NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

MULTI-LAYER PALLETIZATION OF MULTI-SIZE BOXES FOR 2½D AND 3D PROBLEMS

Mingkai Yang

A Thesis
in
The Department
of
Mechanical Engineering

Presented in Partial Fulfillment of the Requirements
for
the Master Degree of Applied Science
at
Concordia University
Montreal, Quebec, Canada

March 1993

ISBN 0-315-84696-8

To My Parents

ACKNOWLEDGMENT

I would like to thank my supervisor Dr. G.H. Abdou for his guidance and support throughout the research. From Windsor to Montreal, it has been always a pleasure for me to work with him. I also want to thank Dr. M O.M. Osman for his generosity Without his financial support, this research could not have been carried out During my two years at Concordia, I have got so much help from so many people to whom I want to show my appreciation. Among them, I want to mention Mr Henry Hong, Mr Siyu Zhang, Mr. Feng Zhou, and Mr. Mourad El-Masry

ABSTRACT

MULTI-LAYER PALLETIZATION OF MULTI-SIZE BOXES FOR 2½D AND 3D PROBLEMS

Mingkai Yang

The palletization problem involves interlocking boxes on a pallet. There are two approaches dealing with the solution to this problem: mathematical and heuristic. Three models are proposed in the thesis. Model 1 is an Integer Linear Programming (ILP) model dealing with boxes with different lengths and widths but same height. Model 2 provides systematic procedures for 3D palletization problems by combining the ILP Model 1 and rules of thumb. Model 3 employs a heuristic, in which the main emphasis is on the random sequence of boxes. All three models are capable of performing multi-layer palletization. The pallet utilization, Work-In-Process (WIP), loading stability and palletization time are used as performance measures. Model 3 is implemented in a physical Robotic Palletization System (RPS) through a "C" program which integrates the subsystems consisting of a vision, a conveyor and a robot.

TABLE OF CONTENTS

LISTS OF TABLES

LIST OF FIGURES

NOMENCLATURE.

α	orientation angle of box
B_i	block i
cx	x coordinate of centroid of box
cy	y coordinate of centroid of box
d	density of box
D_i	number of Type i box available
F	static force
F_i	reaction force between boxes
F_s	minimum force needed to make a box slid
F_t	minimum force needed to make a box topple
F_{ss}	minimum force to make two box slid
F_{tt}	minimum force to make two box topple
F_{ts}	minimum force to make first box topple and second box slid
H_{max}	maximum palletization height
H_i	number of boxes in holding area i
H'i	the maximum number of blocks in holding area for group i
HL	height of a layer
H_r	the remaining height
h_i	the height of the boxes of group i
h	box height

H_{hi}	the difference between H_r and h_i
L	pallet length
l	box length
m	mass of box
MCH	number of robot motions between conveyor and holding area
MCP	number of robot motions between conveyor and pallet
MHP	number of robot motions between holding area and pallet
MPH	number of robot motions between pallet and holding area
N	the number of boxes on the pallet
N_i	the number of boxes in layer i
PIUP	frist pick-up location
PICKUP	second pick-up location
T_{ch}	time needed for robot move between conveyor and holding area
T_{cp}	time needed for robot move between conveyor and pallet
T_{hp}	time needed for robot move between holding area and pallet
T_{ph}	time needed for robot move between pallet and holding area
μ	friction coefficient between box and pallet
μ_1	friction coefficient between boxes
U	pallet utilization
Ud	desired pallet utilization
UL_i	utilization of layer i
UNLD	frist unload location

UNLOAD	second unload location
W	pallet width
w	box width
WIP	work-in-process
X_i	box of Type i
X_{ij}	number of Type i box to be put on layer j
Y_j	total number of boxes to be put on layer j

CHAPTER 1 INTRODUCTION

In industries, products are often packed in boxes which are loaded on pallets for purpose of transportation. The palletization problem involves interlocking boxes on a pallet. Traditionally, palletization is done manually by an operator who decides where and how a box should be placed on a pallet. Most of the time, this method can provide next-to-optimal solutions However, it has a drawback that the palletization speed is very slow Another palletization method is mechanized palletization, which is used in the process of large scale production. The mechanized palletizer can work at very high speed, which is suitable for automated manufacturing. However, the palletizer can be used only in non-flexible environment, where the sizes of both the pallet and the boxes have to be standard. With the development of FMS (Flexible Manufacturing System), a new method which can provide relative high loading speed in a flexible environment is required. The latest development is robotic palletization, in which a robot is controlled by a sequence program and is interfaced with a vision system. The robotics palletization method provides medium speed of pallet loading, high flexibility of box types, and more feasibility of automated palletization. As an important part of the manufacturing process and material handling, robotic palletization is attracting more and more interest from both industries and research institutes.

Palletization could take place in either a manufacturing cell where boxes are loaded on pallets and taken away or in a warehouse situation where boxes are loaded on pallet before shipping. In the former, most of the times, boxes are coming from a

conveyor in random orders and robotic palletization is required, therefore, the focus of palletization problems is mainly in the assembly situation Generally, there are four factors which have the influence on palletization.

1. Box Size

Very often, boxes in the palletization problem have the same dimension However, they could have different dimensions too when the order sizes are different or different parts are assembled on a same assembly line The box size increases with the increase of either the product size or the order size. Boxes with different sizes will produce different layout pattern on a pallet.

2. Box Weight

Generally, boxes are loaded in different layers (some times in columns) on a pallet. The number of layers is determined by the maximum palletization height Usually, in order to determine which layer a box should be placed at, pallet utilization, stability and retrieval order of boxes need to be considered However, box weight needs to be taken into account when the boxes are not solid enough or contain fragile items In, this case, the palletization height should be lower and boxes contains fragile items should be put on the top.

3. Box Orientation

Better interlocking pattern can be achieved if boxes can be rotated Generally, rotation about the height direction is allowed for any boxes. However, rotation about the length/width direction depend on the items inside the boxes. When the rotation about length/width direction is allowed, it is easier to reach better layout in terms of pallet

utilization However, the rotation may change the centre of the gravity of a box, as a result, increase the difficulty of loading boxes in terms of stability.

4. Box Density

Stability is a complicated issue. It is determined by pallet/boxes material and loading patterns which depend on the dimension, density and the mass distribution of boxes When a box contains large quantity of small items, such as cans, the mass distribution is almost uniform. However when a box contains irregular shape items with small quantity, the canter of the gravity of the box will possibly be far away from the geometry canter of the box During palletization, boxes will be loaded on the pallet according to their centre of gravity In a practical situation, the pallet/boxes material could be selected as desired, and the weight and the position of the centroid of boxes can be obtained by placing pressure sensors underneath the box so that the pressures at four corners of a box can be determined. The box size also has influence on the loading stability. Pallet filled with bigger boxes is more stable than pallet filled with smaller boxes. Therefore, the loading pattern will be changed if box sizes are changed.

5. Box Availability

The box availability plays an important role in palletization process. When the number of boxes is unlimited, the best loading pattern is easier to reach. However, in a assembly line situation, where boxes come from a conveyor in a random order, the box may not come when it is desired In order to reduce the number of boxes waiting for loading, boxes which are not the best choice for the available space on the pallet have to be selected.

Generally, palletization problems fall into two categories: the manufacturer's problem and the distributor's problem. The former deals with identical boxes, and the definition of the problem is based on the assumption that a manufacturer always produces the same product during a certain period of time The later involves boxes in different sizes, since a distributor usually has many products in different sizes [14]. However, compared with the manufacturer's problem, the distributor's problem is much more complicated and consists of many different situations. Therefore, it is necessary to categorize the problem from another point of view which can classify the distributor's problem into more details. A classification is given here according to the dimensions of the boxes:

Type 1 problem.	All boxes are identical.
Type 2 problem.	Boxes have different base dimensions (length and width) but the height is same for all boxes.
Type 3 problem.	Boxes can be grouped by same height, although they have different base dimensions.
Type 4 problem.	Boxes cannot be grouped by similar height, and have different base dimensions.

Obviously, Type 1 problem is a manufacturer's problem, and other three are distributer's problems. Each type includes different level of complexity While only one layer is considered, Type 1 problem and Type 2 problem are 2D palletization problems Type 1 problem is the simplest palletization problem, which is defined in some literatures as the 2D cutting problem. Type 2 problem is much more complicated than Type 1

problem because boxes are not identical anymore. Both Type 1 and Type 2 problem could also be defined as 2½D palletization problems if multi-layer palletization is the case in which the height has to be considered. The problem is not taken as of 3D due to the fact that for Type 1 problem and Type 2 problem all boxes have the same height. Type 3 problem adds one more complexity factor over Type 2 problem by considering that different boxes can be grouped by the same height, although the boxes have different base dimensions. There are two loading techniques [14] for this type of palletization problem: layered-palletization and stacked-palletization. In the former, boxes are loaded on to the pallet in form of layers, while in the later, boxes are stacked onto the pallet in form of columns. Type 4 problem deals with the 3D characteristics, except that all boxes have different heights and base dimensions such that boxes cannot be grouped. This is the most complex type among all palletization problems.

In this research, 2½D Type 2 problem and Type 3 problem are studied. Three models are proposed. Some example applications are applied to illustrate the efficiency of the models. A physical robotic palletization system is implemented.

CHAPTER 2 LITERATURE SURVEY AND OBJECTIVES OF THIS RESEARCH

Researchers have attempted to solve palletization problems by two approaches mathematical models and heuristics. Generally speaking, solving a palletization problem by mathematical models is only feasible when all the information of the pallet and the boxes is available. In this situation, the mathematical model is capable of producing optimal solutions. Earlier, researchers tried to find out the optimal solution to the problem by formulating mathematical models and succeeded when they dealt with simpler problems like Type 1 and some of Type 2 problems However, with the increase in the complexity of the problem, it becomes more and more difficult to solve the problem by mathematical models, even for Type 2 problem. Furthermore, in case of automated production process where the automated palletization system is fed by boxes from assembly lines in random sequences, the result from a mathematical model may not be useful because the boxes required by the optimal solution may not available In order to solve this problem, heuristic approaches are introduced. This makes the heuristic algorithms more and more popular to be used to cope with more difficult problems In addition, the rapid advances in microcomputer technology have made this approach feasible and efficient.

2.1 Literatures on Mathematical Models

Many researchers have worked on the simplest palletization problem, Type 1 problem with single layer Since it is a 2D problem, it is basically same as another traditional problem in the field of operational research, the 2D cutting problem. Hinxman had a summary on this issue (1980) [13].

Glimore and Gomory [11] tried to solve the 2D cutting problem by employing Linear Programming approach which involves solving an auxiliary knapsack problem to determine the successor solution at each stage of the simplex method. The limitation of their model is that the model requires that each cut has to be parallel to edges and cut through the whole sheet (guillotine cutting). Beasley [3] developed an Integer Programming model for a 2D cutting stock problem. His work focuses on cutting rectangles from a single sheet and is based on the assumption that the dimension of the rectangles must be integer. The Lagrangian relaxation-based tree search is adopted to produce optimal solutions within reasonable computing time. Chen et al. [7] improved Beasley's model by releasing the restriction on the integer solution space and tried to deal with multiple pallets. The orientation of the boxes on the pallet is considered as well. However, in his model, every pallet is still loaded with only one layer.

Tanchoco and Agee [21] highlighted the different interactions between box sizes, pallet sizes, load height, lift truck capacity and warehouse storage. If the size of pallet is specified, the best loading pattern can be selected. The limitation, though, is that all boxes are assumed to be identical. Smith and DeCani [20] presented a rectangle-packing algorithm for a 2D packing problem. They compared the exact and non-exact packing

method and found that the later is more efficient in terms of CPU time.

Tsai et al. [22] investigated Type 2 palletization problem with a Linear Programming model. They used the linear equation approach introduced by Brown [5] to generate constraints for the model. There is no restriction on the number of boxes for each type in their study. Consequently, it is possible that some types of boxes may not be chosen forever if only pallet utilization is considered.

2.2 Literature on Heuristics

Generally speaking, mathematical models are efficient for Type 1 problem and Type 2 problem when the necessary information is known However, when the problem is more complicated, it may be computationally expensive, or even not feasible, for instance, when the number of boxes are unknown in Type 2 problem. Therefore, many heuristic approaches have been developed to cope with this situation. Moreover, the combination of mathematical model and heuristic may yield better results in some cases

Hodgson [14] approached the 2D pallet loading problem by a combination of Dynamic Programming and heuristics. A system called IPLS (Interactive Pallet Loading System) has been developed. He also introduced the concept of layered pallet loading and stacked pallet loading techniques for 3D palletization problems But he did not give any details for solving a multi-layer palletization problem.

Another similar, but simpler problem is container loading, which differs to the palletization in that stability is less concerned in a container loading problem than in a palletization problem. There are many literatures on this issue George and Robinson

[10] studied a particular 3D container loading problem in which box dimensions and the number of boxes of each type are known. They introduced two rules for selecting an open box on a new layer. Rule A is that the type of box with the largest unpacked number is given higher priority; rule B is that the boxes with the largest smallest side are selected first. These two rules were tested and rule B was proven better. The heuristic succeeded 9 times out of 15 runs. Dowsland [8] proposed a set of strategies for developing heuristics for 3D packing problems. But no examples in the 3D problems are given in his paper. A more detailed discussion about container loading techniques was given by Bischoff and Marriott [4].

Abdou and Lee [1][16] developed a heuristic algorithm for a 2½D palletization problem. The algorithm is capable of dealing with multi-layer loading of boxes in random coming sequence, and considers the pallet utilization, palletization time, and work-in-process. A physical palletization system, which consists of a robot, a vision system, and a conveyor, has been implemented as well. A brief summary of the papers focusing on palletization is shown in Table 2.1[18]. A survey about these papers and other related articles are well presented in [12].

These studies have approached the palletization problems from different angles and have generated interesting results. However, there are still many aspects of palletization problems have not been studied thoroughly. The following lacunae are observed through the literature review:

- lack of a consistent performance measure system. In most studies, the pallet utilization is the only criterion.

- most of the studies only discuss 2D palletization problems 2½D problems have not received enough considerations
- 3D problem has not been studied Although the 3D problem was introduced by Hodgson (1982), no results were reported

Table 2.1 Summary of Literature Review

Attributes	Authors	Abdou 1990	Hodgson 1982	Penington 1988	Thai 1988	Chen 1991	Carpenter 1985	Smith 1980	Proposed 1993
Loading Method	manual		✓					✓	
	mechanic								
	robotic	✓		✓					✓
Dimension	2D				✓	✓	✓	✓	
	2½D	✓	✓	✓					✓
	3D		✓						✓
Availability	limited	✓		✓		✓			✓
	unlimited		✓		✓		✓	✓	✓
Loading Pattern	nested	✓			✓	✓			✓
	guillotine			✓			✓	✓	
Objective	pallet utilization	✓	✓	✓	✓	✓	✓	✓	✓
	stability								✓
	WIP	✓							✓
	loading speed	✓							✓
Loading Stage	single stage	✓			✓	✓	✓	✓	✓
	multi-stage			✓					✓
Method	mathematical				✓				✓
	heuristic	✓		✓		✓	✓		✓
	mixed		✓					✓	✓

2.3 Objectives of This Research

Palletization problems are too complicated to develop a general algorithm for all of them. Researchers have developed algorithms for problems with specific constraints on either pallet/box dimensions or packing methods, such as single-layer loading or multi-layer loading. In this study, the focus will be on the 2½D and 3D problems with some constraints as follows:

- First, boxes could have different dimensions, however, no boxes are allowed to be rotated along neither X nor Y axis. That is, each type of box could have only two orientations on a pallet since it can be rotated about only Z axis.
- Second, multi-layer palletization is considered, but the pallet dimension and the maximum palletization height are fixed.
- Third, no overhanging is allowed during palletization. That is, the area of upper layer must not greater than that of the lower layer.
- Box retrieval order is not considered.
- All boxes have the same density and the mass is uniformly distributed.

Under the assumptions mentioned above, four main objectives are set:

- Develop Model 1: a mathematical model for Type 2 problem. The model is capable of solving multi-layer palletization problems and considers the availability of the boxes as well as the loading stability.
- Develop Model 2: an algorithm for Type 3 problem by adding the 3D feature to the mathematical model developed in the first part.

- Develop Model 3: an algorithm for Type 3 problem which is characterized by random box sequences
- Implement a physical RPS (Robotic Palletization System) with Model 3 Develop programs for the vision system, robot system, and the system integration
- Compare the results from proposed models with those generated by the existing algorithms in the literatures

CHAPTER 3 PERFORMANCE MEASURES

To assess a palletization algorithm, some criteria are needed. Researchers have been focusing on the pallet utilization as a common objective in the study. However, in addition to utilization, there are other factors needed to be considered. When a physical palletization system is built, the performance of the system could be well assessed by four factors: pallet utilization, loading stability, WIP(work-in-process) and palletization time. However, for theoretical models, it is difficult to measure the palletization time for an algorithm. In this case, only WIP is used as the measurement of idle boxes between operations, which has great influence on palletization time.

3.1 Pallet Utilization

Pallet utilization U is expressed by the total volume of boxes loaded on a pallet divided by the maximum palletization volume which is the product of the pallet area multiplied by maximum palletization height H_{max}.

$$U = \frac{\sum_{1}^{N} l_i \times w_i \times h_i}{L \times W \times H_{max}} \qquad (3.1)$$

Since the palletization is carried out by layers, layer utilizations need to be calculated as well.

$$UL_i = \frac{\sum_{1}^{N_i} w_i \times l_i}{W \times L} \qquad (3.2)$$

3.2 Stability

Boxes loaded on a pallet are subject to a variety of forces including static forces and dynamic forces which occur during transportation The stability, an index which measures the ability of the boxes to maintain their position when the forces are applied on them, needs to be studied especially when a pallet is loaded with several layers of boxes. In this study, only static forces are considered

Very few studies have been reported on the issue of loading stability Loschau [17] discussed stability criteria from the point of view of inclination angles, but only for column stacks. However, for layered pallet loading, the issue of loading stability is much more complicated because of the nature of interlocking Different combinations of boxes may yield different stabilities.

Carpenter and Dowsland [6] considered the stability for the palletization problem Three stability criteria were developed by them as follows·

Criterion 1: Each box must have its base in contact with at least two boxes in the layer below. Contacts of less than 5% (or 15% depending on situation) of a box's base area will be ignored.

Criterion 2: Each box must have at least 75% of its base area in contact with the layer below.

Criterion 3: There must be no straight or jagged guillotine cuts traversing more than 75% of the stack's maximum length or width A guillotine cut is a complete vertical cut from top to bottom of a stack

Unfortunately, they did not show the investigation and how the criteria were introduced

Some rules concerning packing stability have been applied to the proposed models When boxes are loaded on a pallet, the loading stability depends mostly on the boxes on the top layer For boxes made of the same material, the bigger the box, the more stable it is Generally, when the differences among box dimensions are not too extreme, the less the number of boxes in a layer, the higher the stability. In this study, the number of boxes in a layer is taken as the stability index based on this assumption.

3.3 Work-In-Process (WIP)

Work-in-process is defined as the temporary storage of boxes waiting to be loaded on to the pallet between or at the end of operations. In different models, WIP is expressed by either the total area held by the boxes which are not loaded on the pallet or the number of boxes in the holding area. In case of automated palletization, the boxes are moved to the holding area or loaded onto the pallet by the robot after they come into the system. The number of boxes in the holding area is changing all the time. Therefore, the maximum number of boxes in the holding area (maximum WIP) is considered as the criterion Since the more the boxes in the holding area, the more movements of the robot is required to move boxes between the conveyor and the holding area, or between the holding area and the pallet, WIP can reflect palletization time in some degrees.

3.4 Palletization Time

The palletization time mainly depends on the movement of the robot used in the RPS and scanning process of the vision system, since the CPU time is considerably less When the actual palletization is carried out, the palletization time can be recorded and compared with. However, the palletization time in a simulation is hard to be obtained since there could be overlap between the time required by the robot and the time required by the vision system. In this research, palletization time is obtained by calculating only the time needed for the robot motions with the assumption that there is no idle time for robot. There are four types of motion for a robot during the palletization process move a box from conveyor to the holding area (MCH), move a box from conveyor to pallet (MCP), move a box from holding area to pallet (MHP), move a box from pallet back to the holding area (MPH). The total palletization time is calculated as follows

$$T=T_{ch}*MCH + T_{cp}*MCP + T_{hp}*MHP + T_{ph}*MPH \qquad (3.3)$$

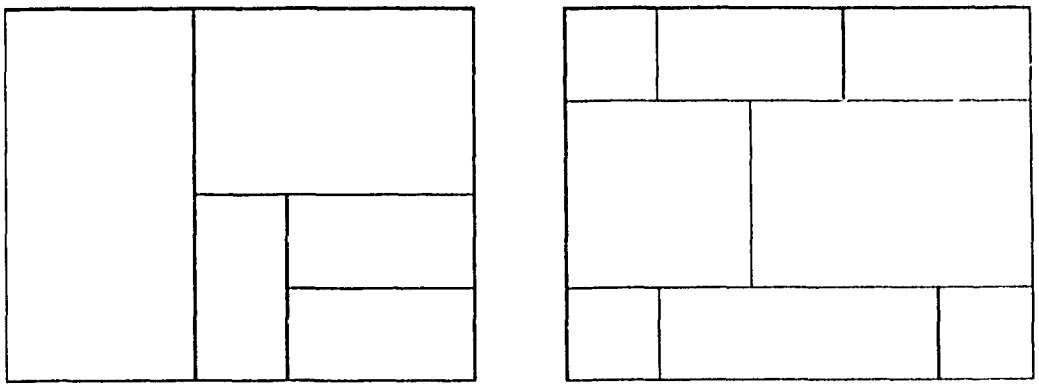
Note: * means product.

