REMARKS

This chapter has shown that it is feasible to attempt to solve a problem such as the pallet loading problem on a personal computer.

The problem that has been considered is restricted by the condition that only orthogonal $2\frac{1}{2}$-dimensional layouts of identical boxes are considered. For a range of commonly used box and pallet sizes, it is possible to calculate an exact solution using a combination of graph theory techniques and heuristics. Boxes must however sometimes be packed into large storage devices, such as shipping containers, and for

this case the exact method requires too much computation time. In preference, one of the heuristic measures that have been discussed should be applied.

When the restrictions to orthogonal, $2\frac{1}{2}$-dimensional layouts of the same sized boxes are lifted, no similarly elegant solutions are available, as will be shown in the next chapter.

It has also been pointed out [Car85] that there are cases for which the optimal layout, as defined above, is not the best layout to use. This occurs when there are other conditions that must be taken into account. Such conditions may include stability during transport, or the ability of the stack to be lifted by applying pressure to the sides.

> The pallet loading problem is generally tackled by attempting to maximise the number of rectangles that can be fitted orthogonally within a larger containing rectangle. These methods may produce solutions which do not satisfy such real-life problems as load stability or transportability. [Car85]

Applying these extra constraints, however, forces the problem away from the pallet loading problem to some other packing problem with its own particular restrictions, and so it must be tackled by a method that will employ the physical attributes of the problem to find a solution.

CHAPTER 3

OTHER CONTAINER PACKING PROBLEMS

The Pallet Loading Problem, discussed in chapter two, has been extensively studied over the last few years, and, as has been shown, a number of algorithms attempting a solution to the problem have been published. This, however, is not the case for the general packing problem. The various restrictions that characterise the pallet loading problem can either be tightened or relaxed, resulting in a new problem, which must have new methods developed in order to take advantage of the restrictions to provide a solution.

In this chapter various container packing problems are discussed together with their suggested solutions.

Initially some problems where the restriction to a single box dimension is retained are discussed. Two authors have considered ways of extending the pallet loading problem. Carpenter and Dowsland [Car85a] considered the pallet loading problem from a more practical viewpoint than the pure mathematical optimisation applied in chapter two. The restriction that the pallet load be stable under a number of conditions was added and given higher priority than maximum volume usage. In other words, if a more stable layout can be attained with slightly fewer boxes in it, then it is considered more feasible, as the risk of damage to the product during transportation and storage is reduced. Han, *et al* [Han86], on the other hand, attempted to improve the overall three-dimensional volume utilisation by relaxing the restriction to flat horizontal layers.

Other problems that have been tackled and are discussed in this chapter include the IPLS (Interactive Pallet Loading System)

formulation of Hodgson [Hod82] and Carlo *et al* [Car85]; the multiple box size algorithms of George and Robinson [Geo80] and Haessler and Talbot [Hae90]; and the use of robots to execute the physical packing as discussed by Malstrom, Meeks and Flemming [Mal86] and Penington and Tanchoco [Pen88].

3.1 A STABLE LAYOUT

Carpenter and Dowsland [Car85] point out that

> Distribution staff are not only concerned with maximising pallet utilisation; the stability of pallet stacks for transportation, both within and outside the warehouse environment, is a further important consideration.

As a result of this observation, a method is required that will pack a maximum number of boxes per pallet while satisfying such real-life constraints as load stability and transportability.

All the methods that have been discussed in the previous chapter attempt to find a layout that will result in a maximum volume utilisation, and place no emphasis on stability of the loads produced.

In order to add such constraints to the algorithm, it is necessary to evaluate individual requirements for each new box, depending on such attributes as box size, construction, rigidity and weight.

For the stability of the load to be ensured, there must be no vertical columns of boxes which are not interlocked with the rest of the pallet load, as well as no boxes that are not sufficiently supported from below. Walls of boxes that do not interact with the rest of the load should also

be avoided. These constraints imply that there must be an interaction between adjacent layers of the load.

Carpenter and Dowsland expressed these conditions by imposing the following three conditions on all the boxes in the load.

- Each box must have its base in contact with the pallet surface or with at least two boxes in the layer below. Contact of less than X% is ignored.
- Each box must have at least Y% of its base in contact with the layer below (or 100% in contact with the pallet surface).
- There must be no straight guillotine cuts traversing more than Z% of the maximum length or width of the load.

Sometimes pallets are not used as a base on which the loading takes place, but boxes are stacked in some formation, and transported in these stacks by 'clamp trucks', which clamp and lift the stack by applying pressure to opposite faces. In order for this to be possible, there must be interaction between boxes on the same layer of the stack.

Carpenter and Dowsland's test for this condition is that at least one pair of opposite sides must be flat, and that at least J% of the length of each box parallel to that pair of edges must be in contact with other boxes in that layer.

The relative degree of importance of these criteria will depend on the nature of the boxes being packed, and thus it will be necessary to calculate values for X, Y, Z and J for each packing situation for which a stable load is required. Typically, the values that are applied fall in the ranges

$5 \leq X \leq 15$;

$75 \leq Y \leq 95$;

$50 \leq Z \leq 100$; and

$40 \leq J \leq 100$.

Each of these values has an impact on the resulting stability of the load. For example, as the value of X increases, more loads will test unstable. (If $X > 50\%$ then no loads will ever test stable!)

Because of the advantages of simplicity, it is a usual requirement that the layout pattern used for the first and all odd numbered layers is a reflection or a rotation of that used for the second and all even numbered layers (Figure 3.1). Carpenter and Dowsland mention that no practical advantage was found by removing this restriction and using multiple layer formats.

In order to analyse the results produced it is first necessary to 'compress' the loaded boxes so as to have a minimum amount of unused space within the layout. This can be achieved easily by pushing all the boxes in the load as far to the top left corner of the pallet as possible. Stability and clampability of the load can then be calculated by checking the given conditions for each box in the load individually.

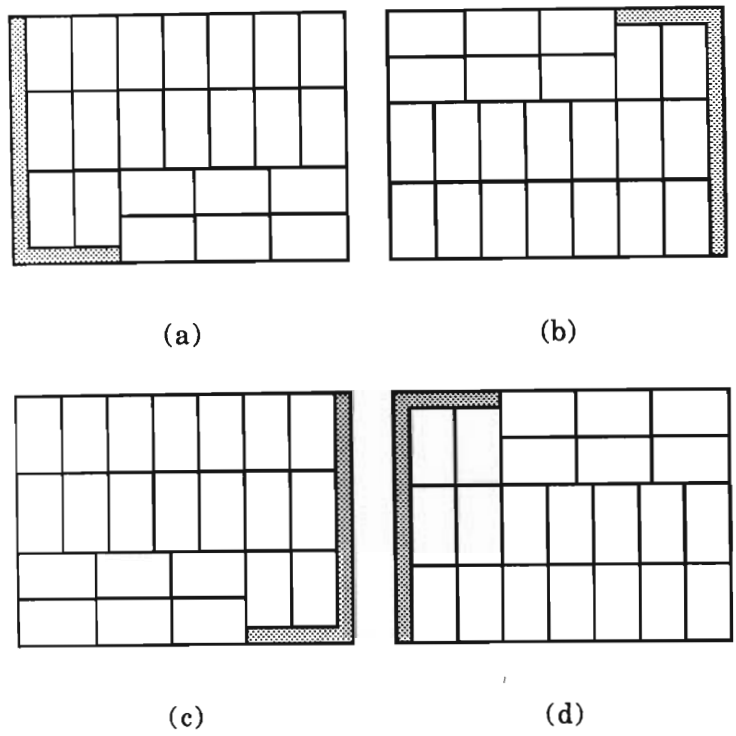


Figure 3.1: Possible layouts for alternate layers of a pallet load, showing (a) original layout, (b) 180° rotation, (c) reflection through short axis, and (d) reflection through long axis.

As a result of their processing, Carpenter and Dowsland show that a single-region packing configuration can never be stable, as it will necessarily consist of guillotine cuts. One and two region layouts have the greatest chance of being clampable, but are usually not stable. Three and four region layouts are generally the most stable. They also make the point that boxes with a square base can never produce a stable layout, as there can be no interlocking between columns unless some of the boxes overhang empty space.

The user thus has the ability to trade optimum volume utilisation off against load stability by modifying the percentage values specified. It is, however, common practise to stabilise a load by inserting packing material into it. For example, sheets of corrugated cardboard are occasionally inserted between layers of boxes to make the load more stable, and thus allow a pure mathematical optimisation to be applied.

3.2 THREE DIMENSIONAL OPTIMISATION

Han *et al* [Han86] attempted to extend the pallet loading problem in order to achieve an overall optimal three-dimensional layout. Their algorithm attempts to load a maximum number of identical small rectangular prisms (of size a×b×c) into a single larger one (X×Y×Z) by relaxing the pallet loading problem's restriction to flat horizontal layers.

Their algorithm uses a similar approach to the pallet loading algorithms of chapter two, but instead of producing a number of layers over the base of the pallet and selecting the best combination of these layers, it repeatidly packs the base of the pallet and one of the two possible vertical faces.

The packing in the L-shaped region formed by the base and vertical side being considered optimises only over the current region and, once established, is not modified later to produce a better overall layout, that is, no backtracking is catered for.

The method used in the optimisation of this L-shaped region at each stage of the algorithm is to pack the base with boxes in each of the three possible vertical orientations and then to select the orientation that gives the largest percentage area cover. The same procedure is then applied to the vertical faces, but instead of using a fixed vertical orientation, the orientation of the boxes with respect to the vertical face of the pallet is held fixed. Once again, the face layout with the largest percentage cover overall is applied. This ensures that once the packing has taken place, it results in a flat surface and a flat wall on and against which the next layer can be packed (figure 3.2).

The pallet size is then reduced by ignoring the dimensions of the L-shaped region that has been packed, and repeating the process with the remainder of the pallet until no more boxes can be packed.

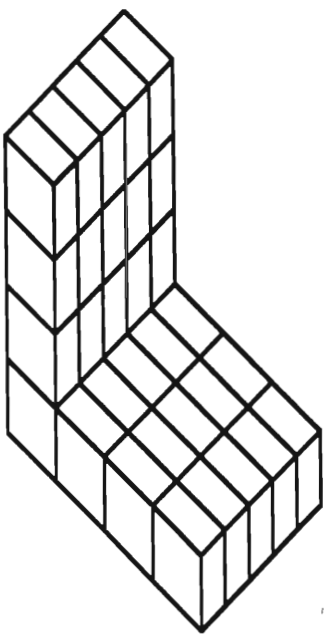


Figure 3.2: An optimised packing layout over the L-shaped region formed by the base and end of the pallet volume.