CHAPTER 4 PROPOSED MODELS

This research employs both mathematical method and heuristic approach to solve palletization problems. A mathematical model for 2½D problem is discussed frist. Tsai et al developed an Integer Linear Programming (ILP) model for Type 2 problem [22], i e boxes with different lengths and widths but same height. His model is complicated and only considers single-layer palletization. The proposed Model 1 is an ILP model too, which simplifies Tsai's modelling procedure and expands the scope to 2½D, i.e. multi-layer palletization In addition, the model considers loading stability and works for limited number of boxes. By combining Model 1 with some rules of thumb, the proposed Model 2 takes one step further dealing with Type 3 problem; i.e. boxes with different height could be loaded in layers. The model provides a systematic procedure for solving Type 3 problem mathematically. Model 2 handles the problem by dividing the original problem into several sub-problems, which can be solved by applying Model 1. Model 2, similar to Model 1, is suitable for the case where all the informations of the boxes are known. But they can not be used for the other cases involving random box sequences. Model 3 has been developed to accommodate the situation where boxes are coming in random sequences. Model 3 employs a heuristic to maximize the pallet utilization while minimizing WIP and palletization time.

4.1 Model 1: ILP Model for 2½D Multi-Size Palletization Problem

Many researchers have studied Type 1 problem and got satisfactory results Since all the boxes of Type 1 problem have the same dimension, it makes the formulation of mathematical models easier. For Type 2 problem, because of the different lengths and widths, it is difficult to build mathematical models. The problems requiring solutions with nested pattern are even more complicated than the ones with guillotine-cut patterns (Figure 4.1).



a. Nested Pattern **b. guillotine-cut patterns**

Figure 4.1 Pallet Loading Patterns

Even though some works have been done about Type 2 problem, very few consider multi-layer palletization. Proposed Model 1 deals with a 2½D problem which has the following conditions:

- there are N types of original boxes. Type 1, Type 2, ..., Type i, .. , Type N They have the same height but different lengths and widths.
- the number of boxes available for Type i D_i is finite.

- two orientations are allowed for each type of boxes on the pallet, therefore, the number of box types is virtually doubled to 2N.

An ILP model has been developed with two objective functions for this problem. The first objective is to reach the maximum pallet utilization in terms of volume, which has been the main objective of many studies. Since palletization goes to several layers, the loading stability has to be considered. Therefore, the second objective is added to ensure that boxes with bigger area are to be put on the top. The objective function of the proposed Model 1 is as follows;

$$Max\ P=\sum_{j=1}^{K}\sum_{i=1}^{2N}\alpha_i X_{ij}+\sum_{j=2}^{K}(Y_1-Y_j) \qquad (4.1)$$

where α_i is the area of box Type i, $\alpha_i = l_i * w_i$

X_{ij} is the number of box Type "i" to be put on layer "j"

Y_j is the total number of boxes to be put on layer j

(Y_1-Y_j) is the difference between the number of box in layer 1 and layer j. Maximizing the sum of (Y_1-Y_j) for all layers, the smaller boxes will be placed in the lower layers.

the pallet utilization can be obtained from:

$$U=\frac{\sum_{j=1}^{K}\sum_{i=1}^{2N}\alpha_i X_{ij}}{L\times W\times K} \qquad (4.2)$$

Four types of practical constraints are set and reflected in the proposed model They are as follows:

(1) Pallet Constraints

The first set of constraints relates to the dimensions of the pallet Since there is no overhanging allowed, the total area of all the boxes on each layer j must not exceed that of the pallet.

$$L*W-\sum_{i=1}^{2N} X_{ij}*\alpha_i \geq 0; \qquad \forall j \tag{4.3}$$

(2). Box Quantity Constraints

The number of boxes of each type available is limited It has to be considered that each box is split into two because of different orientations on the pallet

$$\sum_{j=1}^{K} X_{ij}+\sum_{j=1}^{K} X_{i+N,j} \leq D_i; \qquad \forall j \tag{4.4}$$

where D_i is the number of boxes of Type i and Type i+N available.

(3). Stability Constraints

These constraints ensure that the area of each layer could not exceed the total covered area underneath and that the number of boxes in the upper layer is always smaller than the number of boxes in the lower layer.

$$\sum_{i=1}^{2N} \alpha_i X_{ij} \geq \sum_{i=1}^{2N} \alpha_i X_{ij+1}; \qquad j=1,2,\ldots,K-1. \tag{4.5}$$

$$\sum_{i=1}^{2N} X_{ij}=Y_j; \quad Y_j \geq Y_{j+1}; \qquad \forall j \tag{4.6}$$

(4). Strip Constraints

Constraints in this group have been developed from Brown's linear equation approach [5] which can be described by the following example. Consider a pallet has a length of 4 and a width of 2. There are 3 types of boxes with size of (2,2), (2,1) and (1,2) The height is not considered here. First, divide the pallet into 2 unit width strips of (4,1) Second, find out the box length set, which is (2,1). Then find out all the possible length combinations so that the sum of the lengths of a combination does not exceed the pallet length 4. Eight length combinations which satisfy the condition are:

(a) {2,2}

(b) {2,1,1}

(c) {2,1}

(d) {2}

(e) {1,1,1,1}

(f) {1,1,1}

(g) {1,1}

(h) {1}

Let a denote the number of strips of Combination ***(a)***, b the number of strips of Combination ***(b)***, etc. The number of boxes of Type 1, Type 2, and Type 3 on the pallet is X1, X2, and X3, respectively. There are totally 2 strips, so,

$$a+b+c+d+e+f+g+h=2$$

Since there are two types of boxes of length of 2, of which one has width of 2 and one has width of 1. There are must be 2(X1+X2) strips of length 2 This gives us the equation:

$$2a+b+c+d = 2(X1+X2)$$

Same for the boxes with length of 1, there is only one type Therefore,

$$2b+c+4e+3f+2g+h = X3$$

The three equations above become constraints when X1, X2, and X3 are decision variables in a model. The number of strip constraints increases with the increase of the number of box types.

A BASIC program which consists of 3 modules has been developed for the formulation of the model (APPENDIX B). The flow chart of the program structure is shown in Figure 4.2.

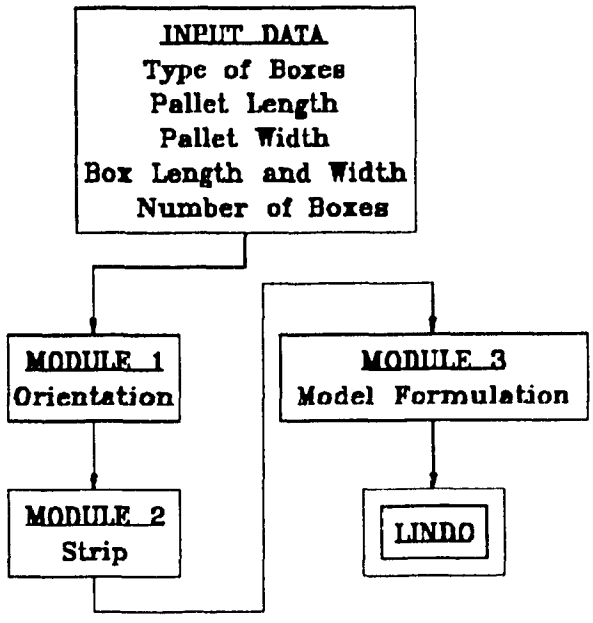


Figure 4. 2 Program Structure

The three modules are.

(1). **Orientation module:** It generates two copies of each type box; one in which the box length lies along the pallet length, and the other in which the box width lies along the pallet length

(2). **Strip module:** It generates all possible combinations of unit length strips of different width, which make up the total length of the pallet.

(3). **Formulation module:** It formulates the problem and store it in an ASCII format which is readable by Lindo.

Methods for Solving LPs and LINDO

LP is a classic problem in the field of operations research. There are mainly two kinds of methods for solving LPs: the graphical method and the algebraic method. The graphical method is an acceptable means of solution if there are only two decision variables. The algebraic method which has been proven to be very successful is the Simplex method, which was developed by George Dantig in 1940's. The essential steps of this method which basically follows a path around the exterior of the feasible region are still incorporated in almost every LP solution procedure used today.

However, the simplex method may take extremely long time to solve problems with very big matrices. Fortunately, the real world is easier than the theory that the standard simplex method suggests. In a typical real world problem, less than 2% of the coefficients will be nonzero. The Revised Simplex method has been developed to exploit this sparsity. It allows the same path to the optimum; however, it stores the results of

the computations in a way that exploits this sparsity so that the number of storage elements required is approximately twice the number of nonzeros The revised simplex method is more applicable to the real world problems and more efficient

Contrast to simplex method, many people have proposed an interior path method The interior path method first popularized by Karmarkar(1985) His algorithm may be competitive with the Simplex method on certain very large problems with very few nonzero coefficients since there are no problems which require an extremely long time to solve with his algorithm

LINDO is a commercial package for solving LPs problems LINDO stands for Linear, INteractive, Discrete Optimizer. It is designed specially for solving LPs(linear programming). The package is designed based on the revised simplex method to solve regular LPs, Quadratic Programming, Integer Programming and Goal Programming It is capable of solving problems with 5000 rows and up to 15000 variables [19]

For the application in solving the integer LP proposed in this study, LINDO gives satisfactory performance. It solves the illustration problem in Chapter 7 in just seconds

4.2 Model 2: Systematic Procedures for Type 3 Problem[2]

In the previous section, an ILP model has been introduced for Type 2 problem However, another dimension of complexity is of significant interest, that is Type 3 problem. Boxes in Type 3 problem have different lengths, widths, and heights However, the number of different heights is finite, therefore, the boxes can be grouped into several groups according to specific heights The easiest way to load these boxes onto a pallet

is to allow only boxes of the same height to be loaded on a layer. Obviously, it may end up with lower pallet utilization by doing so because there may not be enough boxes with the required height A feasible solution is to allow boxes of different heights to form blocks with specific height which corresponds to number of layers on the pallet. The block forming process is very complicated especially when there are many boxes available since the number of combinations by boxes increases dramatically when the number of boxes increases The problem to be discussed in this section is Type 3 problem with the following specifications:

- there are N types of original boxes. Type 1, Type 2, ..., Type i, ..., Type N. They have different lengths, widths and height. However, the number of heights is limited.
- the number of boxes available for Type i D_i is finite.
- two orientations are allowed for each type of boxes on a pallet, therefore, the number of box types is virtully doubled to 2N.
- multi-layer palletization. Each layer has a constant height, however, boxes of different heights are allowed in a layer.
- the maximum palletization height is pre-determined.

For a 3D palletization problem like this, it is very difficult to deal with by a mathematical model like the one proposed earlier. Therefore, some rules of thumb are adopted. The main idea is to divide the pallet into several layers of certain heights. Each

layer remains a constant height. Then, the optimum layout of each layer can be determined by the proposed Model 1 in 5.1. After, the order of layers is determined by conducting a stability criteria analysis. A detailed explanation of the different steps of the proposed approach is as follows:

Step 1 Defining Blocks

The first step in the heuristic is to change the objects from boxes to blocks Three sets of box dimensions are given:

length set $L = \{l_1, l_2, ..., l_i, ...\}$

width set $W = \{w_1, w_2, ..., w_j, ...\}$

height set $H = \{h_1, h_2, ..., h_k, ...\}$

A block is defined as a space with a dimension of $l_i * w_j * h_k$ A block must be filled 100% by boxes and contain at least one box It is noted that a block can be formulated in many ways most of the time This fact increases the complexity of the problem. Changing the study subject from boxes into blocks makes it possible that boxes of different heights could be loaded on same layer Because now, even though only blocks with same height is allowed for a layer, these blocks could contain boxes of different heights.

Step 2. Formulating Blocks

The boxes are stacked both horizontally and/or vertically to form blocks However, a block could be formed by different combination of boxes For example, a

block could be made up by only one big box, or by a combination of several small boxes. The ways to formulate blocks depend on both block dimensions and box dimensions.

Step 3 Determining Initial Block Availabilities

Since blocks could be built in different ways, given certain box availability, there could be many possibilities of block availability In Type 2 problem, the object studied is the boxes with certain box availability In Type 3 problem, this subject is changed into blocks with certain block availability at this step. However, the complexity of this problem is much higher than that of Type 2 problem. Since each block could be made up by different box combinations, the number of alternatives of block formulation increases very rapidly as the box availability increases. To solve the problem of all different block availabilities is very time consuming and not economical justifiable. One way to simplify the problem is to start with the initial block availability, which means that each available block consists only one box.

Step 4. Determining All Possible Layer Combinations

In Type 2 problem, the total number of layers on the pallet could be derived directly by dividing the maximum palletization height, H_{max}, by a fixed block height, h. But for Type 3 problem, the layer's layout on the pallet may not be unique, due to the difference in heights. There could be a number of layer combinations on the pallet depending on the block heights and the maximum palletization height. For example, if there are three different heights, h_1, h_2, and h_3, the number of combinations of layers can

be generated. If the number of layers of the same height on the pallet are denoted by a, b and c, respectively. Therefore, the following constraint must be satisfied

$$a*h_1 + b*h_2 + c*h_3 \leq H_{max} \qquad (4.7)$$

The number of solutions to this inequality give the total number of layer combination (a,b,c). Each layer combinations is considered as an alternative for the palletization problem. So, the solution to the original problem is to be obtained from one of the alternatives.

Step 5. Solving Problem of Layer Combination i with Block Availability j

This is done by determining the optimum layout for each layer separately Since only one block height is allowed for each layer, this becomes Type 2 problem which could be solved by the ILP proposed in 5.1. If a desired utilization level is not reached, then other blocks formulations should be selected to formulate blocks with new availability. The process continues until the desired utilization level is reached or the utilization starts stabilizing or decreasing. After the desired level is reached, the solution to the layer combination i is found, then repeat this step for another layer combination

There are some rules to follow for generating new blocks

- use boxes other than those have been selected for the previous solution
- building bigger blocks first.
- use as less boxes as possible to built a block.

Step 6 Selecting solution of all Layer Combinations Bases on Pallet Utilization

After all solutions are found for all layer combinations, the selection process could begin for the best solution to the original palletization problem. The comparison between layer combinations is conducted in several stages and is based on different priority levels. First priority is given to the pallet utilization. At this level, some layer combinations will be eliminated because of the lower utilization compared with others

Step 7 Determining the Layer Orders on the Pallet for Each Layer Combinations Selected in Step 6.

After the layer combinations are selected based on utilization criterion, the next step is to determine the layer order, the order of loading these layers on to the pallet. The layer order is very crucial, because different orders may yield different stability index for the layout of a layer combination. The stability index is expressed by the minimum force needed to make any box on the pallet to topple or to slide. There are three basic rules for determining the layer order based on stability analysis:

1). the area of a layer could not exceed the area of the layer underneath, that is no overhanging is allowed.

2). the layer with higher stability goes to the top.

3). The upper layers should have as less numbers of boxes as possible compared with lower layers.

By applying these rules, each pre-selected layer combination should have unique

layout which gives the best stability values If more than one layout have the same stability value, then the WIP of these layouts should be compared If there are still layouts have the same value for WIP, then choose one layout arbitrarily

Step 8. Selecting solution of all Layer Combinations Bases on Stability Criterion

After step 7, each layer combination has reached a certain loading pattern In another word, each layer combination has certain value for stability and WIP now At this step, all layer combinations with lower stability indexes will be eliminated

Step 9. Selecting solution of all Layer Combinations Bases on WIP Criterion

The last step is to conduct the last comparison over WIP and to make the final selection. The layer combination with the values achieved at step 5 will be selected for the solution to the original palletization problem. There could be some alternative solutions existing if some layer combinations have the same values for pallet utilization, stability, and WIP.

Among these nine steps, Step 1 to Step 4 are of block formulation. Calculation and Optimization is performed in Step 5. Step 6 to Step 9 are all for selection The comparison based on those three selection criteria, utilization, stability and WIP, are called repeatedly. The comparison is conducted among layer combinations and among layer orders for each layer combination. The systematic procedure of this palletization process is shown in Figure 4.3

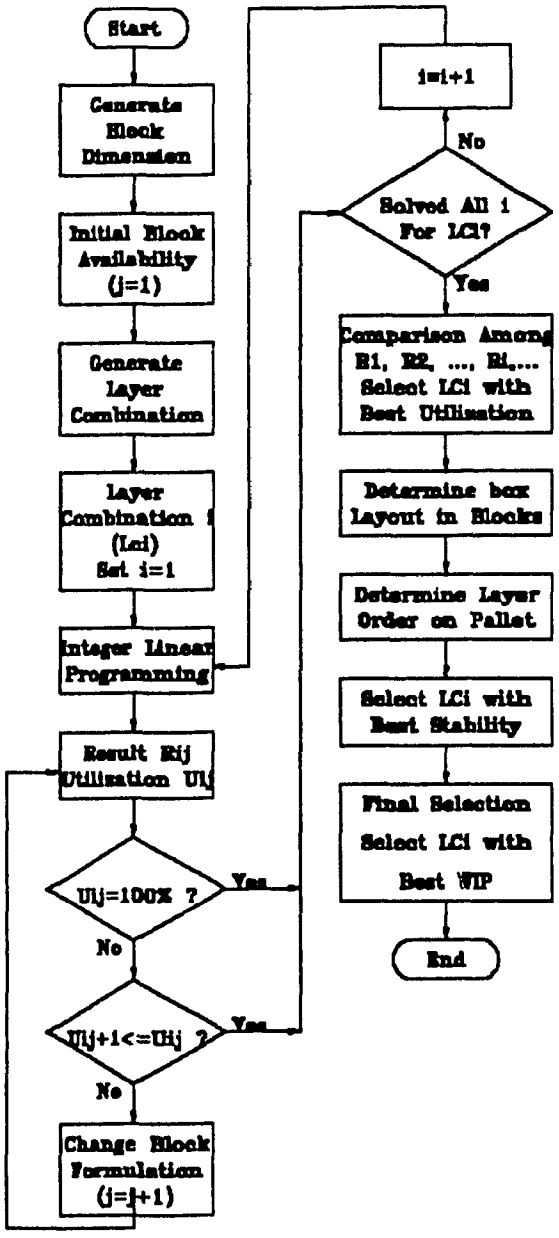


Figure 4.3. Flow Chart for 3D Palletization

4.3 Model 3: Heuristic for Type 3 Problem With Random Box Sequences

Both Model 1 and Model 2 use mathematical approaches to solve problems in a situation where all the informations about pallet dimension, box dimensions, and number of boxes available are known and all boxes are available any time. They are suitable for the material handling system in a warehouse, or distributing centre where all the merchandise are in stock. However, in manufacturing, especially in an automated production environment, the palletization usually takes place at the end of the production line. In this situation, the boxes are often coming from the production line in random sequences. That is, the availability of each type of boxes is not a constant Therefore, the problem cannot be handled with mathematical models like Model 1 and Model 2 proposed earlier. In order to tackle the problem like this, heuristics are needed. Some studies have been conducted and a few models have been developed for Type 1 problem and Type 2 problem. However, there is no literature on the heuristics for 3D problems

The proposed Model 3 employs a heuristic to deal with Type 3 problem with the following specifications:

- boxes can be grouped according to the height A group is a collection of boxes with same height
- maximum number of groups is 3. Boxes in Group 1 have height of h1, which is the shortest height. Boxes in Group 2 have height of h2 Boxes in Group 3 have height of h3, which is the tallest
- each group has a holding area

- the objective is to reach the maximum utilization and maintain low WIP
- layered-palletization technique is applied
- there is no restriction on the number of boxes available
- boxes are coming from a conveyor in random sequences
- single-pallet palletization

Since the problem is of 3D, the model should be capable of maximizing the total palletization space, i e. each layer's area and total palletization height. Generally, there are two basic criteria which can be applied to palletization: one is pallet utilization, another is loading stability. However these two objectives conflict with each other. If the utilization is the priority, the bigger boxes should be selected for lower layers. On the contaray, bigger boxes should be placed on the top when the loading stability is the main concern. Since utilization is the main goal in most situations, it is selected as the objective of the proposed model The heuristic consists of several steps as follows:

Step 1 Start

Initialize the system. Set the remaining palletization height equal to palletization height.

Step 2 Checking The Remaining Palletization Height H_r

If H_r equals to zero, that means all the palletization space has been filled, or H_r has a value, but it is less than the smallest box height, then terminate the procedure. Otherwise, continue.

Step 3. Checking Layer Status

if a new layer needs to be selected Continue Step 4

if a layer has already been started and waiting to be filled, then go to Step 5

Step 4. Selecting the Height for the New Layer

Before load any boxes on a new layer, the height of this layer must determined Because there are three heights in total, h1(minimum), h2, and h3(maximum), and h3 can be formed by combining h1 and/or h2, the priority of the selection of the layer height is given to the highest height possible. The selection follows the following rules

a. if there is a box of group 3 coming into the system, the height of the layer (HL) is set to h3;
b. if there is no box from group 3, and there are two boxes from group 2, then , HL=h2;
c. if there is no box from group 3, and there are less than 2 boxes from group 2, but there are more than three boxes from group 1, then, HL=h1.

Step 5. Fill a Layer with a Box from Group i (Current Layer: hj)

After the height of the layer is set, next is to fill this layer with the boxes or blocks with same height. Before any comparison among boxes is conducted, use all boxes to formulate blocks with the height of hj in the way introduced in section 5.2. Then the blocks are treated as boxes. For selection of box to be loaded on to the pallet, there are some rules to follow:

- select bigger boxes if they fit the remaining area of the layer.
- if the last box of this group in the pool can fit the remaining area of the layer and can complete the layer, select it. Otherwise, leave the smallest box in the holding area.
- if the new box is too big for the remaining area, and there are boxes on the pallet which are smaller than the new box, remove the smaller boxes, and restart this step.

After a layer is completed, go back to Step 2, and repeat Step 2 to Step 5 until the remaining palletization height is zero. The flow chart of this heuristic is shown in Figure 4.4.

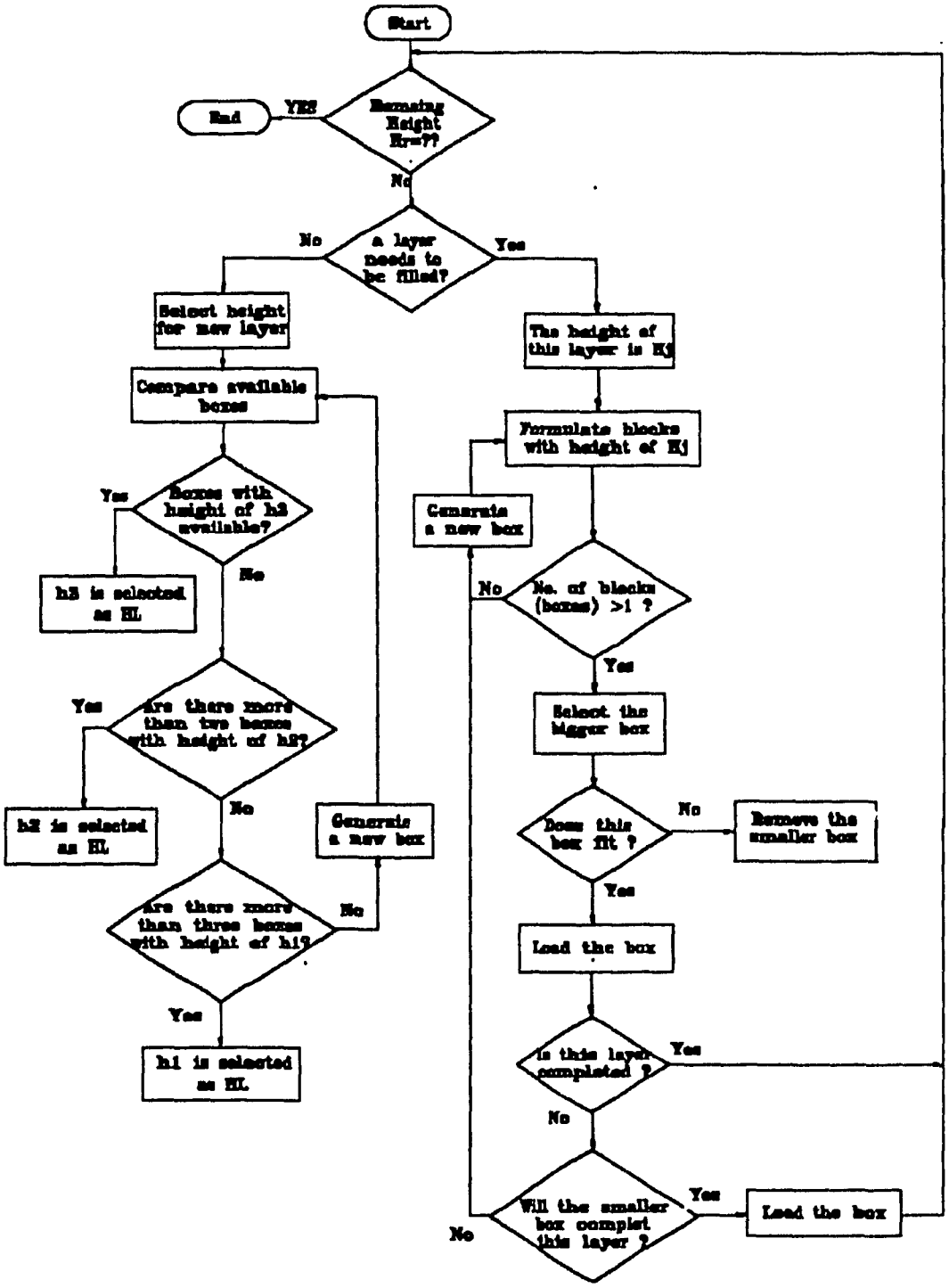


Figure 4.4 Flow Chart for Model 3

CHAPTER 5 PHYSICAL IMPLEMENTATION OF RPS

The proposed Model 3 has been developed for the industrial application where palletization is a part of an automated production process. In order to test the Model, a physical RPS (Robotic Palletization System) has been built. The system includes three sub-systems Vision System, Robot System, and Conveyor System. The three sub-systems are integrated by a 486DX personal computer. Certain softwares such as IP-8 image processing library and RAPL II robot control language are activated by the developed programs written in C language The main program performs integration and interprets the logic of Model 3. Subroutines for robot motion and Vision System have been also developed In this chapter, the hardware, the software, and the developed programs implemented in each subsystem of the RPS are described first, and then, the overall system operation is illustrated by an example.

5.1 Vision System

The vision system consists of an 8-bit image processing digitizer and two cameras. The two cameras grab the images of the box from different views to capture three dimensions. Then, the image is digitized and is processed by the vision subroutine. The vision system outputs the dimensions and the position of the box. Afterwards, the information is sent to the robot controller for box pick-up.

IP-8/AT Videographics System Board[15]

IP-8 is an image processing digitizer developed by Matrox Electronic System Ltd IP-8 is designed to capture either RS-170A or CCIR video images The input section allows up to four cameras to be connected The grabbed image can be sent to the frame buffer either directly, or keyed with a frame buffer mask and then sent to the display The IP-8 block diagram is shown in Figure 5 1 and the specifications of IP-8 are as shown in Table 5.1.

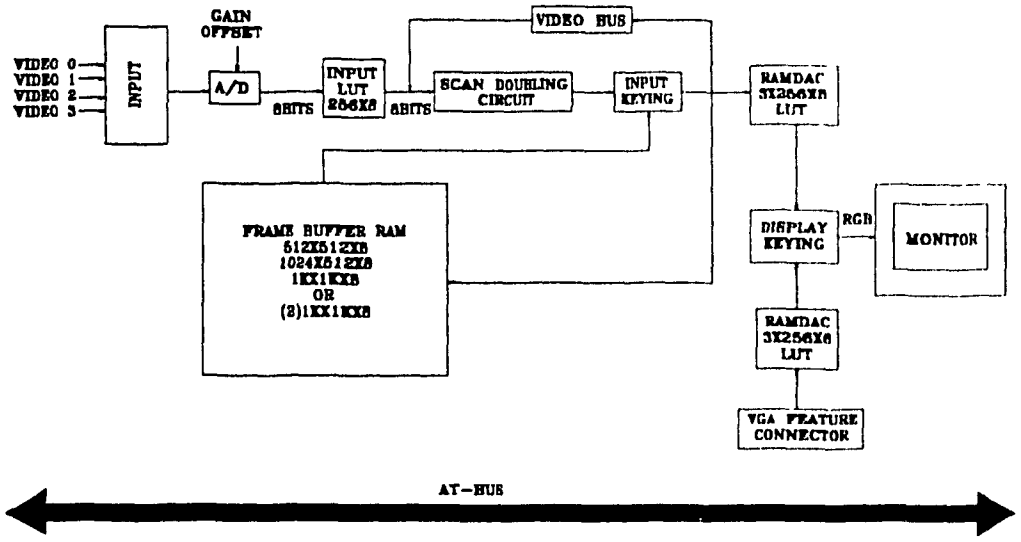


Figure 5.1 IP-8 Block Diagram

IP-8 provides a "C" library containing over 150 routines to accelerate applications development by allowing a high-level programming language such as Turbo C or Microsoft C to be used. The functions fall into thirteen categories General-purpose functions, Video functions, Lookup Table (LUT) functions, Memory Access functions,

Memory Display functions, Keying functions, Grab functions, Window functions, Image functions, graphics functions, Text functions, Picture File functions, Status function.

Table 5.1 The Specifications of IP-8

Host type	PC AT and compatible
Data bus width	8 bits in real mode (DOS), 16 bits in protected mode (OS/2)
Power consumption	8 Watts typical
Card size	13" x 4 5" x 0 6" (33 5 cm x 11 3 cm x 1.5 cm)
Operating temperature	0°C to 55°C (32°F to 132°F)
Color resolution	8-bits/pixel, select 256 colors or intensities from a palette of 16 7 million colors
Variable display resolution	512 x 400 to 1024 x 768
Video bus speed	Up to 30MHz
Horizontal scan rate	7 5 to 34 KHZ
Vertical scan rate	7 5 to 100 Hz
Interlace	Programmable 2 1 interlaced or non-interlaced
Capture window	Programmable. Step size. one pixel
Line capture	Programmable Step size one line
Video output	RGB (VGA levels, syncsand timings-non-interlaced mode)
Output connector	15-pin high-density D-type connector VGA compatible

PULNIX TM-745 Camera

Two Pulnix TM-745 shutter cameras are selected for the 3D image acquisition. The specifications of the camera are shown in Table 5.2

Table 5.2 The Specifications of Pulnix TM-745 Camera

Imager	2/3" interline transfer CCD, HAD type
Pixel	768 (H) x 493 (V)
Cell size	11 μm x 13 μm
Scanning	525 lines 60 Hz, 2 1 interlace
Sync	Internal/external auto switch, HD/VD, 4.0 Vp-p impedance 4.7 KΩ, VD=interlace/non-interlace, HD=15 734KHz±5%
TV resolution	570 (H) x 350 (V)
S/N ratio	50 dB min
Min illumination	0.5 lux f=1 4 without IR cut filter
Video output	1.0 Vp-p composite video, 75Ω
AGC	ON/OFF
Gamma	0.45 or 1 0
Lens mount	C-mount std., mini-bayonet optional
Power required	12 DC, 350 mA
Operating temp	-10°C to 50°C
Vibration & shock	Vibration. 7G (200 Hz to 2000Hz) Shock 70G
Size (W x H x L)	42 (mm) x 32 (mm) x 130 (mm) or 1 65" x 1 26" x 5 12"
Weight	200 gms
Power cable	12P-02
Power supply	K25-12V or PD-12
Auto iris connector	PC-6P

The vision system determines the information of boxes coming from the conveyor The two cameras capture two pictures of a box, one from the top, another from the side, so that all three dimensions could be obtained Since the conveyor belt is black, the

boxes are designed to be white for better contrast A "C" program has been developed (Appendix D) The program structure is shown in Figure 5.2

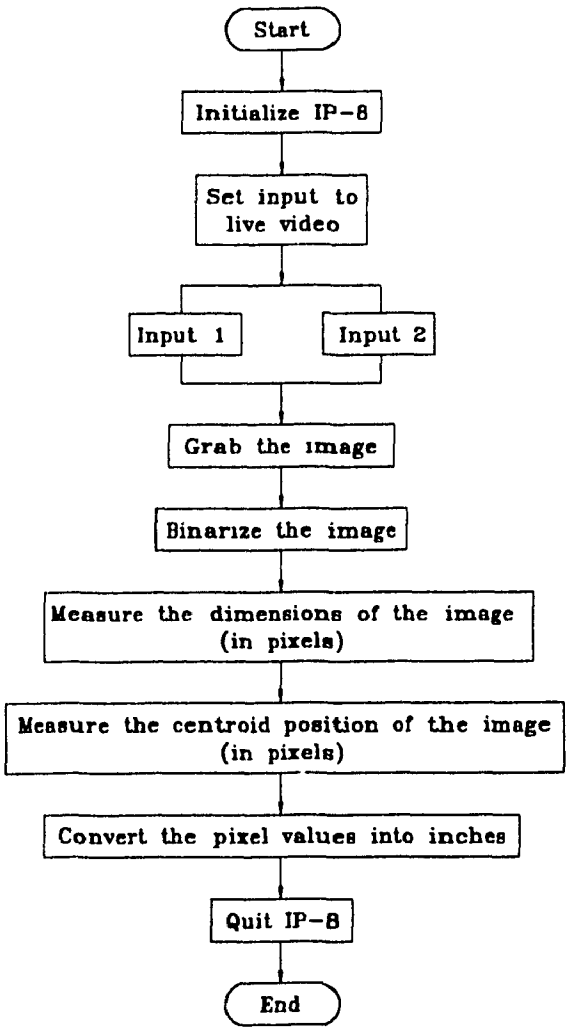


Figure 5.2 Program Structure for Vision Subroutine

There are two versions of this program. One program deals with the case in which there is no orientation angle is allowed That is , the edges of the box being processed

have to be parallel or (normal) to the camera frame. In this case, the length and the width of the box can be measured directly by calling the **CalcProfile** function provided by IP-8 and the processing time for each object is within five seconds Another program deals with the other case in which the orientation angle exists The program uses a scanning process which starts from the left-top corner of the screen. When horizontal scan has the priority, since the image is white and background is black, the top-left corner of the image (x1, y1) is found when a white pixel is encountered When vertical scan has the priority, the left-bottom corner of the image (x2, y2) can be found in the same way

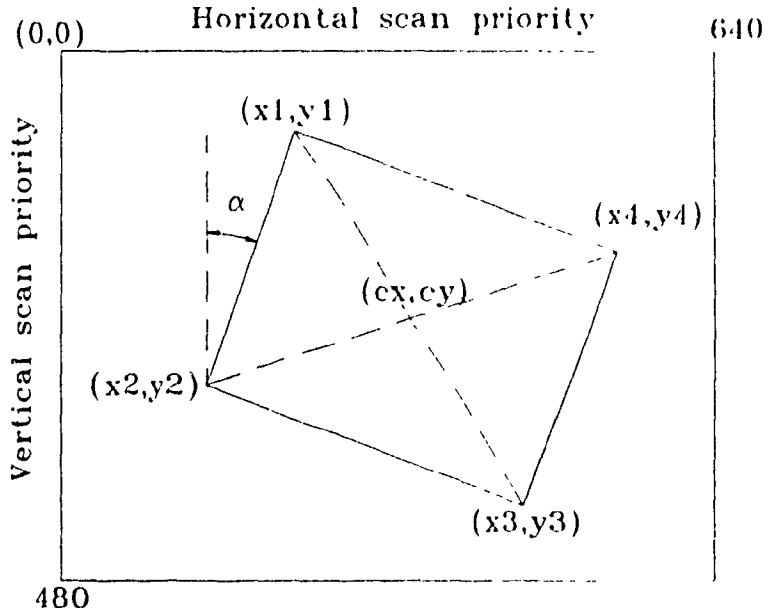


Figure 5.3 Image Processing with Orientation Angle

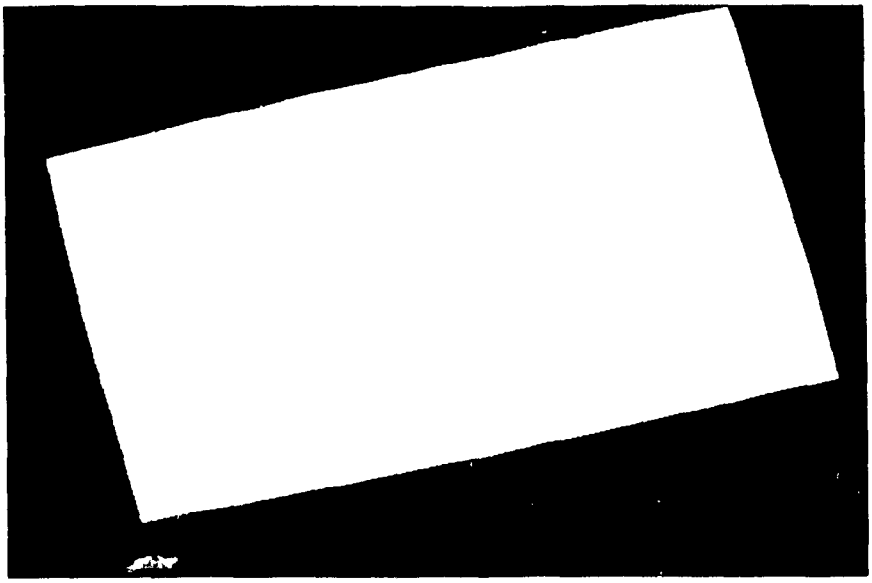
Once the four corners are found, the length, width, the centroid and the orientation angle can be easily obtained

Length	$L = ((x1-x4)^2 + (y1-y4)^2)^{1/2}$	(5.1)
Width	$W = ((x1-x2)^2 + (y1-y2)^2)^{1/2}$	(5.2)
Centroid	$cx = (x1 + x2 + x3 + x4) / 4$	(5.3)
	$cy = (y1 + y2 + y3 + y4) / 4$	
Angle	$\alpha = Arctan ((x1 - x2) / (0.75 * (y2-y1)))$	(5.4)

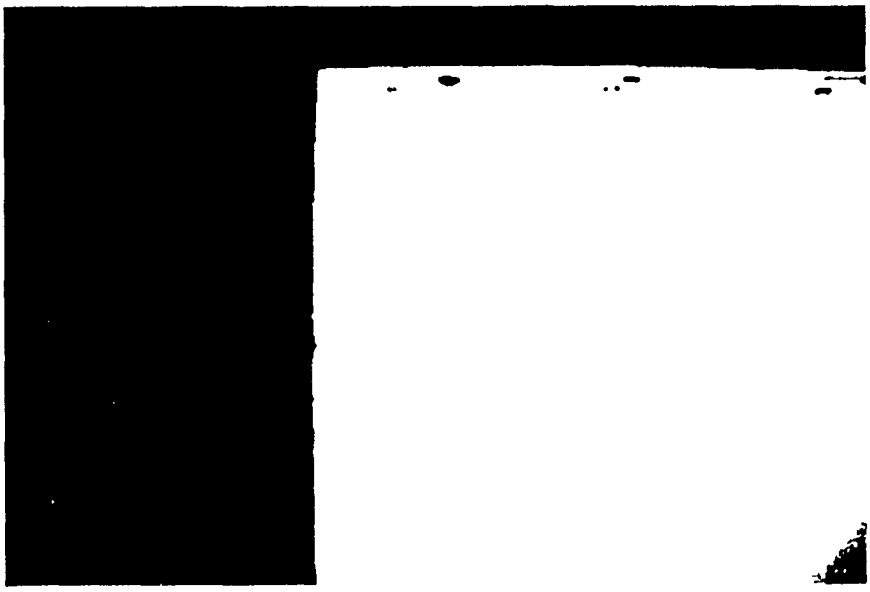
Notice that, when calculating the orientation angle, a value of 0.75 is multiplied to (y2-y1) because of the fact that the screen is a rectangular of 640 x 480 instead of a square One of the output of the vision system is shown in Figure 5.4.