The algorithm results in a layout that can be separated into a number of optimum layers through the application of only guillotine cuts. Each of these layers is optimum from the viewpoint that it uses a maximum percentage of the volume it occupies.

The original method as described by Han required the application of Steudel's heuristic [Ste79] for the placing of the boxes within the layers. As it has been shown (see chapter two) that this is not an optimal method in a large number of cases, the algorithm was modified to use the Smith and DeCani heuristic [Smi80] for this optimisation.

In order to test this algorithm, 10000 randomly generated box and pallet combinations were generated in the range

$1000 \leq X \leq 1999,$

$1000 \leq Y \leq 1999,$

$1000 \leq Z \leq 1999,$

$100 \leq a \leq 499,$

$100 \leq b \leq 399,$

$100 \leq c \leq 299.$

These were packed using Han's algorithm, as well as the strict pallet loading problem solution of Smith and DeCani [Smi80]. Of the 10000 cases, Han's algorithm improved on the strict layering solution in 948 (9.48%) cases, and gave the same results in a further 1248 (12.48%). In the other 7804 cases (78.04%), using Smith and DeCani's algorithm with strict layers produced an overall layout containing more boxes.

A further test was then carried out. Here, the Smith and DeCani algorithm was run three times, once optimising over the base of the pallet and extending this to the height, once optimising over the end of the pallet and extending this down the length, and finally optimising over the side and extending the results across the width of the pallet. The best result obtained from these three runs was tested against those of Han's algorithm. The results of this testing showed that Han's algorithm gave better results than the strict layers approach in only 151 cases (1.51%), and fewer boxes per pallet in 9006 (90.06%) of the tests.

Apart from producing results which provide only questionable advantages, Han's algorithm has a number of other drawbacks. Firstly, it requires more execution time than a direct pallet loading approach, as the optimisation over the layers needs to be carried out at each stage of the algorithm with the reducing pallet dimensions. This

obviously requires more applications of whichever two-dimensional optimisation algorithm is being used.

Secondly, the algorithm is only applicable to situations where the boxes can assume any vertical orientation. If the boxes have to remain upright when packed, then the end and side optimisations will be of no value.

A third drawback of the algorithm is that the suggested layouts are necessarily more complex than those produced by the algorithms of chapter two. This poses two problems: they are harder to depict using a computer (figure 3.3); and the physical packing is more difficult to execute manually. The outcome of this is that the cost of packing the pallet will increase, reducing any savings which would be made by shipping possibly one extra box per pallet.

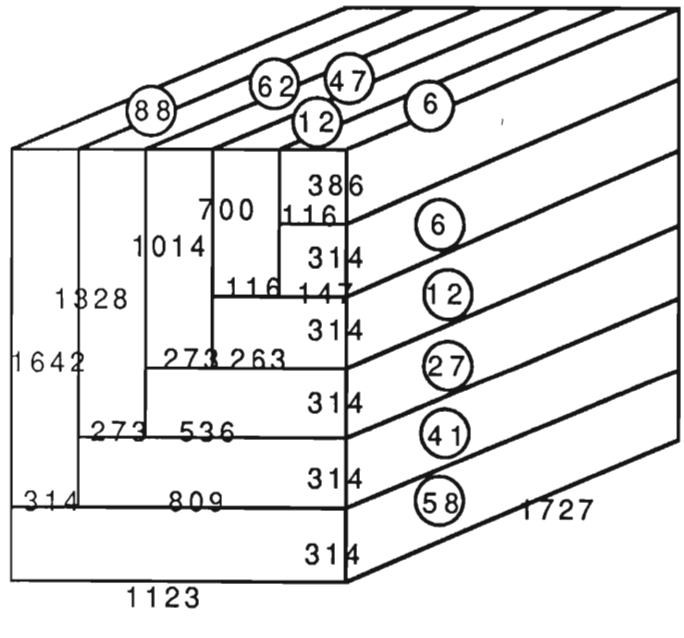


Figure 3.3: A diagram representing the packing of 314×273×116mm boxes onto a 1727×1123×1956mm pallet using Han's algorithm. (Ringed numbers indicate the number of boxes packed within the corresponding slice. Other numbers are slice dimensions)

A final drawback of the algorithm is that, when analysed by the stability constraints of Carpenter and Dowsland [Car85a], the layouts produced are always unstable because of the walls built in the vertical plane.

As one final comment on the algorithm, it should be pointed out that the authors give a detailed analysis for packing a 48×24×40 inch pallet with 11×6×6 inch boxes. This analysis results in 195 boxes per pallet. The Smith and DeCani algorithm [Smi80] used directly with the same data, packs 196 boxes.

3.3 MULTIPLE BOX SIZES

Two separate problems can be addressed under this heading; one being the case where all boxes have their own unique dimensions, and the other attempting to pack multiple instances of each of a number of different box types. In either case, whatever the problem to be solved, the dilemma of George and Robinson may be encountered:

> We have found no previous papers dealing with this precise problem. [Geo80]

and thus a new heuristic will have to be developed.

In this section, some of the problems that have appeared in the literature are reviewed, and their solutions presented.

3.3.1 An Interactive Solution

Hodgson described an interactive method for packing multiple sized boxes onto a pallet [Hod82]. The system he described is called Interactive Pallet Loading System (IPLS) and was developed to assist the United States Air Force (USAF) with the transportation of military supplies.

Because of the nature of the expected loads, Hodgson required that his algorithm be interactive, and provide a packing layout that could be modified by the packers if necessary. Load stability is not an important aspect in this application, as the loads are stabalised by covering them with a nylon net that is pulled tight, holding the boxes in place.

The approach taken is a two-phase method. In the first step, the user guides the program by selecting a set of 'Base Boxes' onto which vertical columns of boxes are stacked so as to maximise use of the resulting volume. This volume is defined by the base dimensions of the base box and the maximum allowable height of the pallet load. Boxes are stacked strictly one on top of another into this area, with none protruding out of the column dimensions. This step can easily and without too much computation time, be solved by a complete search. The user is given the opportunity to modify these columns if desired.

In the second step of the solution procedure, the program places the columns of boxes produced by the first stage onto the floor of the pallet. Since the columns are set up optimally in the first step, the quality of the results produced depends only on the percentage surface area covered by the base boxes.

Hodgson's method of placing the columns onto the pallet results from a simplification of an exact dynamic programming technique.

This optimal technique depends on defining a partition of the pallet floor such as that of figure 3.4. At any stage of the loading algorithm, the left-hand subpallet is completely loaded, and the right-hand subpallet is empty. The partition can be extended by adding one of the remaining boxes to the load, thereby increasing the number of boxes packed. Since any of the unpacked boxes could be added to any point in the partition at every step in the algorithm, and either of the two possible orientations would need to be considered, this optimal technique becomes infeasible, as too much time would be needed to consider all the options.

> ... the computer time required to solve the dynamic program likely would be well beyond any sensible limit for real-world applications. [Hod82]

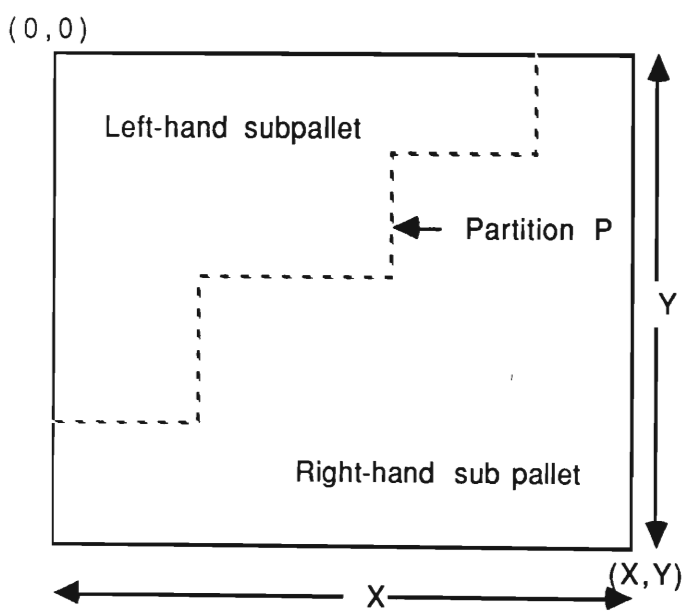


Figure 3.4: A partition of the pallet floor.

Because the optimal approach is not computationally feasible, Hodgson defined a simplification of the method which reduced the amount of computer time required and made the solutions attained more applicable to the physical problem.

A rectangular partition of the pallet floor is defined as a partition which has a profile in the form of a single rectangle (figure 3.5).

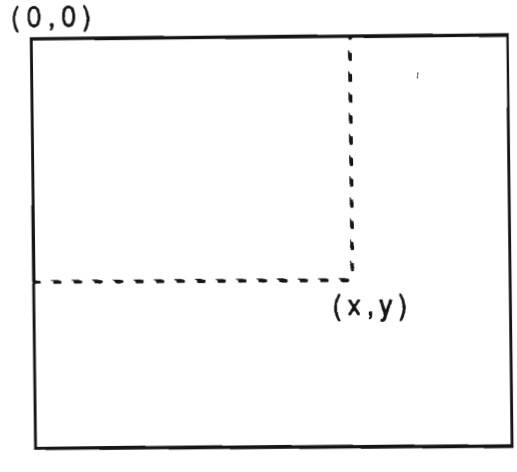


Figure 3.5: A rectangular partition of the pallet floor.

By restricting the partitions considered to rectangular ones only, Hodgson was able to redefine the dynamic programming phase of the algorithm in such a way that a feasible amount of computer time is used. This dynamic program extends the current rectangular partition to form a new one by adding boxes along both sides of the partition. (Figure 3.6)

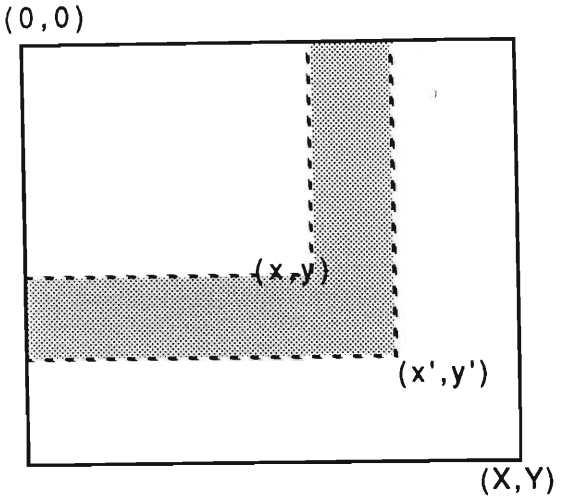


Figure 3.6: Extending the rectangular partition (x,y) to a new rectangular partition (x',y').