5.2 Robot System

The G3211-W330 Gantry Robot System (GRS) is a modular, servo driven, gantry system which can be configured to work with 2 to 8 axes of motion, which is controlled by the RAPL II programming language. The work space of this GRS is 6 x 4 x 2 feet in X, Y , and Z direction. The gantry is fitted with a W330 industrial robot wrist (Table 5.3) and controlled by a C500 robot controller (Table 5.4). A vacuum pump is attached to the wrist (Table 5.5) [9]



a Top View of a Box



b. Side View of a Box

Figre 5.4 Vision System Output

Table 5.3 W330 Industrial Robot Wrist

Performance Specifications

Payload	• joint maximum torque ratings: joint 1: 30 inch pound joint 2. 30 inch pound joint 3 25 inch pound
Working Envelope	• 10.5 inch reach without gripper (267mm) • joint angles: joint 1 +180°/-180° joint 2: +105°/-105° joint 3: +180°/-180°
Speed	• joint speed at 100 %: joint 1: 180 degrees per second joint 2: 180 degrees per second joint 3: 180 degrees per second
Repeatability	• +/- 0.002 inches (+/- 0 05 mm) • +/- 0.20 inches accuracy (+/- 0.51mm)

Mechanical Features

Configuration	•modular, Euler-type wrist design with three degrees of freedom
Drive	•DC servo motors with optical encoders
Transmission	•low vibration, high performance harmonic drives optimized for robotic applications on all joints
Gripper Support	•a vacuum gripper is supported by an external venturi vacuum generator
Homing Sequence	•homing procedure uses limit switches
Connections	•mass termination for all signals
Controller/Software	•W330 is compatible with CRS PLUS controllers and RAPL II
Wrist Weight	•10 pounds (4.5 Kg)

Table 5.4 C500 Robot Controller

Configuration	•self-contained robot controller •automatic program execution for hands-free operation
Programming Language	•RAPL-II multi-task operating system with PCP support
Co-ordinate System	•motor, joint, world, and remote reference frames
Control Type	•PID
Velocity Profiles	•trapezoidal or parabolic in Cartesian co-ordinates •trapezoidal or parabolic in joint angle co-ordinates

Table 5.4 C500 Robot Controller (cont'd)

Processors	•Intel 80286 and 80287
Transputers	•20 MHz T400 for standard servo control, 30 MHz T425 for high performance requirements and 30 MHz T805 kinematics co-processor
Transputer Link	•all links run at speed up to 20 Mbits/sec
No. of Servo Axes	•8 axis
Axis Inputs (TTL)	•home switches •+/- travel limits switches •motor thermal switches
Feedback	•differential encoder with zero cross •third order digital filtering of all encoder signals •a msec closed loop position control •single processor for 8 axis control •pulse rate 200 KHz
Motor Command Signals	•+/- 10 volt motor amplifier control •12 bits conversion resolution
Teaching Methods	•LIMP mode (lead-by-the-hand), off-line and teach pendant
Kinematics	•standard kinematics for each of the following robot system Articulated (5 and 6 axis), Gantry (5 and 6 axis) and XYZ (3 axis)
Path Types	•joint interpolated (point-to-point), straight line, continuous path and relative motion paths
Communication	•dual, fully programmable, RS232 ports, 38 4K baud max •ACI protocol supported on device 1 providing a master/slave network supporting up to 127 slaves •full access to all controller parameters (over 600) available with the optional ROBCOMM-II or ACI tool-kit software packages for custom host computer software interface or diagnostics
Input/output Support	•16 TTL inputs scanned at 50 Hz •16 TTL outputs scanned at 50 Hz •all TTL inputs and outputs can be interfaced with 12 VDC, 24VDC, 110 VAC, and 220 VAC opt-isolators (optional)
Memory	•64K bytes of user battery backed RAM memory provides the following maximums expandable to 256K bytes, 1365 locations, or 5734 program lines, or 4096 variables •512K system FLASH memory
Temperature Range	•+50° to +104°F (+10° to +50°C)
Power Requirements	•120/220 VAC, 60/50 Hz, 350 VA •power circuit uses isolation transformer, doubled-poled power switch and fuses

Table 5.4 C500 Robot Controller (cont'd)

Teach Pendant	
Configuration	•hand held with 10' cable
Controls	•manual operation of all robot joints
	•gripper control
	•teach button for storing current location of robot arm
	•program -abort button
	•LIMP control (on/off)
	•ALIGN control to nearest major plane
	•Emergency Stop push button
Displays	•4 line x 20 character LCD

Table 5.5 Vacuum Pump 3N401-VAC

Pressure Range	•15 to 80 PSI
Flow Rate	•@ 80PSI: 0.3 CFM
Payload	•12 lbs (life at 70 PSI)

The robot is controlled by programs written in RAPL II through C500 controller. RAPL-II programs can be edited by ROBCOMM-II software package. Robcomm-II includes a terminal emulator for direct ASCII communication to robot's RAPL-II operating system. It uses the ACI (Advanced Communication Interface) to achieve reliable communication to C500 controller. RAPL-II programs are saved in the controller and can be executed either from Robcomm-II or any high level programming languages by interfacing any IBM PC/XT/AT/PS-2 or compatible computer with the controller through the serial RS232C port.

To perform the palletization task, a Robot program has been developed. The program mainly performs to control the robot to pick up a box either from the conveyor

belt or from the holding area and load it on the pallet, or to remove a box from the pallet to the holding area. The program receives information about box dimensions, orientations, and box location from the main program which gathers the information from vision subroutine. In the program, four positions are defined PIUP, PICKUP, UNLD, and UNLOAD. When a motion command is sent from the controller, the robot is programmed to move towards point PIUP at a relatively high speed, then it approaches point PICKUP at a lower speed and opens the vacuum pump. After it grabs the box, it moves to point UNLD via PIUP at the high speed, and approaches UNLOAD with low speed and releases the box by closing the vacuum pump at point UNLOAD

5.3 Conveyor System

Table 5.6 Conveyor System

Conveyor belt	
Length	• 5 feet
Width	• 2 feet
Linear Speed Range	• 0 - 0.5 ft/s
Control	• Pendant/Robot Controller
Modes of Operation	• ON/OFF/Controlled
Light Sensors	
Indicators:	
Yellow	• Power On
Green	• Output Energized
Red	
Steady On	• 2.5X Margin
Flashing	• Short Circuit/Overload
Field of View	• 1.5°
Ambient Temp. Range	• -34°C to 70°C
Transmitting LED	• Visible Red, 660nm
Relative Humidity	• 5-95%
Lens Material	• Plastic

An industrial conveyor is selcted for the RPS. Two light sensors were installed at the side of the conveyor, one at the measuring place, another at the pick up place (See Table 5.6) Both conveyor and sensors are controller by C500 controller. The signal from the proximity light sensors is read from COM2 serial port. After the decision is made, commands id send to conveyor, again through COM2, either to stop or to run the convey

5.4 System Integration

The robot, vision system, conveyor are integrated by a computer to perform the automated palletization task. Figure 5.5 shows this integration When a box from the conveyor comes into the grabbing frame of the vision system, the sensor on the conveyor belt will send a signal to controller which stops the conveyor for the vision system taking the image of the box After the task of the vision system is done, the computer sends a command back to the controller to resume the motion of the conveyor. The box is then moved to the pick-up place and the conveyor is stopped again when the light sensor detects the box Meanwhile, the main program applies the heuristic of Model 3 to determine where this box will go. When the box arrives at the pick-up area, the computer sends information about the boxes including the new values of PIUP, PICKUP, UNLD, UNLOAD to the controller which controls the motion of the robot to move the box either to the holding area or on to the pallet.

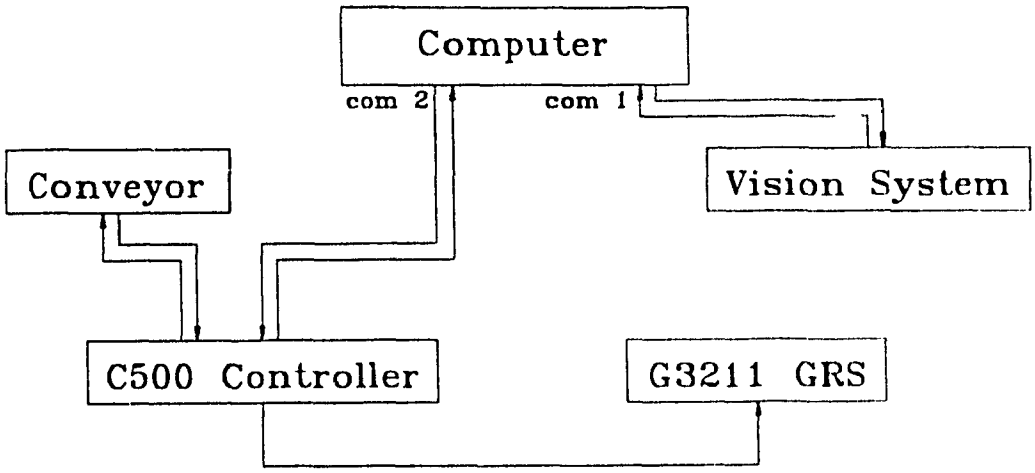


Figure 5.5 Diagram of System Integration

Boxes used in the implementation have dimensions shown in Table 7.4. Ten functions are written in "C" to interpret the heuristic of Model 3. The syntax and the usage of the functions are explained as follows

block20_form()

Format: void block20_form(void)

Parameters none

Description: This function does the block formulation for group 2.

Return value none

block30_form()

Format: void block30_form(void)

Parameters none

Description This function does the block formulation for group 3.

Return value nothing

fill_layer()

Format void fill_layer(void)

Parameters none

Description This function performs filling a layer with selected boxes

Return value nothing

load_box()

Format void load_box(void)

Parameters none

Description This function determines where and how the selected box should be put on the pallet. The available areas on the pallet are located first by scanning the 0 value in the pallet matrix. Then the orientation of the box on the pallet is determined by this rule: if the value W/w is an integer, the box' width should parallel to pallet's width; if the value W/l is an integer, the box's length should parallel to the pallet's width. The loading starts from the left-top corner of the pallet, first from top to bottom, and then from left to right.

Return value nothing

new_layer()

Format: void new_layer(void)

Parameters none

Description: This function determines the height for a new layer

Return value: nothing

put_box()

Format: void put_box(xx, yy, ww, ll)

Parameters (xx, yy) is coordinate of the box's left-top corner on the pallet

ww is the box width

ll is the box length

Description: Fill the pallet with the box selected. The pallet is simulated by a matrix of (48x40). It is initialized by give all the elements with value 0 When a box is loaded on the pallet, the elements of the portion which the box covers are give to the value of 1.

Return value: nothing

remove_box()

Format: void remove_box(void)

Parameter none

Description: This function removes the smaller boxes from the pallet by putting 0 value to the corresponding portion in the pallet matrix

Return value nothing

select_box1()

Format void select_box1(void)

Parameters none

Description This function does the selection of boxes to be loaded on to the pallet from group 1

Return value nothing

select_box2()

Format void select_box2(void)

Parameters none

Description This function does the selection of boxes to be loaded on to the pallet from group 2

Return value nothing

select_box3()

Format void select_box3(void)

Parameters none

Description This function does the selection of boxes to be loaded on to the pallet from group 3

Return value nothing

Operating Experience

A sequence of boxes is used here to illustrate the overall system operation Box 1 coming into the system is a Type 1 box Box 2 is a Type 7 box Since Type 7 has a height of 30, the height of first layer is selected as 30 Next is to fill this layer with boxes/blocks with height of 30. Since there is only one box with height of 30 now, more boxes need to be generated Move Box 1 to the holding area, and the conveyor sends Box 2 to the pick-up place and Box 3 is under the cameras now, which again is a Type 1 box. Move Box 2 to the holding area, and the Box 4 comes into the system, which is a Type 4 box Then Box 3 is moved to the holding area and Box 5 comes which is a Type 4 box again Now, there are two Type 1 boxes, and one Type 7 box in the holding area, a Type 4 box at the pick-up place and another Type 4 under the vision One Type 1 box and twe Type 4 boxes make up one Type 8 block Compare Type 8 block and Type 7 box, the Type 8 block is selected to load onto the pallet Move the two boxes on the conveyor and one of the Type one boxes in the holding area onto the pallet Box 6 comes, it is a Type 7 box Move it to the pick-up place and Box 7 comes which is a Type 1 box. Now, there are two Type 7 boxes. The one on the conveyor is loaded onto the pallet Conveyor transports Box 7 to the pick-up place and new box is a Type 2 box (Box 8). There are a Type 1 box and a Type 2 box on the conveyor now and they can form a Type 8 block. Now, there are 2 Type 1 boxes, one Type 2 box, and one Type 7 box. One Type 1 box and one Type 2 box form one Type 8 box However, this Type 8 block is too big for the remaining area The logic checks the boxes on the pallet and finds that the layer can be filled if the Type 7 box is replaced by block 8 So, the Type

7 box is removed from the pallet and put back to the holding area. And the Type 1 box at the pick-up place is loaded After, the Type 2 box is loaded too. The layer utilization reaches 100%. Now the first layer is completed. Since the maximum palletization height is 40, the height for next layer must be 10. The boxes in the holding area are one Type 1 and 2 Type 7s. Type 1 box then is moved onto the pallet. Meanwhile, two new boxes come into the system Box 9 is a Type 5 and Box 10 is a Type 1 box. Box 9 is moved to the holding area and Box 10 is loaded on the pallet. At this point, the second layer reaches 100% utilization and the remaining palletization height is 0. Therefore the palletization is completed. The physical layout of boxes on the pallet throughout the loading process is shown in Figure 5.6.

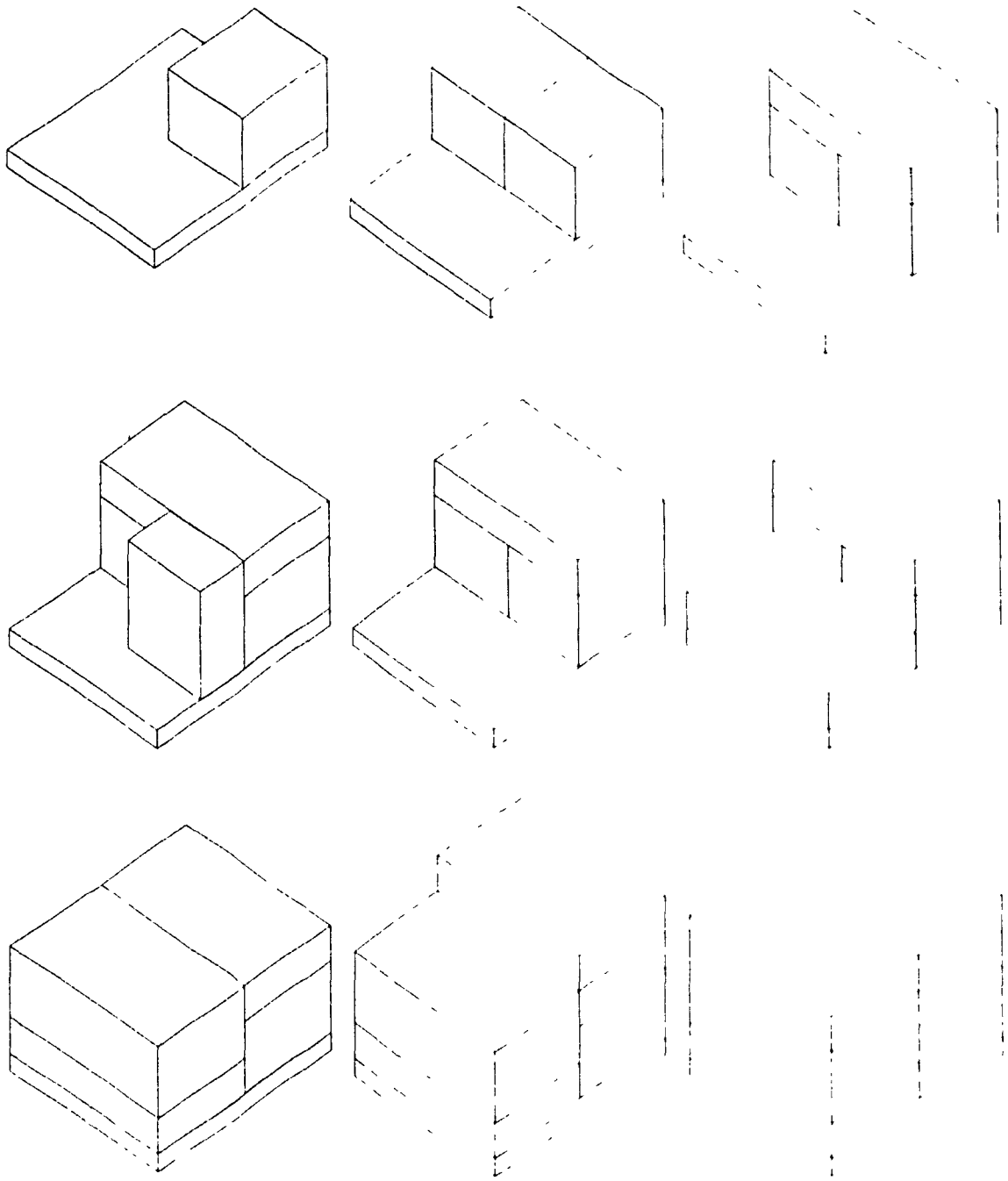


Figure 5.6 Physical Layout of the Palletization Pattern

CHAPTER 6 CASE STUDY APPLICATIONS

Some numerical examples have been tried to demonstrate the proposed models. Three types of boxes with certain box availability are selected for Model 1. Model 2 and Model 3 used seven types of boxes which have the same lengths and widths as those for Model 1 but have different height. For Model 2, the number of each type of boxes is limited to two, while for model 3 there is no limit on the box availabilities.

6.1. Case Study for Model 1

An illustration problem is to be solved here by using the proposed ILP model. In order to compare later on, the input data, as shown in Table 6.1, is taken from Abdou and Lee [1]

Table 6.1. Input Data

Item	Number	Length	Width
Pallet	1	48	40
Box Type 1	2	40	24
Box Type 2	6	24	20
Box Type 3	4	20	12

Since boxes could be arranged in two different orientations, each type of box is converted into two copies The new boxes are shown in Table 6.2.

Table 6.2. Converted Boxes Data

Box Type	Number of Box	Length	Width
Type 1	X1	40	24
Type 2	X2	24	20
Type 3	X3	20	12
Type 4	X4	24	40
Type 5	X5	20	24
Type 6	X6	12	20

where X1+X4=2; X2+X5=6; X3+X6=4.

First, the objective function needs to be formulated. For this particular problem, N=3, and the number of layers of palletization K is assumed to be three

$$Max\ P=\sum_{j=1}^{3}\sum_{i=1}^{6}\alpha_i X_{ij}+\sum_{j=2}^{3}(Y_1-Y_j) \qquad (6.1)$$

After the objective function is determined, the constraints can be set. The first group of constraints is about pallet area.

$$L\times W-\sum_{i=1}^{6}\alpha_i X_{ij}\geq 0; \qquad \forall j \qquad (6.2)$$

The second set of constraints is about box quantities. Each original box type has 2, 6 and 4 boxes respectively. Considering the fact that box Type 1 is same as Type 4, Type 2 is same as Type 5 and Type 3 is same as Type 6, we have

$$\sum_{j=1}^{3}X_{1j}+\sum_{j=1}^{3}X_{4j}\leq 2 \qquad (6.3)$$

$$\sum_{j=1}^{3}X_{2j}+\sum_{j=1}^{3}X_{5j}\leq 6 \qquad (6.4)$$

$$\sum_{j=1}^{3} X_{3j} + \sum_{j=1}^{3} X_{6j} \leq 4 \qquad (6.5)$$

The stability constraints are given by limiting the number of boxes and area of the layer on the top less or equal to the one underneath.

$$\sum_{i=1}^{6} \alpha_i X_{ij} \geq \sum_{i=1}^{6} \alpha_i X_{ij+1}; \qquad j=1,2. \qquad (6.6)$$

$$\sum_{i=1}^{6} X_{ij} = Y_j; \quad Y_j \geq Y_{j+1}; \qquad \forall j \qquad (6.7)$$

The procedure to generate strip constraints is applied as follows:

(1) divide the pallet of 48x40 into 40 unit width strips of 48x1

(2). find out the box length set which is S=(40, 24, 20, 12)

(3). find out all possible combinations of box lengths, which could fit the pallet length of 40. There are 14 combinations found as follows;

a. (40)

b (24, 24)

c. (24, 20)

d. (24, 12, 12)

e (24, 12)

f. (24)

g (20, 20)

h. (20, 12 ,12)

i. (20, 12)

j. (20)

k. (12, 12, 12, 12)

l. (12, 12, 12)

m. (12, 12)

n. (12)

(4). Let A_j, B_j,..., N_j donate the number of unit width strips to be selected on layer j corresponding to the box length combination a, b, .., n. Since the maximum width for each layer cannot exceed the pallet width 40, therefore the following constraints are set:

$$A_j+B_j+C_j+D_j+E_j+F_j+G_j+H_j+I_j+J_j+K_j+L_j+M_j+N_j \leq 40; \qquad \forall j \ (6.8)$$

For length of 12, there is only one box type, Type 6, which has a width of 20

$$2D_j+E_j+2H_j+I_j+4K_j+3L_j+2M_j+N_j \geq 20X_{6j}; \qquad \forall j \ (6.9)$$

For length of 20, there are two box type. Type 3 boxes have a width of 12, and Type 5 boxes have a width of 24

$$C_j+2G_j+H_j+I_j+J_j \geq 24X_{5j}+12X_{3j}; \qquad \forall j \ (6.10)$$

For length of 24, there are two box type as well. Type 2 boxes have a width of 12, and Type 4 boxes have a width of 40

$$2B_j+C_j+D_j+E_j+F_j \geq 40X_{4j}+12X_{2j}; \qquad \forall j \ (6.11)$$

For length of 40, there is only one box type, Type 1, which has a width of 24

$$A_j \geq 24X_{1j}; \qquad \forall j \ (6.12)$$

The ILP model for this problem can be formulated now (see next page)

$$Max\ P=\sum_{j=1}^{3}\sum_{i=1}^{6}\alpha_i X_{ij}+\sum_{j=2}^{3}(Y_1-Y_j)$$

Subject To:

$$L\times W-\sum_{i=1}^{6}\alpha_i X_{ij}\geq 0;\qquad \forall j$$

$$\sum_{j=1}^{3}X_{1j}+\sum_{j=1}^{3}X_{4j}\leq 2$$

$$\sum_{j=1}^{3}X_{2j}+\sum_{j=1}^{3}X_{5j}\leq 6$$

$$\sum_{j=1}^{3}X_{3j}+\sum_{j=1}^{3}X_{6j}\leq 4$$

$$\sum_{i=1}^{6}\alpha_i X_{ij}\geq\sum_{i=1}^{6}\alpha_i X_{ij+1};\qquad j=1,2.$$

$$\sum_{i=1}^{6}X_{ij}=Y_j;\qquad Y_j\geq Y_{j+1};\qquad \forall j$$

$$A_j+B_j+C_j+D_j+E_j+F_j+G_j+H_j+I_j+J_j+K_j+L_j+M_j+N_j\leq 40;\qquad \forall j$$

$$2D_j+E_j+2H_j+I_j+4K_j+3L_j+2M_j+N_j\geq 20_{6j};\qquad \forall j$$

$$C_j+2G_j+H_j+I_j+J_j\geq 24X_{5j}+12X_{3j};\qquad \forall j$$

$$2B_j+C_j+D_j+E_j+F_j\geq 40X_{4j}+12X_{2j};\qquad \forall j$$

$$A_j\geq 24X_{1j};\qquad \forall j$$

X_{ij}, Y_{ij} and $A_j, B_j, \ldots, N_j$ are positive integers

For this particular problem, the optimum value of objective function is 5760 (APPENDIX A). This value consists of a total volume of 5760 for pallet utilization and 6 for stability index. Thus, in this case, the pallet utilization is a 100%. The values of the decision variables are as follows:

Layer 1 $X_{11} = 0$ $X_{21} = 0$ $X_{31} = 0$ $X_{41} = 2$ $X_{51} = 0$ $X_{61} = 0$

Layer 2 $X_{12} = 0$ $X_{22} = 4$ $X_{32} = 0$ $X_{42} = 0$ $X_{52} = 0$ $X_{62} = 0$

Layer 3 $X_{13} = 0$ $X_{23} = 0$ $X_{23} = 2$ $X_{43} = 0$ $X_{53} = 0$ $X_{63} = 4$

The optimal physical layout of the boxes on the pallet for the three different layers is shown in Figure 6.1.

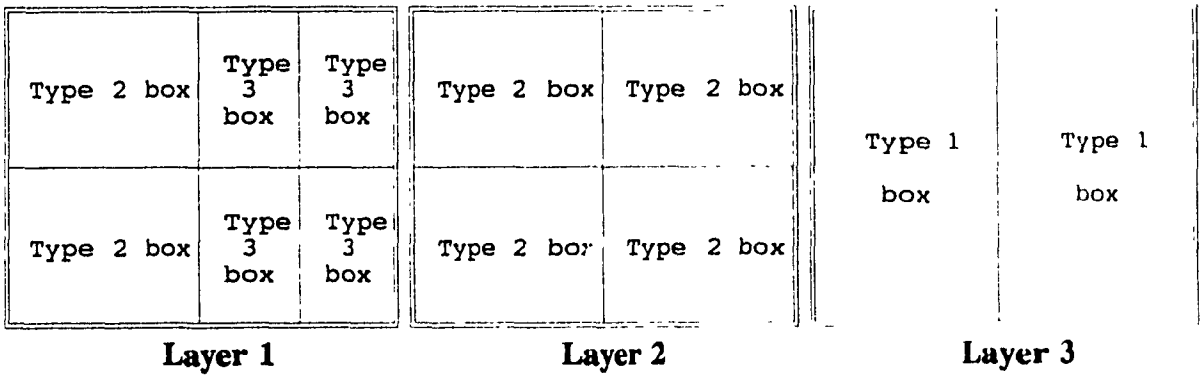


Figure 6.1 Physical layout of multi-layer palletization

6.2 Case Study for Model 2[2]

The boxes used in the example for Model 2 have the same lengths and widths as the boxes used in the example for Model 1. However, these boxes have different heights, which are limited to 10, 20, and 30. There are totally seven types of boxes and each type has two boxes. The pallet is the same. But the palletization height is set as 40. The specific data of these boxes and the pallet are shown in Table 6.3.

Table 6.3 Original Type Of Boxes

Box Type	Length	Width	Height	Volume	No. of Box
1	40	24	10	9600	2
2	40	24	20	19200	2
3	24	20	10	4800	2
4	24	20	20	9600	2
5	24	20	30	14400	2
6	20	12	20	4800	2
7	20	12	30	7200	2
Pallet	48	40	40	76800	1

The process of solving this problem is as follows:

First, the objects are changed from boxes into blocks: eight blocks are defined at this stage. Seven of them have the same dimensions as the seven types of boxes. The other has the biggest dimensions from the box length set, the width set and the height set. The dimensions of these blocks are shown in Table 6.4

Table 6.4. Type of Blocks

Block Type	Length	Width	Height
1	40	24	10
2	40	24	20
3	24	20	10
4	24	20	20
5	24	20	30
6	20	12	20
7	20	12	30
8*	40	24	30

[Note: there is no box which could make up block 8 by itself]

Second, find out the block formulation. For this particular problem, the possible block formulations are as follows:

Table 6.5 Block Formulation

h=30	h=20	h=10
B8*= B1+B2 = B1+B4+2B3 = B1+B4+2B6 = B1+2B3+2B6 = B1+4B3 = B1+4B6 = B1+2B1 = 3B1 = 2B1+2B3 = B1+4B3	B6 = X6 B4 = X4 = 2B3 = 2B6 B2 = X2 = 2B1 = 2B4 = B1+2B3	B3 = X3 B1 = X1 = 2B3
B7 = X7		
B5 = X5 = X3+B4 = B3+2B6 = B3+2B3 = 3B3		

(where Bi stands for the number of block i available)

The initial block availability is obtained by formulating blocks in such a way that one box forms one block. Thus, the initial block availability is: B1 = B2 = B3 = B4 = B5 = B6 = B7 = 2, and B8 = 0.

At next step, all possible layer combinations need to be found. In this example, H_{max} is considered to be 40 inches. Let a donate the number of layers of height of 10, b the number of layers of height of 20 and c the number of layers of height of 30. Then;

$$a*10 + b*20 + c*30 \leq 40.$$

Solving this inequality, only four feasible solutions of layer combinations for this problem are found:

Layer Combination 1. a=4. b=0, c=0: 10 + 10 + 10 + 10

Layer Combination 2. a=2, b=1, c=0: 20 + 10 + 10

Layer Combination 3. a=0, b=2, c=0: 20 + 20

Layer Combination 4. a=1, b=0, c=1: 30 + 10

Now each layer combination becomes a sub-problem. Next is to solve each problem of every layer combination, starting with the initial block availability.

Layer Combination 1. (10 + 10 + 10 + 10)

Since all the heights are same, this is Type 2 problem which could be solved by the ILP model proposed earlier. Only blocks with height of 10 will be used, they are: B1 = B3 = 2. The result is shown in Table 6.6.

Table 6.6 Solution for Layer Combination 1

Layer	Block Combination	Utilization
1	(1)B1 + (2)B3	100%
2	(1)B1	50%
Total		37 5%

where B1 = X1 and B3 = X3.

Since all boxes with height of 10 are selected, this is the optimum solution to the layer combination 1. This result is kept and the algorithm starts to solve the layer combination 2.

Layer combination 2, (20 + 10 + 10)

The input to this sub-problem includes blocks with height of 10 and 20 They are B1 = B2 = B3 = B4 = B6 = 2. Since there are only two heights, the solution to the problem, as shown in Table 4, has two parts: one layer with a height of 20, and another 2 layers with a height of 10.

Table 6.7 Solution for Layer Combination 2

For Height of 20		
Layer	Block Combination	Utilization
1	(2) B2	100%
For Height of 10		
Layer	Block Combination	Utilization
1	(1) B1 + (2) B3	100%
2	(1) B3	50%
Total		87 5%

The layer of height of 20 already reaches 100% utilization and all boxes with height of 10 have been selected Therefore, this is the optimum solution to this layer combination as well

Layer Combination 3, (20 + 20)

This is also Type 2 problem since the height of each layer is 20 The initial input is B2 = B4 = B6 = 2 The solution to this layer combination is shown in Table 7.8.

Table 6.8 Solution for Layer Combination 3

Layer	Block Combination	Utilization
1	(1)B2 + (1)B4 + (2)B6	100%
2	(1)B2 + (1)B4	75%
Total		87 5%

Since this solution does not reach 100% utilization, and there are other ways to built up block 2 and block 4, another alternative of block formulation is generated, by increasing the number of blocks B2 and B4 to 3 (2 X1 make another B2 and 2 X3 make another B4) The solution based on this block availability is shown in Table 6 9.

Table 6.9 Solution for Layer Combination 3 Based on New Block Formulation

Layer	Block Combination	Utilization
1	(1)B2 + (2)B4	100%
2	(2)B2	100%
Total		100%

[Note 3B2 = 2X2 + 2X1, 3B4 = 2X4 + 2X3]

Now, the utilization is 100%. This result is kept as the solution to the problem of layer combination 3.

Layer Combination 4. (30 + 10)

The initial input for this combination is: B1 = B3 = B5 = B7 = 2, and B8 = 0. The result is shown in Table 6.10.

Table 6.10 Solution for Layer Combination 4

For Height of 30		
Layer	Block Combination	Utilization
1	(2) B5 + (2) B7	75%
For Height of 10		
Layer	Block Combination	Utilization
1	(2) B1	100%
Total		81.25%

Same as the layer combination 2, the algorithm has to choose another way to formulate blocks of height of 30 for other block availability since the result is not satisfactory.

Build the biggest block using boxes with height of 20 and boxes with height of 10 (except the two Type 1 boxes, they have been selected for layer of 10). One block 8 is built by one Type 2 box and two Type 3 box. The new input is: B1 = 2, B3 = 0, B5 = 2, B7 = 2, B8 = 1. The result from this input is shown in Table 6.11.

Table 6.11 Solution for Layer Combination 4 Based on New Block Formulation

For Height of 30		
Layer	Block Combination	Utilization
1	(1) B8 + (2) B5	100%
For Height of 10		
Layer	Block Combination	Utilization
1	(2) B1	100%
Total		100%

The utilization reached 100% from this input. Therefore, this result is kept as the solution to the Layer Combination 4

After all the sub-problems are solved, the selection process could begin. Next step is to choose the best solution from the four layer combinations based on comparison about pallet utilization. At this stage, Layer Combination 1 and Layer Combination 2 are eliminated because of lower utilization.

Next is to perform the comparison based on stability criteria. However, before doing this, the layer orders for the remaining Layer Combination 3 and Layer Combination 4 must be determined since the layer order plays a very important role in stability measurement There could be several ways to put boxes in a block, put blocks on a layer and put layers on the pallet. There are six possible different layouts for the Layer Combination 3 (see Figure 6.2) But only one with the best stability is selected, which is layout (a) (See Appendix F)

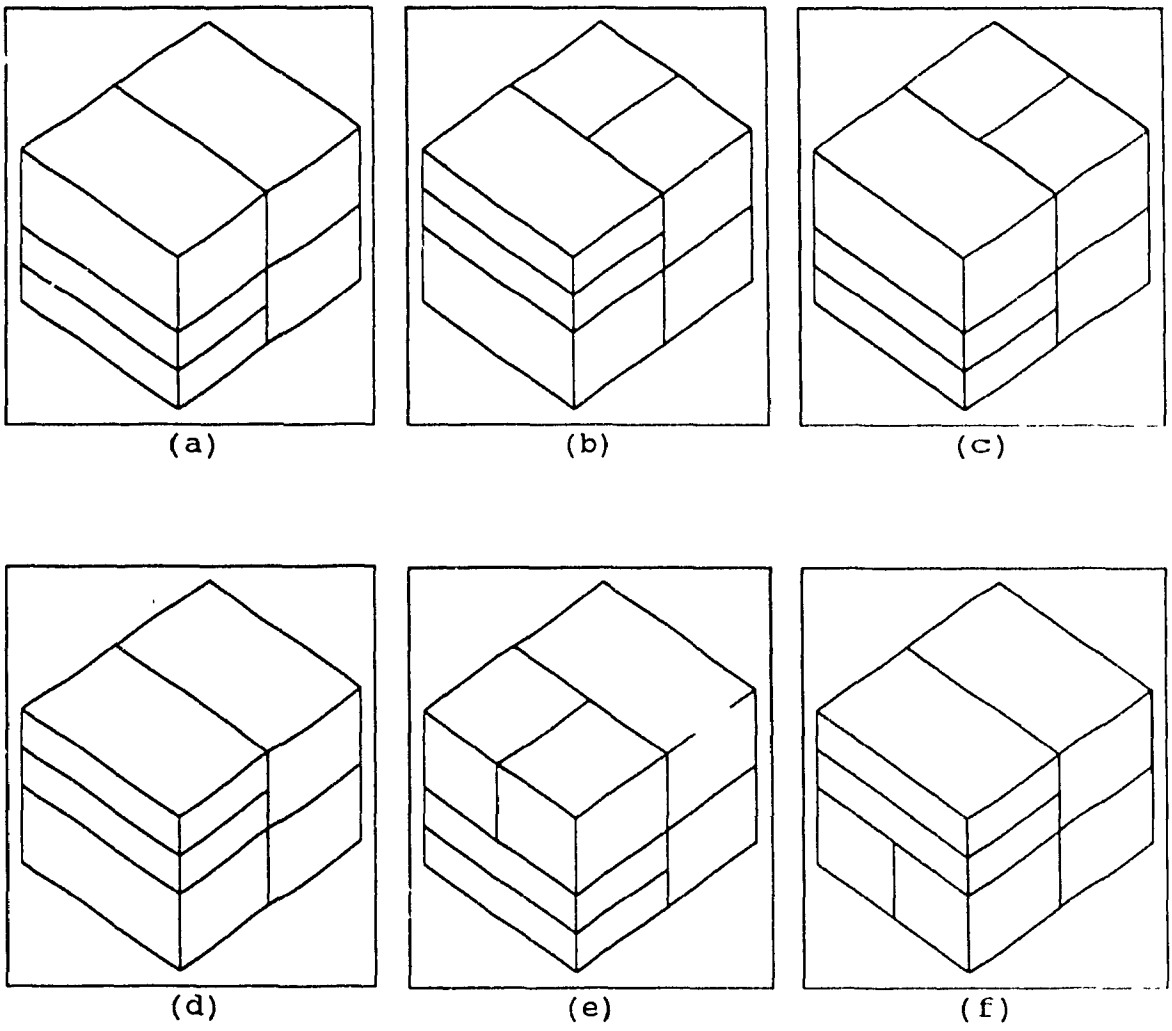


Figure 6.2 Possible Pallet Layouts for Layer Combination 3

Similarly, the best layout for the Layer Combination 4 is found from four possible layouts, which is layout (a) in Figure 6.3.

Since the two solutions have the same stability value, the comparison about WIP must be conducted. In this case, these two layer combinations give the same value on WIP as well. Therefore, they are alternative solutions to the problem. This 3-level comparison is shown in the Table 6.12. Figure 6.4 shows the two alternatives.

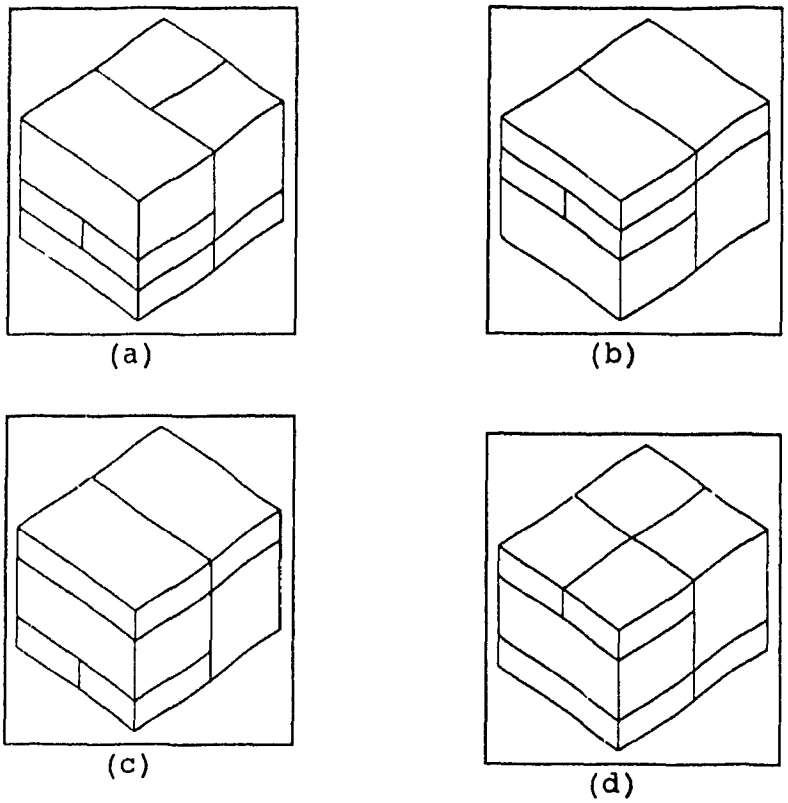


Figure 6.3 Possible Pallet Layouts for Layer Combination 4

a. Alternative 1 **b. Alternative 2**

Figure 6.4 Solutions to the 3D palletization problem

Table 6.12 Selection of Best Solution

A	Boxes selected	Utilization	Stability	WIP
1	(2)X1, (2)X3	37.5%	---	---
2	(2)X1,(2)X2,(2)X3	87.5%	---	---
3	(2)X1,(2)X2,(2)X4	100%	9600	2880
4	(2)X1,(1)X2,(2)X3,(2)X5	100%	9600	2880

6.3 Case Study for Model 3

The Model 3 has been developed for the physical RPS considering the random box-coming sequences. In order to show the effect of random incoming sequence of boxes on the palletization, three different simulations have been conducted for Model 3. Each simulation program simulates the main program used for the system integration which is described in Chapter 5. In order to simulate the conveyor and the vision system, a function called **new_box()** has been developed which uses a random number generator to simulate the random coming boxes

new_box()

Format: void new_box(void)

Parameter none

Description This function generates a random box

Return value nothing

The boxes used in the simulation are the same types of boxes used in the example for Model 2 (see Table 6.4). Seven types of boxes of infinite number fall into three groups

Group 1 (height of 10): Type 1 and Type 3

Group 2 (height of 20): Type 2, Type 4, and Type 6

Group 3 (height of 30): Type 5 and Type 7

Three sets of simulations have been conducted with different inputs.

- Simulation 1: the same input as the example for Model 2; that is, Type 1 boxes to Type 7 boxes, and two boxes for each type.
- Simulation 2: one of the outputs from the example for Model 2, which is: two Type 1 boxes, one Type 2 boxes, two Type 3 boxes and two Type 5 boxes.
- Simulation 3: Type 1 to Type 7 boxes with no limits on the number of boxes available

Each simulation of Model 3 is run for ten different sequences. Pallet utilization, WIP, and Palletization time are used for the comparison. Pallet utilization is measured by formula ***(3.1)*** given in Chapter 4. Maximum number of boxes in the holding area is taken as the index for WIP. The palletization time is calculated by using formula ***(3.3)***:

$$T = T_{ch} * MCH + T_{cp} * MCP + T_{hp} * MHP + T_{ph} * MPH$$

When the robot moves at speed of 4"/sec, each motion takes about 40 seconds. Therefore

the palletization time can be calculated by formula ***(6.13)***

$$T=40*(MCH + MCP + MHP + MPH) \qquad (6.13)$$

The outputs of the three simulations are shown in Table 6.13, Table 6.15, and Table 6.17, while the robot movement frequency and the palletization time are summarized in Table 6.14, Table 6.16, and Table 6.18 correspondingly

Table 6.13 Summary of Outputs from Simulation 1

No	Sequence	TIME (Seconds)	Maximum WIP	Utilization
1	1-3-2-1-3-4-4-5-6-2-6-7-5	680	6	100%
2	1-3-4-4-1-7-2-7-3-2	480	3	100%
3	7-3-4-1-5-5-1-3-7	480	4	100%
4	6-4-5-5-6-3-4-3-1-7-2-7-2	600	5	100%
5	5-4-7-2-2-4-6-1-6-3-1-7-3-5	760	6	94.25%
6	5-5-7-7-2-2-6-6-4-3-4-1-3	880	8	100%
7	5-5-3-4-7-2-7-2-4-6-6-3-1-1	640	6	100%
8	3-6-5-5-2-3-6-4-7-4-7-1-2-1	600	8	100%
9	3-6-1-1-4-2-3-2-6-4-7-5	560	4	100%
10	2-6-5-2-3-7-7-6-5-4-4-3	560	5	100%

Table 6.14 Robot Movement Frequency and Palletization Time:Simulation 1

Sequence No	MCH	MCP	MHP	MPH	T
1	10	3	4	0	680
2	6	5	1	0	480
3	4	5	3	0	480
4	7	6	2	0	600
5	11	5	3	0	760
6	8	6	2	1	680
7	8	6	2	0	640
8	8	5	2	0	600
9	7	6	1	0	560
10	6	6	2	0	560

Table 6.15 Summary of Outputs from Simulation 2

No.	Sequence	TIME (Seconds)	Maximum WIP	Utilization
1	1-3-2-3-5-1-5	440	3	100%
2	5-5-3-3-2-1-1	360	2	100%
3	5-5-1-3-1-2-3	400	3	100%
4	5-5-1-3-3-1-2	400	3	75%
5	1-3-1-2-5-5-3	400	3	100%
6	1-5-1-2-5-3-3	360	2	100%
7	2-3-5-5-1-3-1	400	3	100%
8	3-3-2-5-1-5-1	440	2	100%
9	3-3-2-1-1-5-5	400	2	100%
10	3-3-5-1-1-2-5	400	3	75%

Table 6.16 Robot Movement Frequency and Palletization Time:Simulation 2

Sequence No.	MCH	MCP	MHP	MPH	T
1	4	3	4	0	440
2	2	5	2	0	360
3	3	4	3	0	400
4	4	3	3	0	400
5	4	3	3	0	400
6	2	5	2	0	360
7	3	4	3	0	400
8	4	3	4	0	440
9	3	4	3	0	400
10	4	3	3	0	400

Table 6.17 Summary of Outputs from Simulation 3

No	Sequence	TIME (Seconds)	Maximum WIP	Utilization
1	1-7-1-4-4-7-1-2-5-1	560	3	100%
2	3-5-1-6-5-3-3-2-5-5-2-3	680	5	100%
3	4-4-1-7-5-4-3-1-5-1-1-3	600	5	100%
4	6-3-5-1-1-7-5-2-6-5-4-6-4-5-6-1	760	10	100%
5	1-7-1-4-4-7-1-2-5-1	680	3	100%
6	7-4-3-4-2-1-2-2-4-7-3-4-4-2-2-3-7-7-4-1	880	12	100%
7	1-3-5-4-3-5-4-6-5-3-4-1	680	4	100%
8	2-3-2-2-7-2	280	2	100%
9	3-2-7-6-4-1-4-3-2-2-6-3-6-3-2-2-4-6-6-3-1	880	11	100%
10	3-5-3-2-1-4-6-7-5-5-4-7-7-1	680	7	100%

Table 6.18 Robot Movement Frequency and Palletization Time:Simulation 3

Sequence No	MCH	MCP	MHP	MPH	T
1	4	6	3	1	560
2	9	3	5	0	680
3	6	6	3	0	600
4	13	3	3	0	760
5	4	6	4	1	680
6	13	7	2	0	880
7	8	4	5	0	680
8	3	3	1	0	280
9	13	7	2	0	880
10	10	4	3	0	680

CHAPTER 7 ANALYSIS OF RESULTS

Two different examples in the recent literatures have been used to demonstrate the efficiency of the proposed Model 1. Abdou and Lee [1] have developed a heuristic and a robot sequence program for loading boxes of different sizes on a pallet Their focus was on the physical aspect of robotic palletization. The main objectives of their study are minimization of work-in-process and palletization time, and maximization of pallet utilization. Although, their results have shown 100% pallet utilization for the same set of data used in the example in 6.1, the layout of boxes (Figure 7.1) does not satisfy the stability criteria required for the loading patterns Since the proposed Model 1 considers loading stability, the layout obtained is more reasonable than the one obtained from the heuristic.