The boxes are packed into this new L-shaped area by dividing it into two regions as shown in figure 3.7, and optimally packing boxes into these two regions.

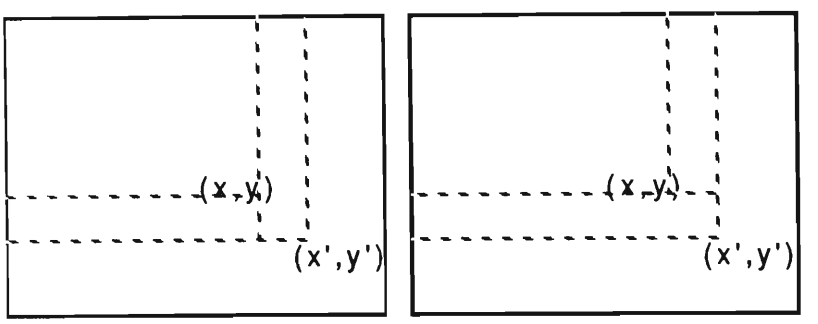


Figure 3.7: The two decompositions of the L-shaped region into rectangular sub-regions.

The layouts thus produced by Hodgson's algorithm are separable by a number of guillotine cuts (figure 3.8). This, although not guaranteeing an optimal layout, does give a number of advantages. It was mentioned that a set of guidelines were required for the packers, which could be modified when the physical packing took place. The restriction to

guillotine cuts implies that the packers can move boxes within the rectangular regions in which they are packed, thus keeping volatile liquids and explosives at the edge of the load as required by the USAF regulations, and also giving some control over the centre of gravity of the loaded pallet.

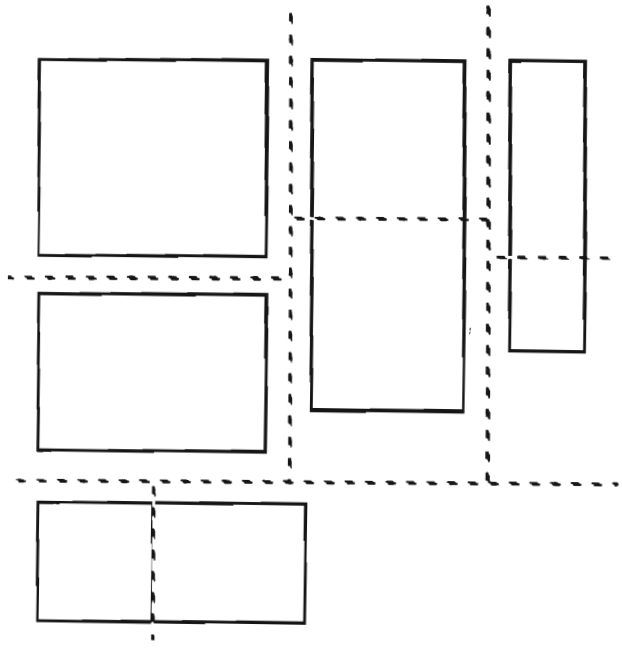


Figure 3.8: The guillotine cuts (dotted lines) used to decompose the layout formed by Hogdson's algorithm. Figure 3.9 shows the corresponding layout.

The procedure of packing boxes is further simplified by adding the restriction that the box with the largest base dimensions must always be packed first. This restriction, introduced by Hodgson, results from observing the physical packing process, and is often necessary for load stability.

As an example of typical results that are produced by this algorithm, consider packing a square pallet, with sides of length 60 inches, with boxes as described in table 3.1. (The example is from [Car85].) The final layout produced by Hodgson's algorithm is shown in figure 3.9.

Box No	Length	Width
1	30	25
2	30	20
3	23	22
4	25	20
5	20	20
6	20	15
7	25	10
8	15	15
9	12	10

Table 3.1: The boxes used for demonstrating Carlo's algorithm.

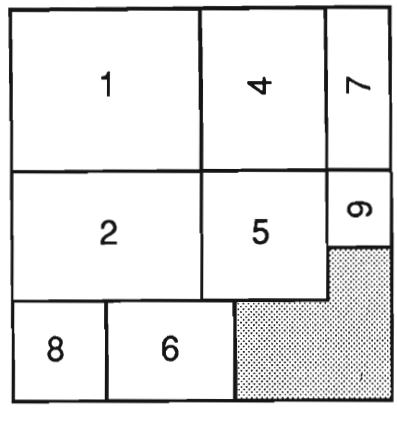


Figure 3.9: The result of applying Hodgson's algorithm to Carlo's example set.

Carlo *et al* [Car85] adapted Hodgson's algorithm so that it could be run on a micro-computer. The reasons for this were that the micro allowed greater user interface capabilities for training staff, as well as the capability of producing graphical output which would help the packers perform the physical packing of the pallet.

Carlo retained the same interactive column generating phase, but defined a new base loading algorithm.

> Since the stacking part of the procedure is optimal, the key to the quality of the solutions generated ... for the three-dimensional problem lies in the dynamic programming base loading heuristic. [Car85]

For Carlo's application, Hodgson's base loading procedure could not be applied, as it required too much computer storage space and time for implementation on a PC. A new adapted version of the algorithm was therefore developed.

This new algorithm relies on a random ordering of the boxes eventually to provide a satisfactory solution. The steps in the algorithm will be demonstrated with reference to the data used to demonstrate Hodgson's method.

The algorithm begins by sorting the boxes in order of decreasing base area, and orienting them so that their longer base dimensions are parallel to the width of the pallet. (This is implicit in the ordering of the boxes in table 3.1.) The first box in the list is then positioned in the top-left corner of the pallet. Subsequent boxes are packed forming a column down the pallet length, not exceeding the width of the first box in the column, until no more boxes will fit below those already packed in the column (figure 3.10).

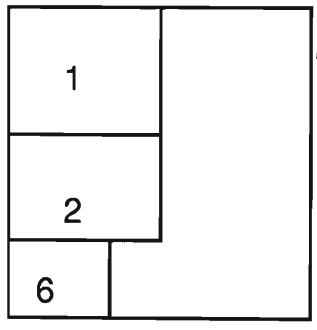


Figure 3.10: The first stage in Carlo's base loading algorithm.

The section of the pallet defined by the length of the pallet and the width of the first box packed is then disregarded, and the algorithm begins again with the reduced pallet and the remaining boxes (figure 3.11).

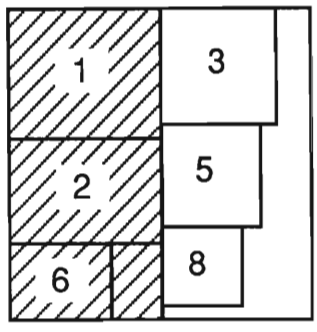


Figure 3.11: The second column of boxes is packed into the unshaded region of the pallet.

The algorithm finally terminates when there are no more boxes to pack, or none of the remaining boxes will fit onto the remaining subpallet. In figure 3.12, the remaining subpallet is only 7 units wide, whereas the minimum box dimension is 10 units, and so no more boxes can be packed.

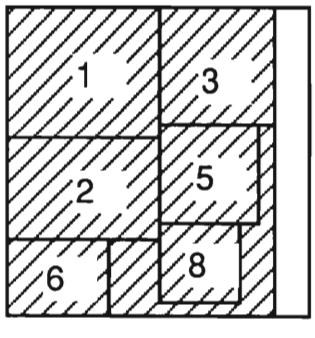


Figure 3.12: No more boxes will fit onto the unshaded subpallet.

The initial layout is defined to be the seed layout, and new layouts are generated by randomly ordering and orienting the boxes, and then following the same steps with the modified box list. As soon as a layout

with a greater area covering is generated, it becomes the new seed layout, and the process continues.

When a user specified amount of time has elapsed, the algorithm terminates, returning the current seed as the layout to use. Figure 3.13 shows the best result produced within one hour of processing on a IBM compatible PC.

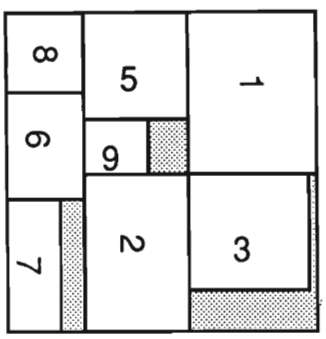


Figure 3.13: The best layout found within an hour of processing with Carlo's algorithm.

This algorithm has an advantage that the user can decide on a trade-off between processing time spent and results obtained. Because the layouts produced can be decomposed by a set of guillotine cuts vertically down the pallet, they allow the packers to move explosives and volatile liquids to the edges of the pallets by changing the order of the boxes in the columns or by re-ordering the columns. This also allows some control over the centre of gravity of the final load.

An extension to the algorithm as specified by Carlo is to allow the boxes to be rotated if they will not fit into the available space. This results in the same solutions being found in less computation time.

Another extension that has been developed is to remove the restriction that the boxes be able to be reordered in the final packing, that is, remove the restriction to guillotine cuts. This is achieved by pushing boxes into the gaps left along the right hand edge of the previous layout, and allowing columns to start at positions other than the top of the pallet once no more columns will fit there. This allows boxes to be placed in regions that in the original algorithm would be left as unusable.

For example, the first seed layout shown in figure 3.12 is adapted to appear as in figure 3.14. Note that box number 7 has been included to start a new column part of the way down the side of the pallet as none of the boxes could be used to start a column at the top of the pallet.

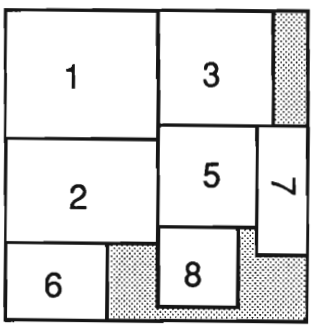


Figure 3.14: The initial seed layout produced by the extended algorithm.

The best layout produced by this algorithm for the same boxes in the same amount of time as figure 3.13 is shown in figure 3.15. This results in 95.2% surface cover as opposed to 87.5%, but at the cost of a rigid layout that cannot be modified by the packers at packing time. Hodgson's original algorithm generated 87.36% volume usage.

This demonstrates that restricting the layouts to those that can be formed by a sequence of guillotine cuts may well adversely affect the

optimality of the volume usage. This point has been noted by a number of authors [Ste79, Bis82, Dec79].

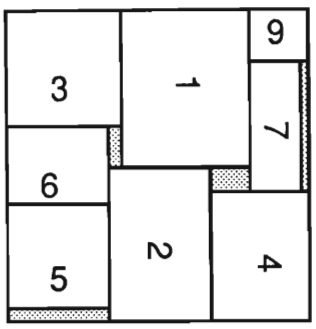


Figure 3.15: The best layout produced by the new algorithm.

3.3.2 Multiple Instances Of The Same Box Type

George and Robinson [Geo80] produced a method of solving a particular packing problem. Their problem involves finding an order of packing and producing a list of instructions for positioning the boxes so that an entire given load can be stacked inside a container. The loads considered typically consist of up to 20 different types of boxes. The boxes can all be rotated in any orientation and none is restricted as to the number of boxes that can be stacked on top of it.

The proposed method fills the container one layer at a time, each layer being a slice of the length of the container occupying the whole width and height. The length usage is defined by the dimensions of the first box packed.

The boxes are first prioritised according to their dimensions and number of boxes of the same type. This priority is used in order to ensure that similar boxes are packed in proximity to each other.

A box type is given 'open' status once some but not all of the boxes of that type have been packed into the container. The first box packed into each layer is selected as the open box with the greatest number remaining, or, if no boxes are of open status, the box with the highest priority is used.

If the first box type packed into the layer does not completely fill the layer, either because there are not sufficient of the boxes, or because there are spaces along the edges of the layer which although too small for that box may contain a box of a different type, then the left over space is filled with boxes of a different type.

These other box types are chosen so as to make maximum use of the space left around the layer, as well as any space left in front of the previous layer. Once these boxes have been selected they are packed into the available space. In order to facilitate this process, the layers are always formed in such a way that their front surface takes the shape of a step function, that is, there are no boxes that jut out from the surface (figure 3.16).

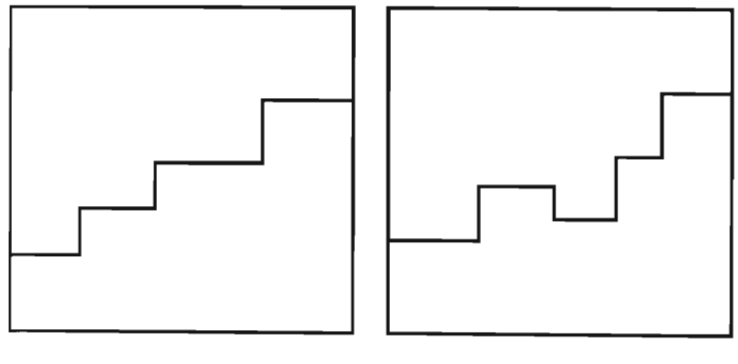


Figure 3.16: A layer surface showing a step function (a) and non-step function (b).

Using their method, George and Robinson succeeded in producing packing layouts for the examples originally commissioned, as well as for a number of theoretical cases. One of the authors calculated solutions by hand, while the other worked on the computer solution. All the problems were solved by hand before the computer solution was completed, however, the exercise proved useful from the point that several ideas that were developed while the hand calculations were being carried out could be included in the computer program, which then is in the position that it can apply these ideas to other examples more quickly than by using hand calculations.

3.3.3 Packing Pallets

The objective of Haessler and Talbot's research [Hae90] was to develop a fast computer program that can be used at order entry time to size an order and produce a layout for that order so as to result in increased vehicle utilisation and reduced freight cost.

The background for this is that a supply company sends stock to retailers in either railcar or truck trailer loads. When a retailer sends in an order for new stock, the order can be adjusted up or down slightly, in order to use the available transport methods more efficiently. The computer program can be applied so that the retailer can immediately, while putting in the order, be informed of these adjustments to the original order, so that they can be approved or rejected early.

As the research was carried out for a particular wholesale distributor, that distributor's precise problem had to be solved. A set number of stock items are available. These are packed into corrugated cardboard cartons and stacked on pallets to form unit loads. Each item thus has fixed dimension in its layers (37×44 inches to 43×52 inches). The

required mix of stock is then to be packed into a railway truck or trailer for transportation. These trucks and trailers also have fixed dimensions. The railway trucks are assumed to have dimensions 730×108×130 inches, and the trailers 650×95×109 inches.

Although recommended, it is not essential that all orders are in multiples of unit loads. Tier (layer) quantities or even single boxes can be ordered. From a packing point of view, since the products are stored in unit loads, it is easier, cheaper, and quicker to pack entire units rather than loose boxes although the volume utilisation may be influenced adversely.

Usually, full unit loads are packed, wherever possible, with layers of boxes being stacked on top of them to fill any available height. Loose boxes are finally packed in by hand to fill any voids in the layout. It is also common to try to restrict these voids to places near the doors of the trailers and trucks to facilitate easy loading.

The overall procedure that was adopted by Haessler and Talbot is firstly to estimate the number of stacks that will fit into the required vehicle. The order is then adjusted to allow the creation of these stacks. An attempt is then made to place the stacks within the vehicle. If the stacks cannot all be fitted, the number of stacks required is reduced by one, the necessary adjustments made to the order, and the fitting process repeated. If all the stacks can be positioned, the number of stacks is incremented, updating the order quantities, and again the fitting process starts again. This is repeated until the optimal number of stacks for the vehicle is found. The adjusted figures on the amount of each stock item are then put forward for confirmation.

The stacks are fitted into the vehicles using two separate algorithms, one for railway trucks, and a different one for trailers. Because the railway trucks are sufficiently large, there is necessarily enough space

for two stacks next to each other across the width. (Maximum width of two unit loads is 104 inches against the width of the trucks which is 108 inches.) Trailers, on the other hand are slightly narrower (95 inches), and thus may not be wide enough to accommodate two stacks in either orientation. Three layouts are therefore considered for trailers (figure 3.17), depending on the dimensions of the actual pallets.

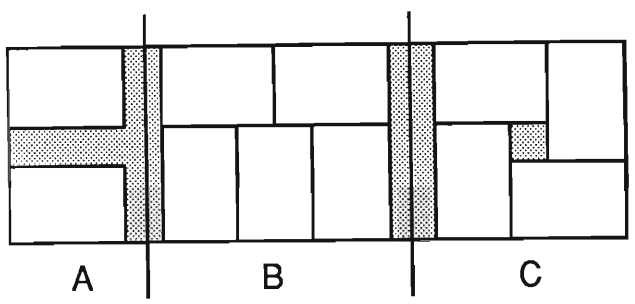


Figure 3.17: The three pallet layout forms used for loading a trailer with boxes.

Haessler and Talbot report considerable savings through using this computer program, as well as increased customer satisfaction in that the sizing of the orders can be calculated as the order is entered, and they can either agree to any changes or request a different order immediately.

3.4 ROBOTIC PALLETISERS

Malstrom, Meeks and Flemming [Mal86] and Pennington and Tanchoco [Pen88] have investigated the use of robots to carry out the physical palletising of boxes. Typically this could be effective in a medium speed environment where boxes arrive for packing in no pre-defined order. Although both authors consider the case where the boxes are not all of the same size, they include a common restriction that the boxes all have an identical height dimension in order that flat layers may be produced on which the next layers can be packed.

Malstrom, Meeks and Flemming developed a linear programming formulation for solving their loading problem. This method requires that 100% utilisation of the pallet deckboard be attained. Since such optimal usage is not always possible, they make use of a dummy box, having surface dimensions of 1×1 units. An upper bound on the number of these dummy boxes that can be used is found by evaluating the basic layout of section 2.2 for each of the box types individually on the pallet, and setting the upper bound equal to the minimum wasted area of the resultant layouts.

The next step in the method is to define a number of 'strips'. These represent all the possible combinations of the set of unique box dimensions that fully utilise the length of the pallet. The formation of these strips is a simple 1-dimensional problem, and the strips thus produced are assumed to be 1 unit wide. These can be represented in a two-dimensional array, A, where

$a_{i,j}$ = the number of times box dimension i is used in strip j.

i=1..number of unique dimensions,

j = 1.. number of strips.

As an example of such an array, the values shown in table 3.2 correspond to the set of strips possible for packing boxes of size 1×2, 2×2, and 3×2 onto a 5×4 pallet. The corresponding strips are shown in figure 3.19.

i	Length	j				
		1	2	3	4	5
1	1	5	3	1	2	0
2	2	0	1	2	0	1
3	3	0	0	0	1	1

Table 3.2: The matrix entries corresponding to the strips formed by box dimensions 1, 2 and 3 over a pallet length of 5.

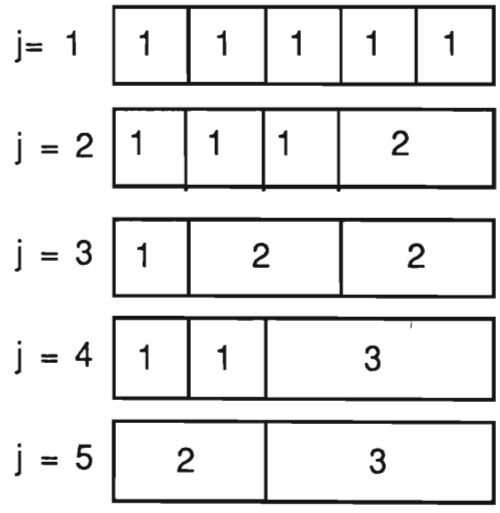


Figure 3.18: The 5 strips corresponding to the values shown in table 3.2.

The linear programming formulation now requires that the function

$$z = \sum_{s=1}^{r} [l_s w_s x_s + l'_s w'_s x'_s]$$

be maximised, subject to the constraints

$$Y = \sum_{j=1}^{m} y_j;$$

$$\sum_{j=1}^{m} a_{ij}y_j = \sum_{s=1}^{r+1}[q_{is}w_s x_s + q'_{is}w'_s x'_s], \text{ for } i=1..k; \text{ and}$$

$$X_D \leq \text{Calculated Bound}$$

where

z = total area used

m = number of strips

r = number of distinct dimensions

l_s = length of box s

w_s = width of box s

x_s = number of boxes of type s in layout

y_j = number of times strip j is used in final layout

$$q_{is} = \begin{cases} 1 \text{ if box type s has dimension i} \\ 0 \text{ else} \end{cases}$$

X_D = number of dummy boxes;

and any symbol followed by a prime (') represents a 90° rotation of the box.

Using this linear programming formulation, the example data from figure 3.18 and table 3.2 give the results of figure 3.9.

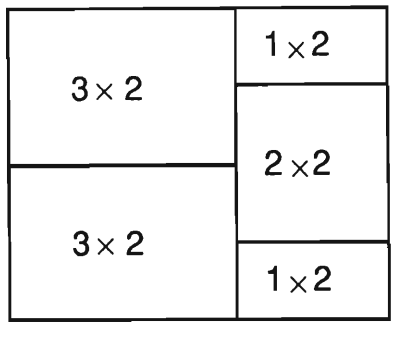


Figure 3.19: The resultant layout produced from the strips of figure 3.18.