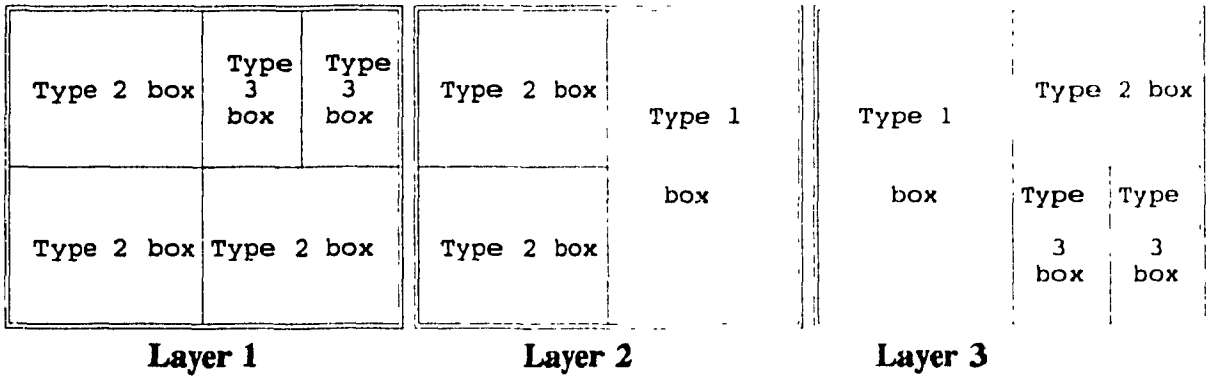


Figure 7.1. Results obtained from Abdou and Lee's Algorithm

Another case was discussed by Tsai et al. [22]. They considered a pallet of 7x4, and three types of boxes: 4x2, 3x2 and 2x2. Two optimal solutions obtained for one layer by applying their model, are illustrated in Figure 7 2 The same set of data was applied

to the proposed Model 1 However, another constraint, which is the number of boxes available for each type, is added. The number of boxes available are 5, 2 and 10, for box Type 1, 2, 3, respectively The resulting loading patterns for the three layers, are shown in Figure 7 3

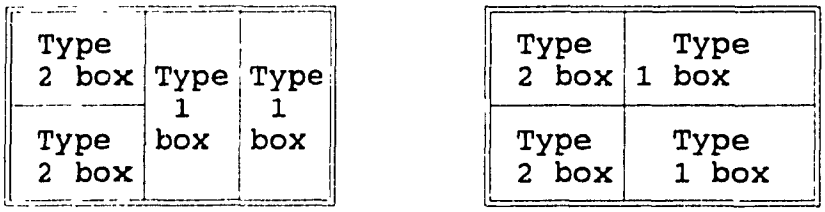


Figure 7.2 Result obtained from Tsai's model

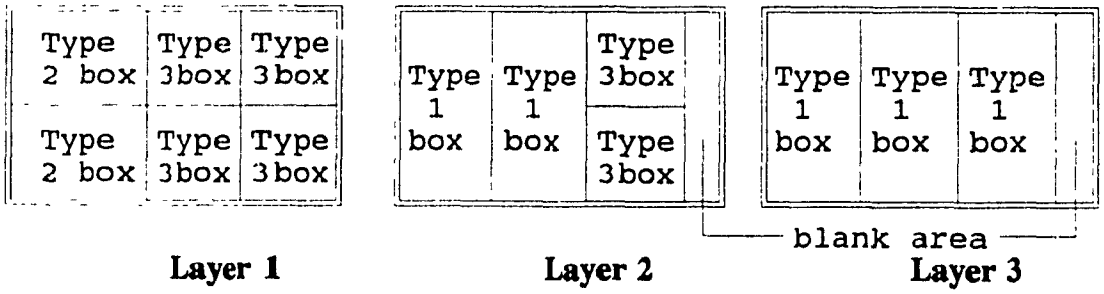


Figure 7.3 Result obtained from the proposed model

The pallet utilization reache by the proposed model is :

$$U = (2*6 + 4*8 + 8*4)/3*28 = 90.5\%$$

The proposed model does not give 100% utilization for this problem, because there are not enough right size boxes available. The solution obtained from the proposed model has shown three practical differences compared with Tsai's solution, as illustrated in Table 7 1.

Table 7.1 The Main Differences Between The Two Solutions

Characteristics	Tsai's model	Proposed model
solution for multi-layer	NO	YES
limits on the No. of boxes	NO	YES
considered stability criteria	NO	YES

The case study of the proposed Model 2 shows that Model 2 can solve 3D palletization problems mathematically and achieve near optimal solution It is useful for warehouse situation when all the information of boxes are available However, at the same time, it shows that the solution of 3D problems by Model 2 could be time consuming and it is not suitable for the situation in which the boxes are coming from a conveyor in random orders.

The three simulations for Model 3 show the effect of random box order on the palletization patterns. Simulation 1 uses the same input used by the case study of Model 2. The results shown in Table 6. 14 illustrate that, Model 3 produces different loading patterns depending on incoming sequences of boxes Nine out of ten runs reach 100% pallet utilization. The results from Sequence 1 and 8 have the same layout as patterns (d) and (b) as shown in Figure 6.3. And Sequence 2 has the in same layout as pattern (a) shown in Figure 6 2. Other sequences have shown different patterns other than the ones illustrated in Figure 6 2 and Figure 6 3 This is due to the different environment of applying the rules of Model 2 and Model 3

Simulation 2 uses one of the outputs from Model 2 The results are shown in Figure 6.16. Because there are only seven boxes are available, two sequences failed to fill

the pallet 100%. However, the WIP and palletization time are significantly low compared with Simulation 1. Among the ten sequences, the results from Sequence 2, 5 and 8 have shown the same layout as patterns (c), (a) and (c) respectively, as shown in Figure 6.3

Simulation 3 takes all seven types of boxes with no limits as input. The results, as shown in Table 6.18 reveal that all sequences generate 100% pallet utilization. However, compared with the results from Simulation 1 and Simulation 2, the palletization time and WIP are obviously higher. Because of the different input, only Sequence 10 produces the same layout as pattern (c), as shown in Figure 6.3

All the results of the simulations for Model 3 show that the model is capable of giving maximum pallet utilization and, at the same time, controls the WIP by forming larger blocks by small boxes. The more boxes available, the easier to reach 100% utilization. However, in some cases, WIP could be higher, such as the WIP resulted from Sequence 4, 6, and 9 in simulation 3. In these cases, if the layer combination is changed to different heights, the WIP could be reduced by removing boxes from the pallet and replacing them with the new pattern of boxes. However, doing so will increase the palletization time due to the reloading process. Therefore, there is a trade off between palletization time and WIP, should exist. One solution to the problem is multi-pallet palletization. Considering Sequence 9 in simulation 3 as an example, the WIP after the palletization is:

Type[2]=4; Type[3]=1; Type[4]=1; Type[6]=5; Type[7]=1,

Among these boxes, the four Type 2 boxes can be loaded on another pallet directly

Therefore, if two pallet are available at the same time, the WIP could be reduced greatly. The comparison also suggests that if the optimal loading pattern could be obtained (for instance by running Model 2) and the feeding could be arranged, the WIP can be better controlled and the efficiency of the palletization process could be increased.

Comparing the results from the case study of Model 2 and Model 3, one significant different is that Model 2 always gives the best layout in terms of loading stability. From the theoretical point of view, Model 2 provides a way to deal with the 3D situation mathematically. From the practical point of view, Model 3 is more suitable for the robotic palletization in automated manufacturing systems.

CHAPTER 8 CONCLUSIONS

The focus of this study is on the development of loading models for the 2½D and 3D palletization problems Model 1 handles the Type 2 palletization problem by employing Linear Programming technique The result shows that the model is capable of optimizing the layout of multi-layer, multi-dimension problems Model 2 provides a solution for the 3D palletization problem by a combination of mathematical model and some rules of thumb. The result generated from Model 2 is near optimal, since the heuristic is used for formulating the blocks Model 2 gives better utilization and stability compared with Model 3, because Model 3 is developed for the physical palletization system, in which WIP and palletization time must be taken in account Model 3 is based on a heuristic and suitable for real industrial situation where the boxes are coming in a random sequence and there are constraints on the WIP and palletization time The case study shows that Model 3 yields 100% utilization and is capable of putting boxes of different heights in one layer by formulating blocks

This study extended the research horizon for Type 2 palletization problem from single layer to multi-layer and explored Type 3 palletization problem by both mathematical and heuristic approach

However, as a result of this working experience, it is found that the palletization problem is too complex and there is still more research needed to be conducted The future study will be focusing particularly on the following topics:

1 improve the current models for more flexible environment by incorporating boxes with no integral proportion

2 develop both mathematical and physical models for Type 4 problem.

3 improve the physical robotic palletization system in terms of robot speed, palletization time, and accordingly, the overall utilization of RPS.

4 implement multi-pallet system, such as two pallets to be loaded at the same time.

REFERENCES

1. Abdou,G and Lee, E., 1992 **Contribution to the Development of Robotics Palletization of Multiple-box Sizes** J. Manufacturing Systems, Vol 11, No 3.

2 Abdou, G.H and Yang, M., 1992 **Systematic Procedures for the Pailetization Problem.** Proceedings of the IASTED International Conference on CONTROL and ROBOTICS, Vancouver, Canada, August 1992

3. Beasley, J.E., 1985. **An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure**, Operations Research, 33, 49-64

4. Bischoff, E.E., and Mariott, M D., **A Comparative Evaluation of Heuristics for Container Loading** European Journal of Operational Research 44(1990)267-276

5. Brown, D.J., 1980. **An Improved BL Lower Bound.** Information Processing Letters, 11, No 1, 37-39.

6. Carpenter, H., and Dowsland, W. B., 1985 **Practical Considerations of the Pallet-Loading Problem** Journal of the Operational Research Society Vol 36 No 6, 489-497, 1985

7. Chen, C.S., Sarin, S. and Ram, B., 1991. **The Pallet Packing Problem for Non-uniform Box Sizes.** International Journal of Production Research, 29, No 10, 1963-1968

8. Dowsland, W.B., **Three-dimensional Packing—Solution Approaches and Heuristic Development** International Journal of Production Research 1991, vol 29, No. 8, 1673-1685

9. **G3000 User's Manual** CRS PLUS, 1990

10. George, J.A and Robinson, D.F., **A Heuristic for Packing boxes into a Container** Computers and Operational Research 7(1980)147-156

11. Glimore, P.C. and Gomory, R E., 1965 **Multistage Cutting Stock Problems of Two and More Dimensions** Operations Research, 13, 94-120

12. Harald Dyckhoff and Ute Finke, 1992. **Cutting and Packing in Production and Distribution** Physica-Verlag Heideberg, Germany

13 Hinxman, A.I, 1980. **The Trim-Lose and Assortment Problems: A Survey** European Journal of Operational Research 5(1980)8-18

14 Hodgson, T.J, 1982. **A Combined Approach to the Pallet Loading Problem**. IIE Transactions, 14, No.3 175-182.

15 **IP-LIB User's Guide & Reference Manual** Matrox. 1991.

16. Lee, E., 1990 **Automated Palletization of Multiple Box Sizes** Master Thesis. University of Windsor

17. Loschau, G., 1989. **Stability Criteria for Column Stocks**. Packaging Technology and Science, 2, 155-163.

18 Penington, R.A and Tanchoco, J.M., 1988 **Robotic Palletization of Multiple Box Sizes.**International Journal of Production Research, vol.26, No. 1, 95-105

19 Schrage, L., 1991 **LINDO An Optimization Modelling System** Fourth Edition. The Scientific Press, South San Francisco. 1991

20 Smith, A. and DeCani, P., 1980. **An Algorithm to Optimize the Layout of Boxes in Pallet** Journal of Operational Research Society, 31, 573-578

21. Tanchoco, J.M and Agee, M.H.,1981. **Plan Unit Loads to Interact with All Components of Warehouse System**. Industrial Engineering, June 1981, 36-48.

22. Tsai, R.D., Malstrom, E.M., and Meeks, H.D., 1988. **A Two-Dimensional Palletizing Procedure for Warehouse Loading Operations.** IIE Transactions, 20, No. 4, 418-425.

APPENDIX A
ILP Model for The Illustration Problem in 6.1: Model and the Result

```
MAX     960 X11 + 960 X21 + 480 X31 + 480 X41 + 240 X51 + 240 X61
        + 960 X12 + 960 X22 + 480 X32 + 480 X42 + 240 X52 + 240 X62
        + 960 X13 + 960 X23 + 480 X33 + 480 X43 + 240 X53 + 240 X63
        + Z - Y

SUBJECT TO

     2)  X11 + X21 + X12 + X22 + X13 + X23 <=  2
     3)  X31 + X41 + X32 + X42 + X33 + X43 <=  6
     4)  X51 + X61 + X52 + X62 + X53 + X63 <=  4
     5)  960 X11 + 960 X21 + 480 X31 + 480 X41 + 240 X51 + 240 X61 <=  1920
     6)  960 X12 + 960 X22 + 480 X32 + 480 X42 + 240 X52 + 240 X62 <=  1920
     7)  960 X13 + 960 X23 + 480 X33 + 480 X43 + 240 X53 + 240 X63 <=  1920
     8)  A1 + B1 + C1 + D1 + E1 + F1 + G1 + H1 + I1 + J1 + K1 + L1 + M1
    + N1 <=  40
     9)  A2 + B2 + C2 + D2 + E2 + F2 + G2 + H2 + I2 + J2 + K2 + L2 + M2
    + N2 <=  40
    10)  A3 + B3 + C3 + D3 + E3 + F3 + G3 + H3 + I3 + J3 + K3 + L3 + M3
    + N3 <=  40
    11) - 20 X61 + 2 D1 + E1 + 2 H1 + I1 + 4 K1 + 3 L1 + 2 M1 + N1 >=  0
    12) - 20 X62 + 2 D2 + E2 + 2 H2 + I2 + 4 K2 + 3 L2 + 2 M2 + N2 >=  0
    13) - 20 X63 + 2 D3 + E3 + 2 H3 + I3 + 4 K3 + 3 L3 + 2 M3 + N3 >=  0
    14) - 24 X41 - 12 X51 + C1 + 2 G1 + H1 + I1 + J1 >=  0
    15) - 24 X42 - 12 X52 + C2 + 2 G2 + H2 + I2 + J2 >=  0
    16) - 24 X43 - 12 X53 + C3 + 2 G3 + H3 + I3 + J3 >=  0
    17) - 40 X21 - 20 X31 + 2 B1 + C1 + D1 + E1 + F1 >=  0
    18) - 40 X22 - 20 X32 + 2 B2 + C2 + D2 + E2 + F2 >=  0
    19) - 40 X23 - 20 X33 + 2 B3 + C3 + D3 + E3 + F3 >=  0
    20) - 24 X11 + A1 >=  0
    21) - 24 X12 + A2 >=  0
    22) - 24 X13 + A3 >=  0
    23)  X11 + X21 + X31 + X41 + X51 + X61 - X12 - X22 - X32 - X42 - X52
   - X62 <=  0
    24)  X12 + X22 + X32 + X42 + X52 + X62 - X13 - X23 - X33 - X43 - X53
   - X63 <=  0
    25)  960 X11 + 960 X21 + 480 X31 + 480 X41 + 240 X51 + 240 X61
   - 960 X12 - 960 X22 - 480 X32 - 480 X42 - 240 X52 - 240 X62 >=  0
    26)  960 X12 + 960 X22 + 480 X32 + 480 X42 + 240 X52 + 240 X62
   - 960 X13 - 960 X23 - 480 X33 - 480 X43 - 240 X53 - 240 X63 >=  0
```

```
27)  A1 >=  0
28)  B1 >=  0
29)  C1 >=  0
30)  D1 >=  0
31)  E1 >=  0
32)  F1 >=  0
33)  G1 >=  0
34)  H1 >=  0
35)  I1 >=  0
36)  J1 >=  0
37)  K1 >=  0
38)  L1 >=  0
39)  M1 >=  0
40)  A2 >=  0
41)  B2 >=  0
42)  C2 >=  0
43)  D2 >=  0
44)  E2 >=  0
45)  F2 >=  0
46)  G2 >=  0
47)  H2 >=  0
48)  I2 >=  0
49)  J2 >=  0
50)  K2 >=  0
51)  L2 >=  0
52)  M2 >=  0
53)  N2 >=  0
54)  A3 >=  0
55)  B3 >=  0
56)  C3 >=  0
57)  D3 >=  0
58)  E3 >=  0
59)  F3 >=  0
60)  G3 >=  0
61)  H3 >=  0
62)  I3 >=  0
63)  J3 >=  0
64)  K3 >=  0
65)  L3 >=  0
66)  M3 >=  0
67)  N3 >=  0
68)  X13 + X23 + X33 + X43 + X53 + X63 - Z =   0
69)  X11 + X21 + X31 + X41 + X51 + X61 - Y =   0
END
```

GIN 62

LP OPTIMUM FOUND AT STEP 83
OBJECTIVE VALUE = 5764.00000
ENUMERATION COMPLETE. BRANCHES= 0 PIVOTS= 83

LAST INTEGER SOLUTION IS THE BEST FOUND
RE-INSTALLING BEST SOLUTION...