Pennington and Tanchoco, on the other hand, pointed out that such a pure arithmetic optimisation may not be feasible because it does not take into account the possibility of interference between the robot's grippers and the boxes that are already packed on the pallet. They then developed an algorithm that takes into account the physical attributes of the robot.

> The algorithm must consider box placements on the pallet so the gripper cannot interfere with the boxes already packed on the pallet. [Pen88]

In order to accomplish this, and to take into account the fact that the robot gripper they were using is approximately one third of the width of their pallet, they developed a method that fills the pallet in three sections, as shown in figure 3.20.

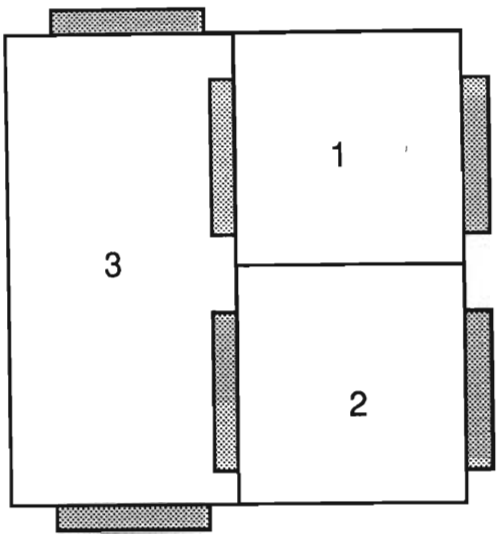


Figure 3.20: The three regions of Pennington's algorithm. The positions of the grippers relative to each region are shaded. The numbers show the order in which the sections are filled in order to avoid interference.

Each of the three regions is filled with single boxes in a single orientation, (c.f. Basic Method of section 2.2) with the largest available box type being considered first.

Both the Malstrom, Meeks and Fleming paper [Mal86] and that of Pennington and Tanchoco [Pen88] result from tests carried out with scale models of robots, pallets and boxes, although the results are just as applicable to the full sized problem and a larger robot.

The papers are also both limited from the point of view that all the boxes being packed must have a common vertical dimension. This restriction will need to be relaxed before it can be applied to any physical packing problems.

3.5 CONCLUDING COMMENTS

The general container packing problem cannot be tackled directly, as any procedure that will work well for one particular sub-problem will inevitably be useless when applied to a similar problem with slightly different restrictions.

As a result of this observation, it has been noted that for any packing problem that is encountered, problem specific heuristics must be developed in order to find the best solution to that exact problem.

CHAPTER 4

CONCLUSIONS

The general packing problem occurs in a variety of different forms, and has numerous industrial and commercial applications.

The problem is NP-complete, and, as a consequence, most published research has been directed at finding good approximations to the optimal solution in reasonable amounts of computing time, rather than attempting to solve the problem exactly. Various heuristic measures have been developed, and each of these uses the physical attributes of the particular problem it addresses, to refine its solutions. As a result of this, the methods which have been developed are problem specific and no general method that will efficiently solve all packing problems has been found.

> A general method is like a size 48 cloth, it can cover everybody but does not fit very well. [Dow85a]

This work has addressed the packing problem from the point of view of packing boxes into a shipping container or onto a pallet. This problem, known in operations research literature as the container packing problem, is commonly encountered in situations where bulk quantities of goods need to be transported or stored. In particular, the pallet loading problem in which a maximum number of identical rectangular cartons are to be packed onto a pallet in flat horozontal layers, has received much attention.

Since it is desirable for a number of reasons to solve the problem, and no easy solution is likely due to the NP-completeness of the problem, it is reasonable to apply heuristics that use the speed and processing power of modern computers to determining a solution. Because of the wide availability of cheap, yet powerful micro-computers and PC's, and the user-friendly graphic based interface of which they are capable, it was decided to investigate methods that could solve the packing problems within reasonable time constraints using these computers.

A search of the available literature resulted in a number of methods for solving the packing problems being found. For the pallet loading problem, three general approaches have emerged. The first of these requires that an optimal placement of boxes along the perimeter of the pallet be found and extended into the pallet centre, in order to maximise the total overall number of boxes packed. The second approach involves selecting a pre-defined layout pattern, and examining all combinations that conform to that general pattern, in order to determine the best combination to use. The third approach uses a combination of graph theory techniques and heuristics to find a solution. Although this last approach finds an exact solution, it requires far more computation time than the other two methods, and in a number of cases requires mainframe computer power, or is completely infeasible.

By including the upper bound on the number of boxes that can be packed, the amount of computation time required to reach a solution can be reduced. This has been employed by Dowsland in a number of ways in order to produce an exact method that will solve most problems in a given range within reasonable time constraints.

For the general container packing problem no such neat arrangement of ideas exists, as each author has addressed a particular problem, and the approaches that will solve one problem efficiently may be completely useless when applied to a different yet similar problem.

REFERENCES

Bak80: B.S. Baker, E.G. Coffman & R.L Rivest (1980) ORTHOGONAL PACKINGS IN TWO DIMENSIONS. *Siam Journal on Computing* 9 (4) pp 846-855

Bak81: B.S. Baker, D.J. Brown & H.P. Katseff (1981) A 5/4 ALGORITHM FOR TWO-DIMENSIONAL PACKING. *Journal of Algorithms* 2 pp 348-368

Bar79: F.W. Barnes (1979) PACKING THE MAXIMUM NUMBER OF M×N TILES IN A LARGE P×Q RECTANGLE. *Discrete Mathematics* 26, pp 93-100.

Bea85: J.E. Beasley (1985) BOUNDS FOR TWO DIMENSIONAL CUTTING. *Journal of the Operational Research Society* 36 (1) pp 71-74

Bea85a: J.E. Beasley (1985) ALGORITHMS FOR UNCONSTRAINED TWO-DIMENSIONAL GUILLOTINE CUTTING. *Journal of the Operational Research Society* 36 (4) pp 297-306

Bis82: E. Bischoff & W.B. Dowsland (1982) AN APPLICATION OF THE MICRO TO PRODUCT DESIGN AND DISTRIBUTION. *Journal of the Operational Research Society* 33, pp 271-280.

Bro71: A.R. Brown (1971) OPTIMUM PACKING AND DEPLETION *Macdonald/American Elsevier Computer Monographs*

Car85: H. Carlo, T.J. Hodgson, L.A. Martin-Vega & E.R. Stern (1985) MICRO-IPLS: PALLET LOADING ON A MICROCOMPUTER. *Computing And Industrial Engineering* 9 (1) pp 29-34.

Car85a: H. Carpenter & W.B. Dowsland (1985) PRACTICAL CONSIDERATIONS OF THE PALLET LOADING PROBLEM. *Journal of the Operational Research Society* 36 (6) pp 489-497

Cha87: F. Chauncy, R. Loulou, S. Sadones & F. Soumis (1987) A TWO PHASE HEURISTIC FOR STRIP PACKING: ALGORITHM AND PROBABILISTIC ANALYSIS. *Operations research letters* 6 (1) pp 25-33

Cof78: E.G. Coffman, J. Y-T. Leung & D.W. Ting (1978) BIN PACKING: MAXIMISING THE NUMBER OF PIECES PACKED. *Acta Informatica* 9 pp 263-271

Csi89: J. Csirik (1989) AN ON-LINE ALGORITHM FOR VARIABLE-SIZED BIN PACKING. *Acta Informatica* 26 pp 697-709

Dag90: C.H. Dağli (1990) KNOWLEDGE-BASED SYSTEMS FOR CUTTING STOCK PROBLEMS. *European Journal of Operations Research* 44 pp 160-166

Dec78: P.DeCani (1978) A NOTE ON THE TWO-DIMENSIONAL RECTANGULAR CUTTING STOCK PROBLEM. *Journal of the Operational Research Society* 29 pp703-706

Dei89: A.G. Deighton (1989) AN INVESTIGATION INTO OPTIMUM CONTAINER PACKING. Unpublished Honours thesis University of Natal, Durban.

Dei91: A.G. Deighton, J.J. Meyerowitz (1991) AN IMPROVED ALGORITHM FOR PALLET LOADING submitted to Journal of the Operational Research Society.

Dow84: K.A. Dowsland (1984) THE THREE DIMENSIONAL PALLET CHART: AN ANALYSIS OF THE FACTORS AFFECTING THE SET OF FEASIBLE LAYOUTS FOR A CLASS OF TWO DIMENSIONAL PACKING PROBLEMS. *Journal of the Operational Research Society*. 35 (10) pp 895-905.

Dow84a: W.B. Dowsland (1984) THE COMPUTER AS AN AID TO PHYSICAL DISTRIBUTION MANAGEMENT. *European Journal of Operational Research* 15 pp 160-168

Dow85: K.A. Dowsland (1985) DETERMINING AN UPPER BOUND FOR A CLASS OF RECTANGULAR PACKING PROBLEMS. *Computing And Operations Research* 12 (2) pp 201-205.

Dow85a: K.A. Dowsland (1985) A GRAPH-THEORETIC APPROACH TO THE PALLET LOADING PROBLEM. *New Zealand Operational Research* 13 (2) pp 77-86

Dow87: K.A. Dowsland (1987) AN EXACT ALGORITHM FOR THE PALLET LOADING PROBLEM. *European Journal of Operational Research* 31 pp 78-84.

Dow87a: K.A. Dowsland (1987) A COMBINED DATA-BASE AND ALGORITHMIC APPROACH TO THE PALLET-LOADING PROBLEM. *Journal of the Operational Research Society* 38 (4) pp 341-345

Eil71: S. Eilon & N. Christofides (1971) THE LOADING PROBLEM. *Management Science* 17 (5) pp 259-268

Fis79: J.C. Fisck & M.S. Hung (1979) A HEURISTIC FOR SOLVING LARGE LOADING PROBLEMS. *Naval Research Logistics Quarterly* 26 (4) pp 643-650

Geo80: J.A. George & D.F. Robinson (1980) A HEURISTIC FOR PACKING BOXES INTO A CONTAINER. *Computing and Operations Research* 7 pp 147-156

Gil64: P.C. Gilmore & R.E. Gomroy (1964) MULTISTAGE CUTTING STOCK PROBLEMS OF TWO AND MORE DIMENSIONS. *Journal of the Operational Research Society* 13 pp 94-120

Hae90: R.W. Haessler & F.B. Talbot (1990) LOAD PLANNING FOR SHIPMENTS OF LOW DENSITY PRODUCTS. *European Journal of Operational Research* 44 pp 289-299

Han86: C.P. Han, K. Knott & P.J. Egbelu (1986) A HEURISTIC APPROACH TO THE THREE DIMENSIONAL CARGO LOADING PROBLEM *Proceedings of the eighth annual conference on computers and industrial engineering* 109-113

Hin80: A.I. Hinxman (1980) THE TRIM-LOSS AND ASSORTMENT PROBLEMS: A SURVEY. *European Journal of Operational Research* 5 pp 8-18.

Hod82: T.J. Hodgson (1982) A COMBINED APPROACH TO THE PALLET LOADING PROBLEM. *IIE Trans (USA)* 14 (3) pp 175-182.

Joe87: C. Jones & F. Whatron (1987) CUTTING-PATTERN ENUMERATION ON A MICROCOMPUTER: A CASE STUDY. *Advances in Manafacturing Technology. Proceedings of the First National Conference on Production Research, England* pp 51-56

Lou82: E. Loukakis, C. Tsouros (1982) DETERMINING THE NUMBER OF INTERNAL STABILITY OF A GRAPH. *International Journal of Computer Mathematics* 11 pp 207-220.

Mal86: E.M. Malstrom, H.D. Meeks & J.R. Fleming (1986) A ROBOTIC PALLET-LOADING ALGORITHM FOR DYNAMIC CARTON DISTRIBUTIONS *Proceedings of the 7th international conference on automation in warehousing, C.A. USA* pp 137-144.

Pen88: R.A. Penington & J.M.A. Tanchoco (1988) ROBOTIC PALLETIZATION OF MULTIPLE BOX SIZES *International Journal of Production Research* 26 (1) pp 95-105

Rin87: A.H.G. Rinnooy-Kan, J.R. DeWit & R.T. Wijmenga (1987) NONORTHOGONAL TWO DIMENSIONAL CUTTING PATTERNS. *Management Science* 33 (5) pp 670-684

Roo86: Roodman (1986) NEAR OPTIMAL SOLUTIONS TO ONE-DIMENSIONAL CUTTING STOCK PROBLEMS. *Computing and Operational Research* 13 (6) pp 713-719

Smi80: A.Smith, P. De Cani (1980) AN ALGORITHM TO OPTIMISE THE LAYOUT OF BOXES IN PALLETS. *Journal of the Operational Research Society* 31 pp 573-578.

Ste79: H.J. Steudel (1979) GENERATING PALLET LOADING PATTERNS: A SPECIAL CASE OF THE TWO-DIMENSIONAL CUTTING STOCK PROBLEM. *Management Science* 25 (10) pp 997-1004.

Ste84: H.J. Steudel (1984) GENERATING PALLET LOADING PATTERNS WITH CONSIDERATIONS OF ITEM STACKING ON END AND SIDE SURFACES. *Journal of Manufacturing Systems* 3 (2) pp 135-143.

Yao80: A. C-C. Yao (1980) NEW ALGORITHMS FOR BIN PACKING. *Journal of the Association for Computing Machinery* 27 (2) pp 207-229

APPENDIX A

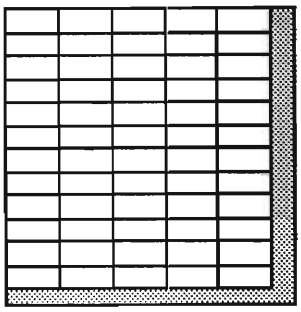
PALLET LOADING RESULTS

To facilitate camparison of the methods for solving the Pallet Loading problem, the examples presented in Chapter Two are reproduced here. Results produced for packing each example by the various methods are grouped together on a single page. The figures marked with an asterix (*) denote that the upper bound was reached.

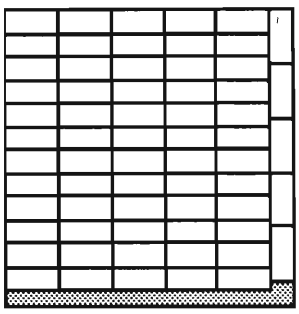
The test examples used for demonstration are given in table A.1 (Reproduced from Table 2.2)

Test	X	Y	a	b	Bound
a	38	38	7	3	68
b	20	20	7	2	28
c	20	15	7	4	10
d	20	15	7	3	14
e	14	11	4	3	12
f	14	13	4	3	15
g	22	16	5	3	23

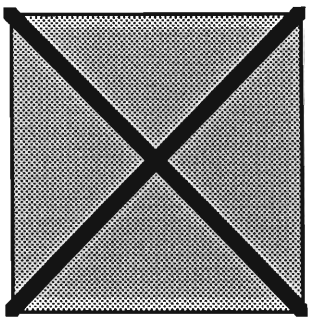
Table A.1: The examples used for demonstrating the Pallet Loading Heuristics of Chapter 2. The upper bound on the number of boxes that can be packed is shown.



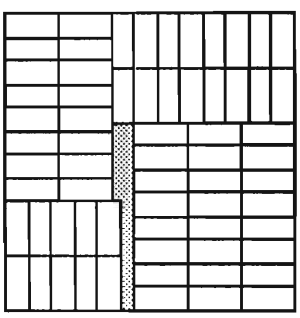
(i)



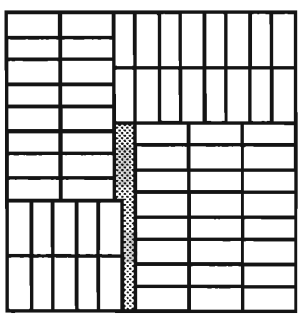
(ii)



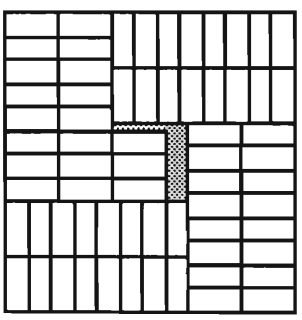
(iii)



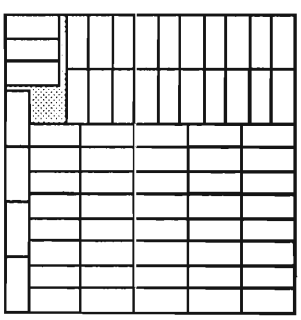
(iv)



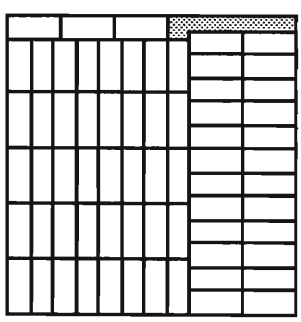
(v)



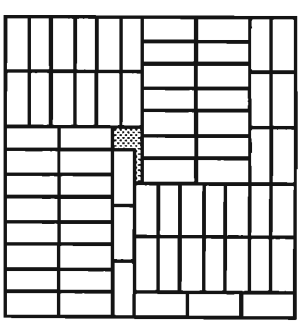
(vi)



(vii)

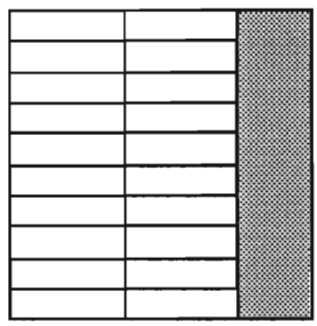


(viii)

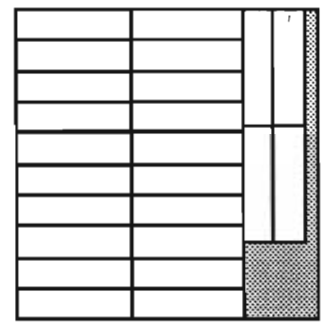


(ix)*

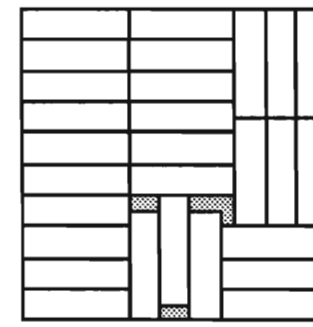
Pallet	:	38×38
Box	:	7×3
Upper bound	:	68
(i) - Basic	:	60
(ii) - Improved Basic	:	65
(iii) - [Dow87]	:	X
(iv) - [Ste79]	:	66
(v) - [Ste84]	:	66
(vi) - Improved Steudel	:	67
(vii) - [Smi80]	:	67
(viii) - [Bis82]	:	67
(ix) - [Dei91]	:	68



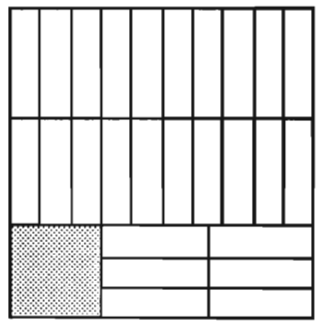
(i)



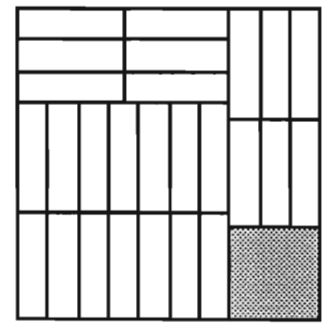
(ii)



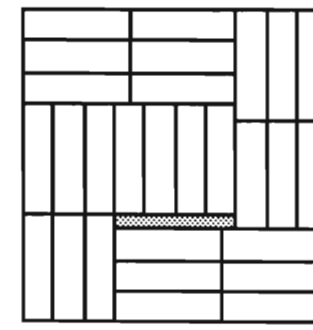
(iii)*



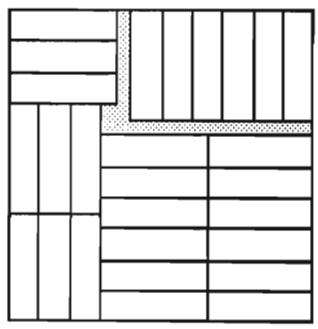
(iv)



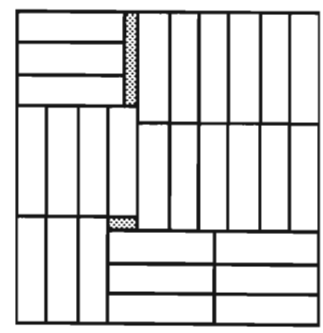
(v)



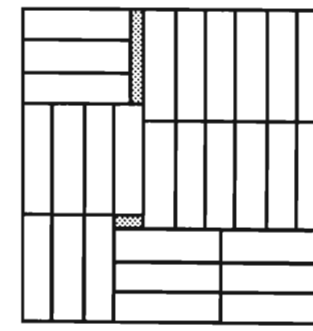
(vi)



(vii)