OBJECTIVE FUNCTION VALUE

1) 5764.0000

VARIABLE	VALUE	REDUCED COST
X11	.000000	-960.000000
X21	2.000000	-960.000000
X31	.000000	-480.000000
X41	.000000	-480.000000
X51	.000000	-240.000000
X61	.000000	-240.000000
X12	.000000	-960.000000
X22	.000000	-960.000000
X32	4.000000	-480.000000
X42	.000000	-480.000000
X52	.000000	-240.000000
X62	.000000	-240.000000
X13	.000000	-960.000000
X23	.000000	-960.000000
X33	2.000000	-480.000000
X43	.000000	-480.000000
X53	.000000	-240.000000
X63	4.000000	-240.000000
Z	6.000000	-1.000000
Y	2.000000	1.000000
A1	.000000	.000000
B1	40.000000	.000000
C1	.000000	.000000
D1	.000000	.000000
E1	.000000	.000000
F1	.000000	.000000
G1	.000000	.000000
H1	.000000	.000000
I1	.000000	.000000
J1	.000000	.000000

K1	.000000	.000000
L1	.000000	000000
M1	.000000	.000000
N1	.000000	.000000
A2	.000000	.000000
B2	40.000000	.000000
C2	.000000	.000000
D2	.000000	.000000
E2	.000000	.000000
F2	.000000	.000000
G2	.000000	.000000
H2	.000000	.000000
I2	.000000	.000000
J2	.000000	.000000
K2	.000000	.000000
L2	.000000	.000000
M2	.000000	.000000
N2	.000000	.000000
A3	.000000	.000000
B3	20.000000	.000000
C3	.000000	.000000
D3	.000000	.000000
E3	.000000	.000000
F3	.000000	.000000
G3	.000000	.000000
H3	.000000	.000000
I3	.000000	.000000
J3	.000000	.000000
K3	20.000000	.000000
L3	.000000	.000000
M3	.000000	.000000
N3	000000	.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	.000000	.000000
3)	.000000	.000000
4)	.000000	.000000
5)	.000000	.000000
6)	.000000	.000000
7)	.000000	.000000
8)	.000000	.000000
9)	.000000	.000000
10)	.000000	.000000

11)	.000000	.000000
12)	.000000	000000
13)	.000000	.000000
14)	.000000	.000000
15)	.000000	.000000
16)	.000000	.000000
17)	.000000	.000000
18)	.000000	.000000
19)	.000000	.000000
20)	.000000	.000000
21)	.000000	.000000
22)	.000000	.000000
23)	2.000000	.000000
24)	2.000000	.000000
25)	.000000	.000000
26)	.000000	.000000
27)	.000000	.000000
28)	40.000000	.000000
29)	.000000	.000000
30)	.000000	.000000
31)	.000000	.000000
32)	.000000	.000000
33)	.000000	.000000
34)	.000000	.000000
35)	.000000	.000000
36)	.000000	.000000
37)	.000000	.000000
38)	.000000	.000000
39)	.000000	.000000
40)	.000000	.000000
41)	40.000000	.000000
42)	.000000	.000000
43)	.000000	.000000
44)	.000000	.000000
45)	.000000	.000000
46)	.000000	.000000
47)	.000000	.000000
48)	.000000	.000000
49)	.000000	.000000
50)	.000000	.000000
51)	.000000	.000000
52)	.000000	.000000
53)	.000000	.000000
54)	.000000	.000000

55)	20.000000	.000000
56)	000000	.000000
57)	.000000	.000000
58)	.000000	.000000
59)	.000000	.000000
60)	.000000	.000000
61)	.000000	.000000
62)	.000000	.000000
63)	.000000	000000
64)	20 000000	.000000
65)	.000000	.000000
66)	.000000	.000000
67)	.000000	.000000
68)	.000000	.000000
69)	.000000	.000000

NO. ITERATIONS= 84
BRANCHES= 0 DETERM.= 1.000E 0

APPENDIX B
BASIC Program for Formulating ILP Model

```
5 REM MAIN PROGRAM
10 READ LP,WP,N1,NL
32 DIM L(50), W(50), N(50),X(50),S(50),Y(50,50),K(50,50),P(50,50)
40 FOR I=1 TO N1
60 READ L(I),W(I),NB(I)
70 S(I)=L(I)*W(I)
72 W(N1+I)=L(I)
80 L(N1+I)=W(I)
81 S(N1+I)=L(N1+I)*W(N1+I)
83 REM PRINT L(I),I,L(N1+I),N1+I
100 NEXT I
110 N=2*N1
111 FOR I=1 TO N
112 L1(I)=L(I):W1(I)=W(I)
113 NEXT I
120 GOSUB 500
140 GOSUB 1500
150 GOSUB 3000
160 END
357 FOR I=1 TO M1
500 REM LENGTH SET FORMULATION MODULE
510 FOR I=1 TO N
520 FOR J=I+1 TO N
530 IF L(I)=L(J) THEN 560
550 NEXT J
552 K=K+1:X(K)=L(I)
555 REM PRINT X(K),K
560 NEXT I
570 FOR I=1 TO 2*K
580 FOR J=I+1 TO 2*K
590 IF X(I) > X(J) THEN GOTO 610
600 ELSE 601
601 H=H+1
602 X(K+H)=X(I)
604 GOTO 620
610 NEXT J
612 M=M+1
615 IF X(I)>0 THEN L(M)=X(I):M1=M
620 NEXT I
630 RETURN
```

```
1500 REM STRIP COMBINATION MODULE
1510 FOR I=1 TO M1.B=LP/L(I)
1511 N(I)=INT(B)
1520 NEXT I
1522 FOR K10=0 TO N(10)
1530 FOR K9=0 TO N(9)
1535 FOR K8=0 TO N(8)
1540 FOR K7=0 TO N(7)
1550 FOR K6=0 TO N(6)
1560 FOR K5=0 TO N(5)
1570 FOR K4=0 TO N(4)
1580 FOR K3=0 TO N(3)
1590 FOR K2=0 TO N(2)
1595 FOR K1=0 TO N(1)
1610
L1=K1*L(1)+K2*L(2)+K3*L(3)+K4*L(4)+K5*L(5)+K6*L(6)+K7*L(7)+K8*L(8)+K9*
L(9)
1620 L2=L1+K10*L(10)
1625 IF L2=0 THEN 1700
1630 IF L2<=LP THEN PRINT
K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,L2:O=O+1:PRINT O
1640 P(O,1)=K1.P(O,2)=K2:P(O,3)=K3:P(O,4)=K4:P(O,5)=K5
1650 P(O,6)=K6:P(O,7)=K7:P(O,8)=K8:P(O,9)=K9:P(O,10)=K10
1652 L3=P(O,1)*L(1)+P(O,2)*L(2)+P(O,3)*L(3)+P(O,4)*L(4)+P(O,5)*L(5)
1653 L3=L3+P(O,6)*L(6)+P(O,7)*L(7)+P(O,8)*L(8)+P(O,9)*L(9)+P(O,10)*L(10)
1655 IF L3>LP THEN 1700
1656 ELSE GOTO 1657
1657 FOR A=1 TO 10
1658 K(O,A)=P(O,A)
1659 NEXT A
1700 NEXT K1
1710 NEXT K2
1720 NEXT K3
1730 NEXT K4
1740 NEXT K5
1750 NEXT K6
1760 NEXT K7
1770 NEXT K8
1780 NEXT K9
1790 NEXT K10
1800 RETURN
2000 DATA 48,40,3,3
2010 DATA 20,12,5
2020 DATA 40,24,5
```

```
2030 DATA 24,20,5
3000 REM MODEL FORMULATION
3001 OPEN "O",#1,"PALLET.DAT"
3010 PRINT #1, "MAX  ",
3020 FOR J=1 TO NL
3030 FOR I=1 TO N
3031 A$="+":B$=STR$(S(I)):C$="Y" D$=STR$(I):E$=STR$(J)
3032 IF S(I)>=100 THEN B1$=RIGHT$(B$,3)
3033 IF S(I)>=10 AND S(I)<100 THEN B1$=RIGHT$(B$,2)
3034 IF S(I)<10 THEN B1$=RIGHT$(B$,1)
3035 IF I>=10 THEN D1$=RIGHT$(D$,2)
3036 IF I<10 THEN D1$=RIGHT$(D$,1)
3037 E1$=RIGHT$(E$,1)
3038 F$=A$+B1$+C$+D1$+E1$
3040 PRINT #1, F$,
3050 NEXT I
3060 NEXT J
3065 PRINT #1,  "+ XT - XB"
3070 PRINT #1, "SUBJECT TO"
3075 PRINT "BOX NUMBER CONSTRAINTS"
3080 FOR I=1 TO N1
3090 FOR J=1 TO NL
3091 A$="+Y".B$=STR$(I):C$=STR$(J):D$=STR$(I+N1):C1$=RIGHT$(C$,1)
3092 IF I>=10 THEN B1$=RIGHT$(B$,2)
3093 IF I<10 THEN B1$=RIGHT$(B$,1)
3094 IF I+N1>=10 THEN D1$=RIGHT$(D$,2)
3095 IF I+N1<10 THEN D1$=RIGHT$(D$,1)
3096 E$=A$+B1$+C1$+A$+D1$+C1$
3100 PRINT #1, E$;
3110 NEXT J
3120 PRINT #1, "=" NB(I)
3130 NEXT I
3140 PRINT
3150 PRINT "PALLET AREA CONSTRAINTS"
3160 SP=LP*WP
3170 FOR J=1 TO NL
3180 FOR I=1 TO N
3181 A$="Y":B$=STR$(I):C$=STR$(J):C1$=RIGHT$(C$,1)
3182 IF I>=10 THEN B1$=RIGHT$(B$,2)
3183 IF I<10 THEN B1$=RIGHT$(B$,1)
3184 D$=A$+B1$+C1$
3190 PRINT #1, "+" S(I) D$;
3200 NEXT I
3210 PRINT #1, "<=" SP
```

```
3220 NEXT J
3230 PRINT
3240 PRINT "LAYER AREA CONSTRIANTS"
3250 FOR J=1 TO NL-1
3260 FOR I=1 TO N
3261 A$="Y".B$=STR$(I):C$=STR$(J):C1$=RIGHT$(C$,1)
3262 IF I<10 THEN B1$=RIGHT$(B$,1)
3263 IF I<10 THEN B1$=RIGHT$(B$,1)
3264 D$=A$+B1$+C1$
3270 PRINT #1, "+" S(I) D$;
3280 NEXT I
3300 FOR I=1 TO N
3301 A$="Y":B$=STR$(I):C$=STR$(J+1):C1$=RIGHT$(C$,1)
3302 IF I>=10 THEN B1$=RIGHT$(B$,2)
3303 IF I<10 THEN B1$=RIGHT$(B$,1)
3304 D$=A$+B1$+C1$
3310 PRINT #1, "-" S(I) D$;
3320 NEXT I
3325 PRINT #1,  ">=0"
3330 NEXT J
3340 PRINT
3350 PRINT "STABILITY CONSTRAINTS"
3360 FOR J=1 TO NL-1
3370 FOR I=1 TO N
3371 A$="+Y".B$=STR$(I):C$=STR$(J):C1$=RIGHT$(C$,1)
3372 IF I>=10 THEN B1$=RIGHT$(B$,2)
3373 IF I<10 THEN B1$=RIGHT$(B$,1)
3374 D$=A$+B1$+C1$
3380 PRINT #1, D$;
3390 NEXT I
3400 FOR I=1 TO N
3401 A$="-Y".B$=STR$(I):C$=STR$(J+1):C1$=RIGHT$(C$,1)
3402 IF I>=10 THEN B1$=RIGHT$(B$,2)
3403 IF I<10 THEN B1$=RIGHT$(B$,1)
3404 D$=A$+B1$+C1$
3410 PRINT #1, D$;
3420 NEXT I
3430 PRINT #1, "<=0"
3440 NEXT J
3450 PRINT "STRIP CONSTRANTS":PRINT O
3460 FOR J=1 TO NL
3470 FOR I=1 TO O
3471 A$="+U":B$=STR$(I):C$=STR$(J):C1$=RIGHT$(C$,1)
3472 IF I>=10 THEN B1$=RIGHT$(B$,2)
```

```
3473 IF I<10 THEN B1$=RIGHT$(B$,1)
3474 D$=A$+B1$+C1$
3480 PRINT #1, D$,
3490 NEXT I
3500 PRINT #1, "=" WP
3505 PRINT
3510 NEXT J
3520 PRINT
3530 FOR T=1 TO 10
3531 KIT=0
3533 IF L(T)>0 THEN 3535
3534 ELSE GOTO 3940
3535 FOR J=1 TO NL
3540 FOR I=1 TO O
3550 KIT=KIT+K(I,T)
3551 A$="+":B$=STR$(K(I,T)):C$="U":D$=STR$(I):E$=STR$(J)
3552 IF K(I,T)>=10 THEN B1$=RIGHT$(B$,2)
3553 IF K(I,T)<10 THEN B1$=RIGHT$(B$,1)
3554 IF I>=10 THEN D1$=RIGHT$(D$,2)
3555 IF I<10 THEN D1$=RIGHT$(D$,1)
3556 E1$=RIGHT$(E$,1)
3557 F$=A$+B1$+C$+D1$+E1$
3560 IF K(I,T)>0 THEN PRINT #1, F$;
3570 KIT=KIT+K(I,T)
3580 NEXT I
3590 IF KIT>0 THEN GOTO 3610
3600 IF KIT=0 THEN 3780
3610 FOR W=1 TO N
3620 IF L(T)=L1(W) THEN GOTO 3640
3630 IF L(T)<>L1(W) THEN GOTO 3730
3640 A$="-":B$=STR$(W1(W))
3650 IF W1(W)>=10 THEN B1$=RIGHT$(B$,2)
3660 IF W1(W)<10 THEN B1$=RIGHT$(B$,1)
3670 C$="Y"
3680 D$=STR$(W):IF W>=10 THEN D1$=RIGHT$(D$,2)
3690 IF W<10 THEN D1$=RIGHT$(D$,1)
3700 E$=STR$(J):E1$=RIGHT$(E$,1)
3710 F$=A$+B1$+C$+D1$+E1$
3720 PRINT #1, F$;
3730 NEXT W
3740 PRINT #1, ">=0"
3750 PRINT
3760 NEXT J
3770 PRINT
```

```
3780 NEXT T
3790 FOR I=1 TO N
3800 A$="+Y":B$=STR$(I):B1$=RIGHT$(B$,1):C$=STR$(NL):C1$=RIGHT$(C$,1)
3810 D$=A$+B1$+C1$
3820 PRINT #1, D$;
3830 NEXT I
3840 PRINT #1, "-XT=0"
3850 FOR I=1 TO N
3860 A$="+Y":B$=STR$(I):C$="1":B1$=RIGHT$(B$,1)
3870 D$=A$+B1$+C$
3880 PRINT #1, D$;
3890 NEXT I
3900 PRINT #1, "-XB=0"
3910 NN=N*NL + O*NL + 2
3920 PRINT #1, "END"
3930 PRINT #1, "GIN" NN
3935 CLOSE
3940 RETURN
```

APPENDIX C
C Program for MODEL 3

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int l[9]={0,40,40,24,24,24,20,20,40};
int w[9]={0,24,24,20,20,20,12,12,24},
int h[9]={0,10,20,10,20,30,20,30,30};
int N=7;
int L=48;
int W=40;
int H=40;
int HG[4]={0,10,20,30};
static int No,RN[20],G[4],Type[10],HR,HL,x[10],y[10],nn,mm,ls,ws;
static int check,HA[4],s[20],X[4][10],LN,pallet[50][50],i,j,k,kk;
float S,U,UL;


/* MAIN PROGRAM */


void main()

{

printf("**************************************\n");
printf("****** Program for Heuristic 2 ******\n");
printf("**************************************\n");


/* initailizing */


S=L*W;
HA[1]=0; HA[2]=0; HA[3]=0;
G[1]=0; G[2]=0; G[3]=0;

No=1; LN=1; nn=1; mm=0;

for(i=1;i<=N+1;i++) s[i]=l[i]*w[i];

U=0.0; UL=0.0,
```

```
check=0,
HR=H;

for(i=0,i<10;i++)
{
X[1][i]=0,
X[2][i]=0;
X[3][i]=0,
Type[i]=0;
}

randomize();

while(HR != 0)
{

switch(check)
{
case 0:
        new_layer();
        check=1;
        break;
case 1:
        fill_layer();
        ++LN;
        nn=1;
        printf("UL=%f\n",UL);
        UL=0.0;
        HR=0;
/*      HR=HR-HL; */
        check=0;
        break;
}

G[1]=Type[1]+Type[3];
G[2]=Type[2]+Type[4]+Type[6];
G[3]=Type[5]+Type[7];

printf("G[1]=%d G[2]=%d G[3]=%d\n",G[1],G[2],G[3]);

}


}                       /* end of program */
```

```
new_layer()                    /* Function for New Layer Selection */

{

switch(HR)
{
case 10:
        HL=10;
        break;
case 20:
        while(HL==0)
        {
        new_box();
        block20_form();
        if(G[2] >= 2) HL=HG[2];
        else if(G[1] >=3) HL=HG[1];
        }
        break;
case 30:
        while(HL==0)
        {
        new_box();
        block30_form();
        if(G[3] > 0) HL=HG[3];
        else if(G[2] > 1) HL=HG[2];
        else if(G[1] > 2) HL=HG[1];
        }
        break;
case 40:
        while(HL==0)
        {
        new_box();
        if(G[3] > 0) HL=HG[3];
        else if(G[2] > 1) HL=HG[2];
        else if(G[1] > 2) HL=HG[1];
        }
        break;
}


printf("HL=%d\n",HL);

return;
```

```
}


fill_layer()
{

int size;

/* printf("nn=%d\n",nn); */

/* initializ the pallet */


for(i=1; i<=W; i++)
{
for(j=1, j<=L, j++) pallet[i][j]=0;
}


switch(HL)
{
case 10:
        select_box1();
        break;
case 20.
        select_box2();
        break;
case 30:
        select_box3();
        break;
}


return;
}


/* Function for Selecting Boxes to load */

select_box1()
{

        for(i=1;;i++)
        {
```

```
        if(G[1] > 1)
        {
        if(Type[1] >= 1) X[LN][nn]=1;
        else X[LN][nn]=3;
        printf("X[%d][%d]=%d\n",LN,nn,X[LN][nn]);

        load_box();

        if(G[1] == 1)
        {
        for(i=1; i<=8 ;i++)
        {
        if(Type[i] == 1 && h[i] == 10)
        {
        if((UL+s[i]*1.0/S) == 1)
        {
        X[LN][nn]=i;
        load_box();
        }
        break;
        }
        }
        }

        if(UL == 1) break;
        }
        else new_box();
        }

return;
}

select_box2()
{

        for(i=1;;i++)
        {
        block20_form();
        if(G[2] > 1)
        {
        if(Type[2] >=1) X[LN][nn]=2;
        else if(Type[4] >= 1) X[LN][nn]=4;
        else X[LN][nn]=6;
        printf("X[%d][%d]=%d\n",LN,nn,X[LN][nn]);
```

```
        load_box();

        if(G[2] == 1)
        {
        for(i=1; i<=8 ;i++)
        {
        if(Type[i] == 1 && h[i] == 20)
        {
        if((UL+s[i]*1.0/S) == 1)
        {
        X[LN][nn]=i;
        load_box();
        }
        break;
        }
        }
        }
        if(UL == 1) break;
        }
        else new_box();
        }

return;
}

select_box3()
{

        for(i=1;;i++)
        {
        block30_form();
        if(G[3] > 1)
        {
        if(Type[8] >= 1) X[LN][nn]=8;
        else if(Type[5] >= 1) X[LN][nn]=5;
        else X[LN][nn]=7;
        printf("X[%d][%d]=%d\n",LN,nn,X[LN][nn]);

        load_box();

        if(G[3] == 1)
        {
        for(i=1; i<=8 ;i++)
        {
```

```
        if(Type[i] == 1 && h[i] == 30)
        {
        if((UL+s[i]*1.0/S) == 1.0)
        {
        X[LN][nn]=i;
        load_box();
        }
        break;
        }
        }
        }

        if(UL == 1) break;
        }
        else new_box();
        }

return;
}

load_box()
{
int k;

for(i=1; i<=L; i++)
{
for(j=1; j<=W; j++)
{

if(pallet[j][i] == 0)
{
ws=0; ls=0;

for(kk=1; kk<=W; ++kk)
{
if(pallet[kk][i] == 0) ++ws;
}

for(k=1; k<=L; k++)
{
if(pallet[j][k] == 0) ++ls;
}

x[nn]=j-1; y[nn]=i-1;
```

```
printf("ws=%d ls=%d %d %d\n", ws,ls,x[nn],y[nn]);

if( W%l[X[LN][nn]] == 0.0)
{
if(ls >= w[X[LN][nn]] && ws >= l[X[LN][nn]])
{
if(l[X[LN][nn]] == 20 && l[X[LN][nn-1]] == 20 && x[nn-1] == 0)
{
x[nn]=0; y[nn]=y[nn-1]+12;
}
put_box(x[nn],y[nn],w[X[LN][nn]],l[X[LN][nn]]);
++nn;
mm=1;
}
}

else if( W%w[X[LN][nn]] == 0.0)
{
if(ls >= l[X[LN][nn]] && ws >= w[X[LN][nn]])
{
if(w[X[LN][nn]] == 20)
{
if(x[nn] == 20 && y[nn] == 12)
{
x[nn] = 0; y[nn] = 24;
}
if(x[nn] == 0 && y[nn] == 12)
{
x[nn] = 20; y[nn] = 24;
}
}
put_box(x[nn],y[nn],l[X[LN][nn]],w[X[LN][nn]]);
++nn;
mm=1;
}
}
break;
}

}
if(mm==1) break;
}
```

```
if(mm == 0) remove_box();
else
{

UL=UL+s[X[LN][nn-1]]*1.0/S;

for(i=1; i<=8 ;i++)
{
if(X[LN][nn-1] == i)
{
Type[i]=Type[i]-1;
/* printf("Type[%d]=%d\n",i,Type[i]);
printf("x[%d]=%d  y[%d]=%d \n",nn-1,x[nn-1],nn-1,y[nn-1]),
*/
switch(i)
{
case 1:
        --G[1];
        break;
case 2:
        --G[2];
        break;
case 3:
        --G[1];
        break;
case 4:
        --G[2];
        break;
case 5:
        --G[3];
        break;
case 6:
        --G[2];
        break;
case 7:
        --G[3];
        break;
case 8:
        --G[3];
        break;
}
break;
}
}
```