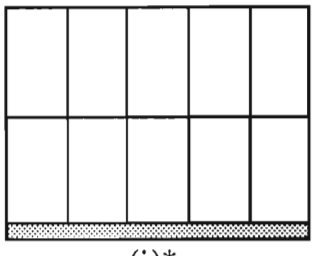


(viii)*

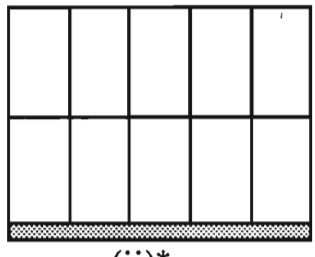


(ix)*

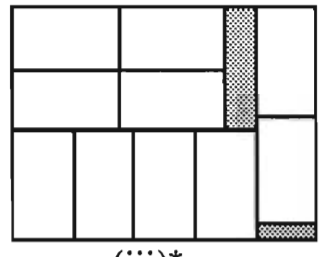
Pallet	:	20×20
Box	:	7×2
Upper bound	:	28
(i) - Basic	:	20
(ii) - Improved Basic	:	26
(iii) - [Dow87]	:	28
(iv) - [Ste79]	:	26
(v) - [Ste84]	:	26
(vi) - Improved Steudel	:	28
(vii) - [Smi80]	:	27
(viii) - [Bis82]	:	28
(ix) - [Dei91]	:	28



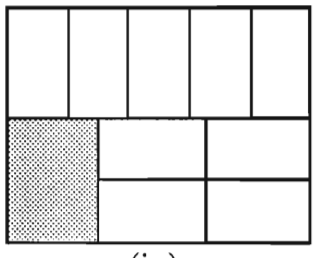
(i)*



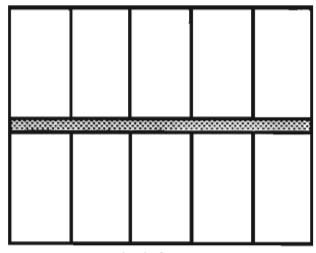
(ii)*



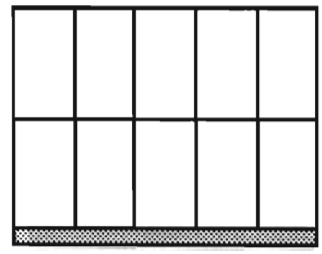
(iii)*



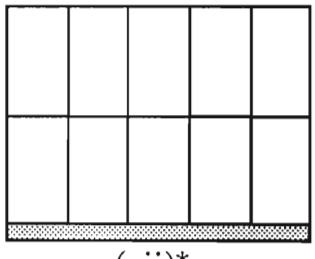
(iv)



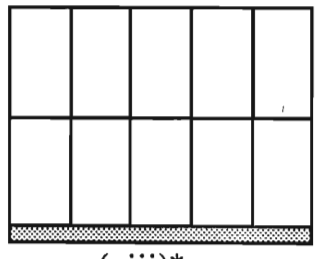
(v)*



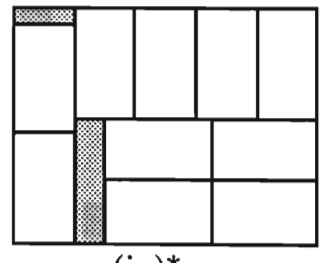
(vi)*



(vii)*

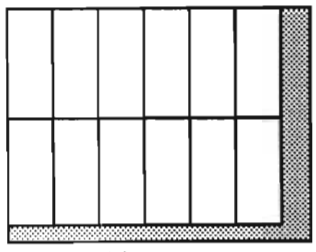


(viii)*

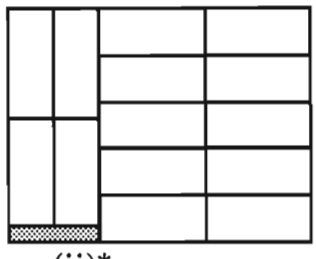


(ix)*

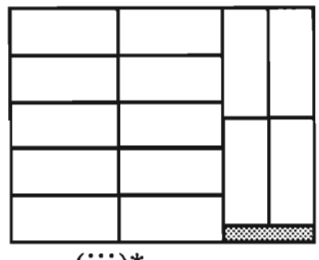
Pallet	:	20×15
Box	:	7×4
Upper bound	:	10
(i) - Basic	:	10
(ii) - Improved Basic	:	10
(iii) - [Dow87]	:	10
(iv) - [Ste79]	:	9
(v) - [Ste84]	:	10
(vi) - Improved Steudel	:	10
(vii) - [Smi80]	:	10
(viii) - [Bis82]	:	10
(ix) - [Dei91]	:	10



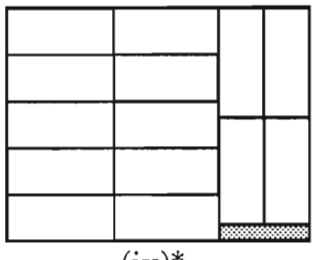
(i)



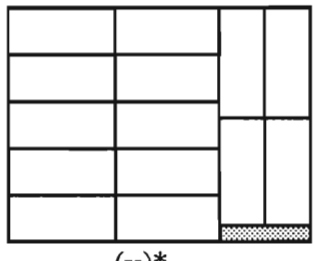
(ii)*



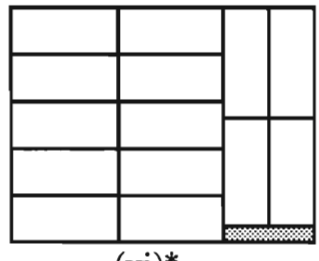
(iii)*



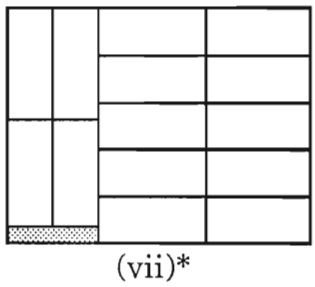
(iv)*



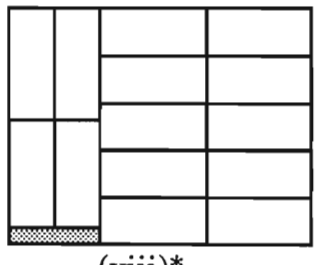
(v)*



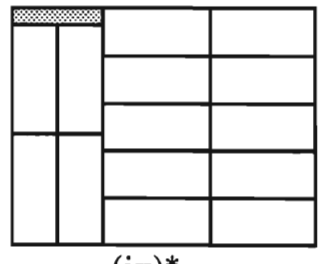
(vi)*



(vii)*

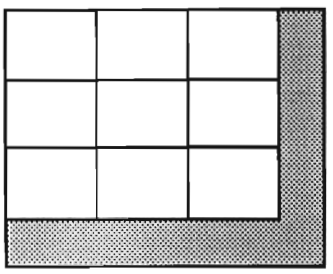


(viii)*

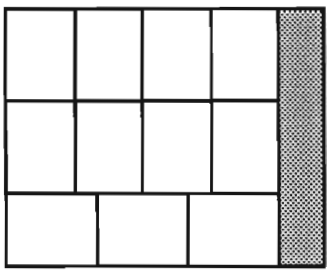


(ix)*

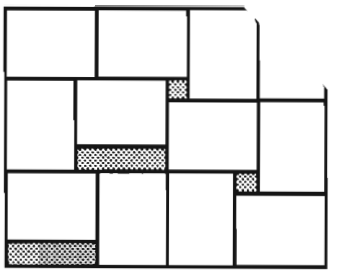
Pallet	:	20×15
Box	:	7×3
Upper bound	:	14
(i) - Basic	:	12
(ii) - Improved Basic	:	14
(iii) - [Dow87]	:	14
(iv) - [Ste79]	:	14
(v) - [Ste84]	:	14
(vi) - Improved Steudel	:	14
(vii) - [Smi80]	:	14
(viii) - [Bis82]	:	14
(ix) - [Dei91]	:	14



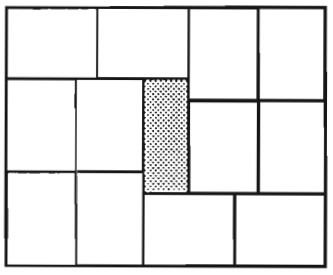
(i)



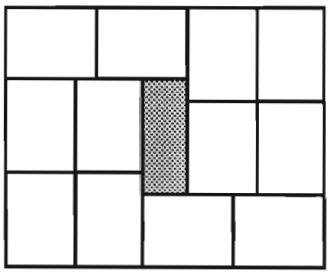
(ii)



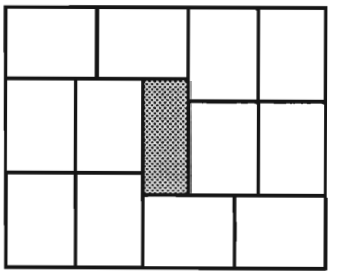
(iii)*



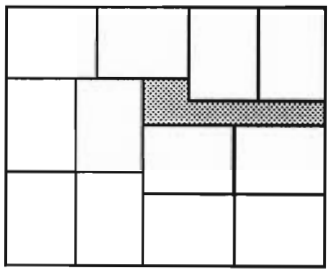
(iv)*



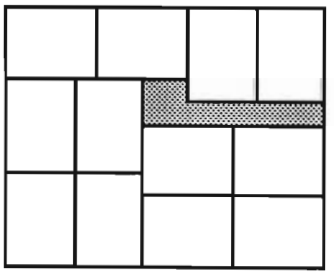
(v)*



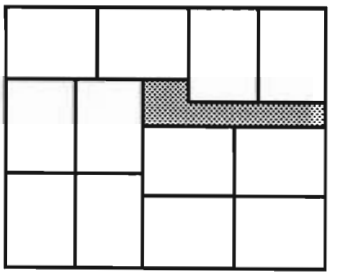
(vi)*



(vii)*

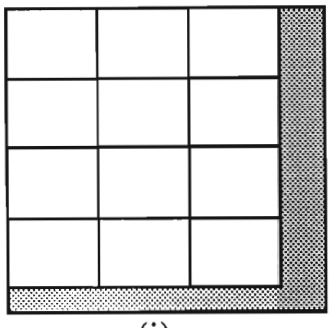


(viii)*

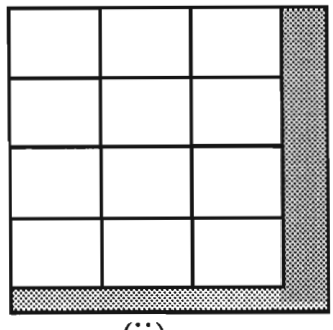


(ix)*

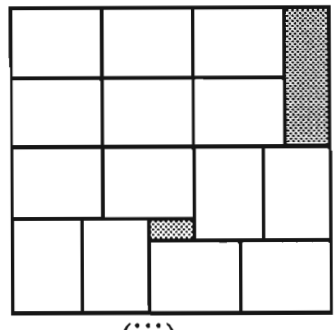
Pallet	:	14×11
Box	:	4×3
Upper bound	:	12
(i) - Basic	:	9
(ii) - Improved Basic	:	11
(iii) - [Dow87]	:	12
(iv) - [Ste79]	:	12
(v) - [Ste84]	:	12
(vi) - Improved Steudel	:	12
(vii) - [Smi80]	:	12
(viii) - [Bis82]	:	12
(ix) - [Dei91]	:	12



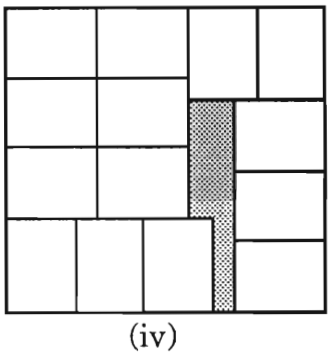
(i)



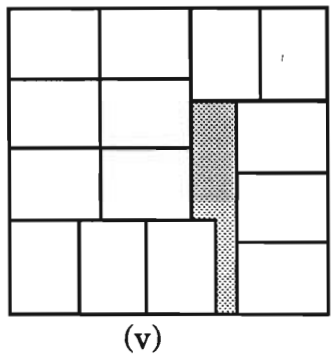
(ii)



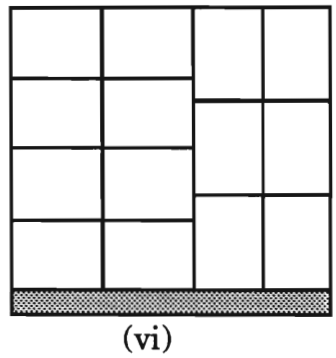
(iii)



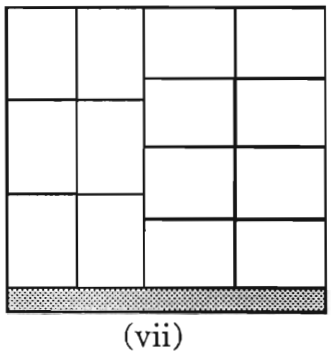
(iv)



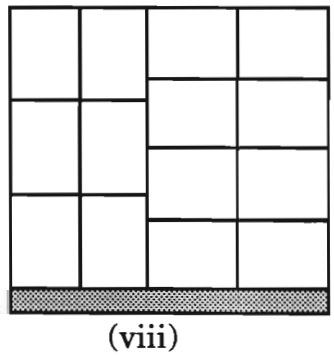
(v)



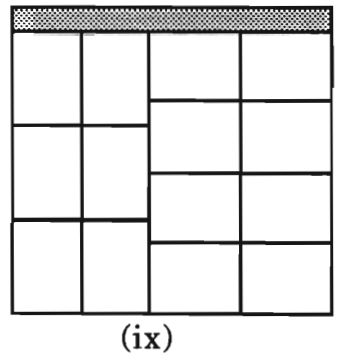
(vi)



(vii)

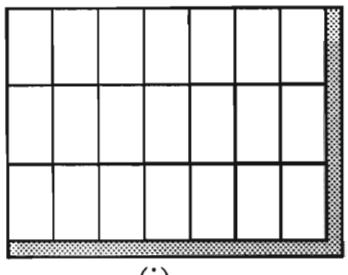


(viii)

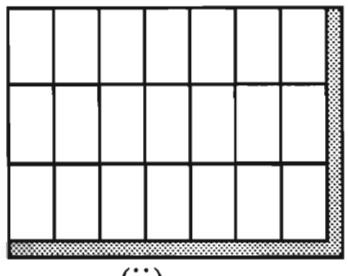


(ix)

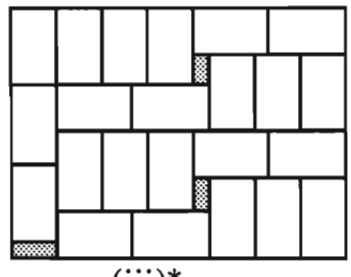
Pallet	:	14×13
Box	:	4×3
Upper bound	:	15
(i) - Basic	:	12
(ii) - Improved Basic	:	12
(iii) - [Dow87]	:	14
(iv) - [Ste79]	:	14
(v) - [Ste84]	:	14
(vi) - Improved Steudel	:	14
(vii) - [Smi80]	:	14
(viii) - [Bis82]	:	14
(ix) - [Dei91]	:	14



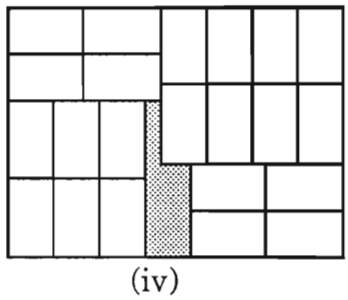
(i)



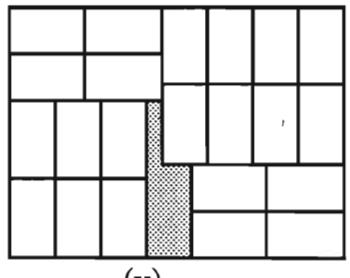
(ii)



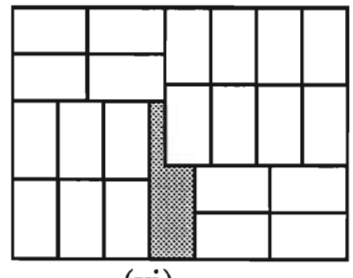
(iii)*



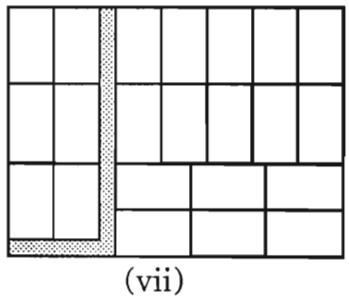
(iv)



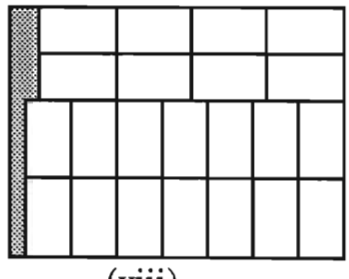
(v)



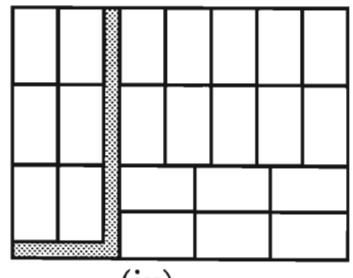
(vi)



(vii)



(viii)



(ix)

Pallet		:	22×16
Box		:	5×3
Upper bound		:	23
(i)	- Basic	:	21
(ii)	- Extended Basic	:	21
(iii)	- [Dow87]	:	23
(iv)	- [Ste79]	:	22
(v)	- [Ste84]	:	22
(vi)	- Extended Steudel	:	22
(vii)	- [Smi80]	:	22
(viii)	- [Bis82]	:	22
(ix)	- [Dei91]	:	22

APPENDIX B

MAXIMUM INDEPENDENT SETS

Dowsland's formulation of the pallet loading problem in terms of graph theory concepts [Dow87] requires that a method be applied to find a maximum independent set in the graph. A maximum independent set of a graph G is a set M of vertices of G such that no two vertices in M are adjacent.

Since Dowsland's algorithm is not dependent of the manner in which the maximum independent set is found, any method that solves this problem could be applied. The method of Loukakis and Tsouros [Lou82] was recommended by Dowsland.

This algorithm uses a branch and bound approach to solve the problem. Since finding maximum independent sets is NP-hard [Lou82], as the size of the graph increases, the computation time required for solving the problem increases rapidly.

The development of the algorithm requires the building of a search tree. Any node of this search tree can be fully described by a quadruple of the form

$$Q = (M, M^{+}, \tilde{M}, R(M))$$

where

M is the current independent set,

M^+ is a set containing no elements of M,

R(M) is the set of nodes that can be used to extend M, and

$\tilde{M}$ is the set of vertices not adjacent to M.

To move from one node to the next in the search, a vertex from the set R(M) is added to the set M to produce an independent set of higher order. The other sets of vertices are then updated to these changes.

The bounding phase of the algorithm implements various tests to determine wether or not a specific subset will contain a maximum independent set. These tests are based on the current elements of Q resulting from the previous branching phase. If this bounding process removes a node, then backtracking occurs, with the most recently added element of M being removed and inserted into M^+.

The following three tests are used for the bounding process:

MT1 Let $Q=(M, M^+, \tilde{M}, R(M))$, $u \in R(M)$ and $n(Adj(u) \cap Adj(R(M)))=1$ then any vertex set containing all of M and none of $M^+ \cup \{u\}$ cannot be a maximum independent set.

MT2 Let $Q=(M, M^+, \tilde{M}, R(M))$, $u \in \tilde{M}$ and $Adj(u) \cap R(M) = \{\}$ then adding any element of R(M) to M cannot produce a stable set.

MT3 Let $Q=(M, M^{+}, \tilde{M}, R(M))$ and M^{*} be the maximum stable set found so far. Then if $n(M) + n(R(M)) \leq n(M^{*})$ then no augmentation of M by elements of R(M) can improve on M^{*}.

The final algorithm is then stated formally as in algorithm A.1.

This algorithm formed the basis of Dowsland's method for the pallet loading problem, but only once the bounds had been tightened by introducing constraints from the physical problem was the approach feasible, as the computation time required was reduced.

Algorithm A.1

To find a maximum independent set in a graph.

```
proc MaxIndSet (G)
    M = []
    M+ = []
    M~ = []
    R(M) = []
    M* = []
    repeat
        while not (|R(M)|+|M| ≤ |M*|) or MT3 do
            u = FirstOf (R(M))
            if Adj(u) ∩ R(M) = [] then
                u = u*
            endif
            M = M ∪ [u]
            M~ = M~ - Adj(u)
            R(M) = R(M) - ([u] ∪ Adj(u))
        enddo
        if R(M) = [] and |M| > |M*| then
            M* = M
        endif
        if not (M=[] and |R(M)| ≤ |M*| or MT3 then
            r = LastNotMarkedOf (M)
            FR = M - [1..r-1]
            M = M - FR
            M+ = M+ ∩ [r] - [r+1..Max]
            M~ = M~ ∪ Adj(FR) ∩ M+ - Adj(M)
            R(M) - V(G) - (M ∪ Adj(M) ∪ M+)
        endif
    until M=[] and |R(M)|≤ |M*| or MT3
    Return(M*)
endproc
```