```
}

printf("%d  %d  %d\n",G[1], G[2], G[3]);

mm=0,

return,
}


put_box(xx,yy,ww,ll)
{
int a,b,

/* printf("%d %d\n",ww,ll), */

for(i=1, i<=ll; i++)
{
a=xx+i,
for(j=1, j<=ww; j++)
{
b=yy+j,
pallet[a][b]=1,
}
}



return,
}


remove_box()
{
int a,b,c,d;

printf("nn=%d\n",nn);

for(i=1, i<=nn-1, i++)
{
printf("X[%d][%d]=%d\n",LN,i,X[LN][i]),

if( ((s[X[LN][nn]]-s[X[LN][i]])*1.0/S+UL) == 1)
{
```

```
if( W%l[X[LN][i]] == 0.0)
{
for(a=1; a<=l[X[LN][i]]. a++)
{
for(b=1, b<=w[X[LN][i]]. b++)
{
c=a+x[i];
d=b+y[i];
pallet[c][d]=0;
}
}
}
else if( W%w[X[LN][i]] == 0.0)
{
for(a=1; a<=w[X[LN][i]]; a++)
{
for(b=1; b<=l[X[LN][i]]; b++)
{
c=a+x[i];
d=b+y[i];
pallet[a][b]=0;
}
}
}

UL=UL-s[X[LN][i]]*1.0/S;

printf("x[%d]=%d y[%d]=%d\n",i,x[i],i,y[i]),

for(k=1; k<=8 ;k++)
{
if(X[LN][i] == k)
{
printf("%d     ",Type[k]);
++Type[k];
printf("%d    \n ",Type[k]);
printf("Put type[%d] box back to the holding area\n",k);

switch(k)
{
case 1:
        ++G[1];
        break;
case 2:
```

```
        ++G[2];
        break,
case 3:
        ++G[1];
        break;
case 4:
        ++G[2];
        break;
case 5:
        ++G[3],
        break,
case 6
        ++G[2];
        break;
case 7.
        ++G[3];
        break;
case 8.
        ++G[3],
        break;
}
break;
}
}

}
}

return;
}


/* Function for New Box Generation */

new_box()

{

RN[No] = random(N)+1;

printf("No   %d box is a type %d box.\n", No, RN[No]);

for(i=1,i<=7;i++)
{
```

```
if(RN[No]==i) ++Type[i];
}

if(h[RN[No]]==10) ++G[1];
if(h[RN[No]]==20) ++G[2];
if(h[RN[No]]==30) ++G[3];

/* printf("%d %d %d\n",G[1],G[2],G[3]); */
/*if((G[1]+G[2]+G[3]) != No) printf("ERROR!!!\n");*/

++No;

return;

}



block20_form()  /* Function for Generating Blocks with height of 20 */

{

while(Type[3] >= 2)
{
Type[3]=Type[3]-2;
Type[4]=Type[4]+1;
++G[2];
}

while(Type[1] >= 2)
{
Type[1]=Type[1]-2;
Type[2]=Type[2]+1;
++G[2];
}

while(Type[1] >= 1 && Type[3] >= 2)
{
Type[1]=Type[1]-1;
Type[3]=Type[3]-2;
Type[2]=Type[2]+1;
++G[2];
}
```

```
return;
}



block30_form()   /* Function for Generating Blocks with height of 30 */

{

while(Type[3] >=1 && Type[4]>=1)
{
Type[3] = Type[3]-1;
Type[4] = Type[4]-1;
Type[5] = Type[5]+1;
++G[3];
}

while(Type[3] >=1 && Type[6]>=2)
{
Type[3] = Type[3]-1;
Type[6] = Type[6]-2;
Type[5] = Type[5]+1;
++G[3];
}

while(Type[3] >=3)
{
Type[3] = Type[3]-3;
Type[5] = Type[5]+1;
++G[3];
}

while(Type[1] >=1 && Type[2]>=1)
{
Type[1] = Type[1]-1;
Type[2] = Type[2]-1;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[4]>=1 && Type[6]>=2)
{
Type[1] = Type[1]-1;
```

```
Type[4] = Type[4]-1;
Type[6] = Type[6]-2;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[4]>=1 && Type[3]>=2)
{
Type[1] = Type[1]-1;
Type[4] = Type[4]-1;
Type[3] = Type[3]-2;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[4]>=2)
{
Type[1] = Type[1]-1;
Type[4] = Type[4]-2;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[3]>=2 && Type[6]>=2)
{
Type[1] = Type[1]-1;
Type[6] = Type[6]-2;
Type[3] = Type[3]-2;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[3]>=4)
{
Type[1] = Type[1]-1;
Type[3] = Type[3]-4;
Type[8] = Type[8]+1;
}

while(Type[1] >=1 && Type[6]>=4)
{
Type[1] = Type[1]-1;
Type[6] = Type[6]-4;
Type[8] = Type[8]+1;
}

while(Type[1]>=3)
{
```

```
Type[1] = Type[1]-3;
Type[8] = Type[8]+1,
}

while(Type[1] >=2 && Type[3]>=2)
{
Type[1] = Type[1]-2;
Type[3] = Type[3]-2;
Type[8] = Type[8]+1;
}

while(Type[2] >=1 && Type[3]>=2)
{
Type[2] = Type[2]-1;
Type[3] = Type[3]-2;
Type[8] = Type[8]+1;
}

for(i=1; ;i++)
{
if(Type[8]==i) G[3]=G[3]+i;
break;
}

return;

}
```

APPENDIX D
C Program for Vision System

Program 1. For Boxes with no Orientation Angle

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <alloc.h>
#include <c:\iplib\inc\ip.h>

#include "c:\example\profile\exam.h"

UDWORD          *xprof,*yprof;
WORD              err_code;
unsigned char   black = 0x00;
unsigned char   gray  = 0x80;
unsigned char   white = 0xff;
unsigned long  x_profile,y_profile;
int                    i, j, k, ii;

VOID main()
{
switch ( err_code = VG_InitLib() )
        {
        case IP_8:
                break;
        case IP_BUSY:
                printf( "WARNING: IP-8 Board already in use.\n" );
                break;
        case IP_BOARD:
                printf( "ERROR: IP-8 board or BIOS not present\n" );
                printf( "       or wrong BIOS version\n");
                exit(0);
        case IP_DRIVER:
                printf( "ERROR: Driver (IP8DRV.SYS) not present\n" );
                printf( "       or wrong driver version\n" );
                exit(0);
        case MEM_POINTER:
                printf( "ERROR: IP-8 memory Contention\n" ),
                printf( "       Couldn't create POINTERs to memory\n" ),
                exit(0);
        default:
```

```
            printf( "ERROR: IP-8 Library Initialization,\n" );
            printf( "       Error code = %d\n", err_code );
            exit(0),
        }

VG_Reset(YES),
VG_SetInterlaceOutSt(ON);
VG_SetDispType(LIVE_VIDEO);

printf("Hit 'c' to continue...\n");
for(i=0,;i++)
{
if(getch()=='c')
break;
}
VG_SetFrameBufConfig( SGLE );
VG_SetAccessBufSt(ON);
VG_SetVideoKeyMode(ONLY_IP_OUT);
VG_SetGenLockSt(ON);
VG_SetYAxisDirection(Y_AXIS_DOWN);
VG_SetScanMode(PROGRESSIVE);
VG_SetDispType(MEMORY_VIDEO);

VG_SetKeyComp(MEMORY);
VG_SetKeyOut(LIVE);
VG_SetKeyType(SPECIAL);

/* VG_LabelImage();


printf("%d\n",VG_LabelImage());
if(VG_LabelImage() == 0) exit(0);
                */

VG_SetGrabRegUpd(DIRECT);
VG_SetGrabField(EVENBOTH);
VG_SetGrabFrameBuf( 0 );
VG_SetGrabSourcePos( 0, 0, _VG_XDispSize, _VG_YDispSize );
VG_SetGrabDestPos( 0, 1 );
VG_SetImageSourcePos( 0, 0, 0, _VG_XDispSize, _VG_YDispSize );
xprof=(UDWORD *)malloc( _VG_XDispSize*sizeof(UDWORD) );
yprof=(UDWORD *)malloc( _VG_YDispSize*sizeof(UDWORD) );

for(i=0; i < 640; i++) *(xprof+i)=0;
```

```
for(i=0; i < 480; i++) *(yprof+i)=0,

VG_SingleGrab();
while ( VG_GetGrabSt() == ON);

VG_SetObjectColor(&white,&white);

VG_CalcProfile( xprof, yprof );
VG_DrawProfile( 100, &gray, UPPER_LEFT_PROFILE, xprof, yprof );
printf("\r                \r");

ii=0;
x_profile=0;
y_profile=0;

for(i=0; i<639; i++)
        {
        printf("xprof[%d]=%lu\n",i,*(xprof+i));
        if(*(xprof+i) == 0 & *(xprof+i+1) != 0 & *(xprof+i+2) !=0) j=i+1,
        if(*(xprof+i) != 0 & *(xprof+i+1) == 0)
        {
        k=i;
        break;
        }
        }


for(i=j+5; i<=k-5; i++)
        {
        x_profile=x_profile + *(xprof+i)/255;
        ii++;
        }
x_profile=x_profile/ii;
printf("x_profile=%lu\n",x_profile);

ii=0;

for(i=0; i<479; i++)
        {
        if(*(yprof+i) == 0 & *(yprof+i+1) != 0 & *(yprof+i+2) != 0) j=i+1;
        if(*(yprof+i) != 0 & *(yprof+i+1) == 0)
        {
        k=i;
        break;
```

```
        }
        }
for(i=j+5, i<=k-5, i++)
        {
        y_profile=y_profile + *(yprof+i)/255;
        ii++,
        }
y_profile=y_profile/ii;
printf("y_profile=%lu\n",y_profile);

free( xprof );
free( yprof );

VG_QuitLib();
}
```

Program2 For Boxes With Orientation Angle

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <alloc.h>
#include <c:\iplib\inc\iplib.h>
#include "c:\example\profile\exam.h"

#define ALL_MASK 0xFF
#define MAX_NUM_INPUTS 4
#define FONT  "\\IPINTER\\SUPPORT\\FNT8X16"

ALOGLUT         *log_lut;
UBYTE           *trans_buf;
AFONT           *font_ptr;
UDWORD              *xprof,*yprof,*histo_buf;
WORD            err_code;
unsigned char                black = 0x00;
unsigned char   gray  = 0x80;
unsigned char    white = 0xff;
int             x_profile, y_profile;
int              i, j, k, ii, s, t, m, n;
int             *color1, *color2;
```

```
VOID main()
{
UBYTE       color;
int     a1, b1, a2, b2, a3, b3, a31, a32, b31, b32, a4, b4, a41, a42, b41, b42,
int         center_x, center_y;
long    I;
float   alpha;

switch ( err_code = VG_InitLib() )
        {
        case IP_8:
                break;
        case IP_BUSY:
                printf( "WARNING: IP-8 Board already in use \n" ),
                break;
        case IP_BOARD:
                printf( "ERROR: IP-8 board or BIOS not present\n" );
                printf( "       or wrong BIOS version\n");
                exit(0);
        case IP_DRIVER:
                printf( "ERROR: Driver (IP8DRV.SYS) not present\n" );
                printf( "       or wrong driver version\n" );
                exit(0);
        case MEM_POINTER:
                printf( "ERROR: IP-8 memory Contention\n" );
                printf( "       Couldn't create POINTERs to memory\n" ),
                exit(0);
        default:
                printf( "ERROR: IP-8 Library Initialization,\n" ),
                printf( "       Error code = %d\n", err_code );
                exit(0);
        }

VG_Reset(YES);
VG_SetInterlaceOutSt(ON);
VG_SetVideoSource(1, ALL_MASK);
VG_SetDispType(LIVE_VIDEO);

printf("Hit 'c' to continue...\n");
for(i=0;;i++)
{
if(getch()=='c')
break;
}
```

```
VG_SetFrameBufConfig( SGLE );
VG_SetAccessBufSt(ON),
VG_SetVideoKeyMode(ONLY_IP_OUT),
VG_SetGenLockSt(ON),

VG_SetYAxisDirection(Y_AXIS_DOWN);
VG_SetScanMode(PROGRESSIVE);
VG_SetDispType(MEMORY_VIDEO);

VG_SetKeyComp(MEMORY);
VG_SetKeyOut(LIVE),
VG_SetKeyType(SPECIAL),

VG_SetGrabRegUpd(DIRECT),
VG_SetGrabField(EVENBOTH);
VG_SetGrabFrameBuf( 0 );
VG_SetGrabSourcePos( 0, 0, _VG_XDispSize, _VG_YDispSize );
VG_SetGrabDestPos( 0, 0 );
VG_SetImageSourcePos( 0, 0, 0, _VG_XDispSize, _VG_YDispSize );

xprof=(UDWORD *)malloc( _VG_XDispSize*sizeof(UDWORD) );
yprof=(UDWORD *)malloc( _VG_YDispSize*sizeof(UDWORD) );
histo_buf=(UDWORD *)malloc( 256*sizeof(UDWORD) );
trans_buf=(UBYTE *)malloc( 256*sizeof(UBYTE) );
log_lut=(ALOGLUT *)malloc( 256*sizeof(ALOGLUT) );

font_ptr = VG_OpenFont( FONT );
VG_SelectFont( font_ptr );

for(i=0, i < 640; i++) *(xprof+i)=0;
for(i=0; i < 480; i++) *(yprof+i)=0;

VG_SingleGrab();
while ( VG_GetGrabSt() == ON);

VG_CalcHisto( histo_buf );

VG_SetLogLUT( log_lut, color1, 1, 256, 0 );
VG_ModifHisto(1, 0, 255, 1000, histo_buf, trans_buf);
VG_SetPhyLUT( log_lut, 0 );

VG_SetImageSourcePos( 0, 0, 0, 640, 480 );
VG_SetImageDestPos( 0, 0, 0 );
VG_TranslateImage( trans_buf );
```

```
VG_SetObjectColor(&white, &white);
VG_CalcProfile( xprof, yprof );

a3=0; b3=0;

VG_SetWinType( 0, WINDOW_PTR );
VG_SetWinOrig( 0, 0, 0);
VG_SetWinSize( 0, _VG_XDispSize, _VG_YDispSize );

VG_SetWinCtrl( 0, 1, 1, XPRIO, ROINC );
for( I = 0; ; I++ )
        {
        VG_GetWinPixel( 0, &color );
        a1=I / _VG_XDispSize;
        b1=I % _VG_XDispSize;
        if(color == 255) break;
        }

VG_SetWinCtrl( 0, 1, 1, XPRIO, ROINC );
for( I = 0; ; I++ )
        {
        VG_GetWinPixel( 0, &color );
        if(color == 255 )
        {
        a31=I / _VG_XDispSize;
        b31=I % _VG_XDispSize;
        if(b31 > b3)
        {
        a3=a31; b3=b31;
        }
        }
        if(I/_VG_XDispSize == 300) break;
        }

VG_SetWinCtrl( 0, 1, 1, YPRIO, ROINC );
for( I = 0; ;I++ )
        {
        VG_GetWinPixel( 0, &color ),
        a2=I % _VG_YDispSize;
        b2=I / _VG_YDispSize;
        if(color == 255) break;
        }

/*
```

```
VG_SetWinCtrl( 0, 1, 1, YPRIO, ROINC );
for( I = 0, , I++ )
        {
        VG_GetWinPixel( 0, &color ),
        if(color == 255 )
        {
        a41=I % _VG_YDispSize;
        b41=I / _VG_YDispSize;
        if(a41 > a4)
        {
        a4=a41, b4=b41,
        }
        }
        if(I/_VG_YDispSize == 500) break;
        }
                                        */

center_x=(a2+a3)/2,
center_y=(b2+b3)/2;
alpha=90 0 - atan((float)abs(a1-a2)/(0.75*(float)abs(b1-b2)))*180/3.1415926;
/*
printf( "a1 = %d, b1 = %d\n", a1, b1 );
printf( "a2 = %d, b2 = %d\n", a2, b2 );
printf( "a3 = %d, b3 = %d\n", a3, b3 );
printf( "a4 = %d, b4 = %d\n", a4, b4 ); */
printf("center_x=%d, center_y=%d\n", center_x, center_y);
printf("x_profile=%d\n",profile(a1, a2, b1, b2));
/* printf("x_profile=%d\n",profile(a3, a4, b3, b4)); */
printf("y_profile=%d\n",profile(a1, a3, b1, b3));
/* printf("y_profile=%d\n",profile(a4, a2, b4, b2)); */
printf("The box has a %f degree orientation error.\n", alpha);

free( xprof ),
free( yprof ),
free( histo_buf ),
free( trans_buf );
free( log_lut ),

VG_CloseFont( font_ptr ),
VG_QuitLib(),
}

profile(int s1, int s2, int t1, int t2)
{
```

```
int profile;
long X1, X2, X;

X1=(long)(s1-s2)*(s1-s2);
X2=(long)(t2-t1)*(t2-t1),
X=X1+X2;

printf("X1=%ld, X2=%ld, X=%ld\n", X1, X2, X),

profile=sqrt(X);

return(profile),
}
```

APPENDIX E
Robot Program

```
SPEED FAST
MOVE PIUP
APPRO PICKUP, 1
OPEN
SPEED SLOW
MOVE PICKUP
DELAY 1
DEPART 1
SPEED FAST
MOVE PIUP
MOVE UNLD
APPRO ULOAD, 1
SPEED SLOW
MOVE ULOAD
FINISH
CLOSE
DELAY 1
SPEED FAST
MOVE UNLD
STOP
$
```

APPENDIX F
Stability Analysis

This stability study only considers static forces. The dynamic forces occurred during the transportation of a fully loaded pallet are not considered.

The ability of a box maintaining in its position is determined by how big the minimum force needed to make it unstable. There are two situations which are considered unstable for a box: sliding or toppling. The stability factor is defined as the minimum force needed to either topple or slid a box. Let's look at the simplest case first: one box on a pallet (see Figure F.1).

Case 1. Single Box (Figure F.1)

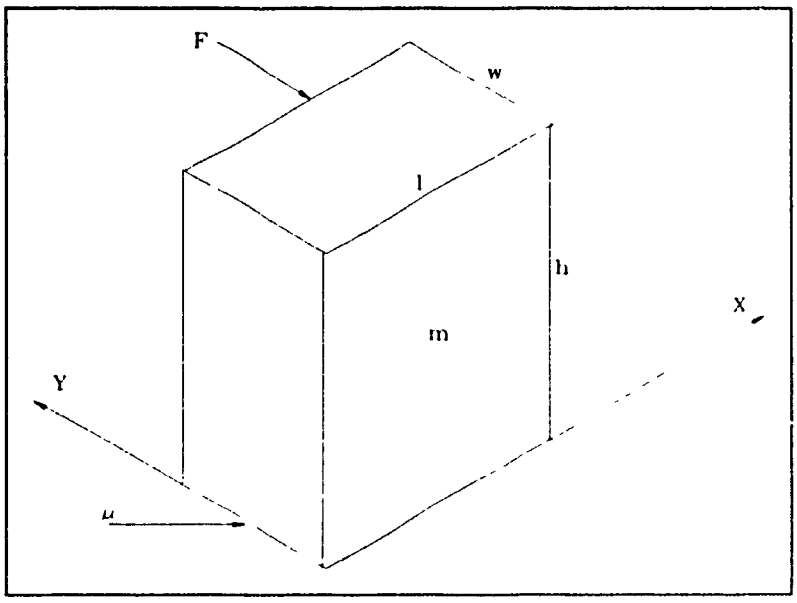


Figure F.1 Single Box Situation

a Sliding Situation

The minimum force needed to make a box slid is the friction force generated between the surfaces of the pallet and the box.

$$F_s = mg^*\mu \qquad (F.1)$$

b Toppling Situation

The minimum force needed to topple to box is calculated as follows:

$$F_t * h = mg * (w/2); \quad F_t = mg * (w/2h)$$

when $\mu > w/2h$, that is, $h > w/2\mu$

Let d donate the density of the box, then we have:

$$F_t = mg * (w/2h) = d * w * l * h * (w/2h) = d * w^2 * l/2 \quad (F.2)$$

It is noticed that the stability of a box is only influenced by its density, width, and length. It is interesting to find that the height does not contribute to stability as it appears to. The reason is. on one hand, as the height increases, the centre of gravity of the box moves up. Therefore, the stability of the box will be reduced. On the other hand, as the height increases, the volume of the box increases too. As a result, the box will be heavier. So, the stability increases. These two opposite affects cancel each other so that the influence on the stability from the height of a box is eliminated.

Let's compare the force F_t and F_s:

when $F_t > F_s$ $\quad mg * (w/2h) > mg^*\mu$

that is: $\mu < w/2h$, or $h < w/2\mu$

box will slide.

when $F_t < F_p$ $\quad mg * (w/2h) < mg*\mu$

that is : $\mu > w/2h$, *or* $h > w/2\mu$

the box will topple.

when $F_t = F_p$ $\quad \mu = w/2h$, or $h = w/2\mu$

toppling and sliding will happen at same time.

Example: $\quad w = 12,\ h = 30,\ \mu = 0.5$

$F_t = 0.5mg,\ F_t = 0.2mg$

Since $F_{min} = F_p$ the box will topple.

Case 2. Double Boxes

Situation 1. Force is along Y axis

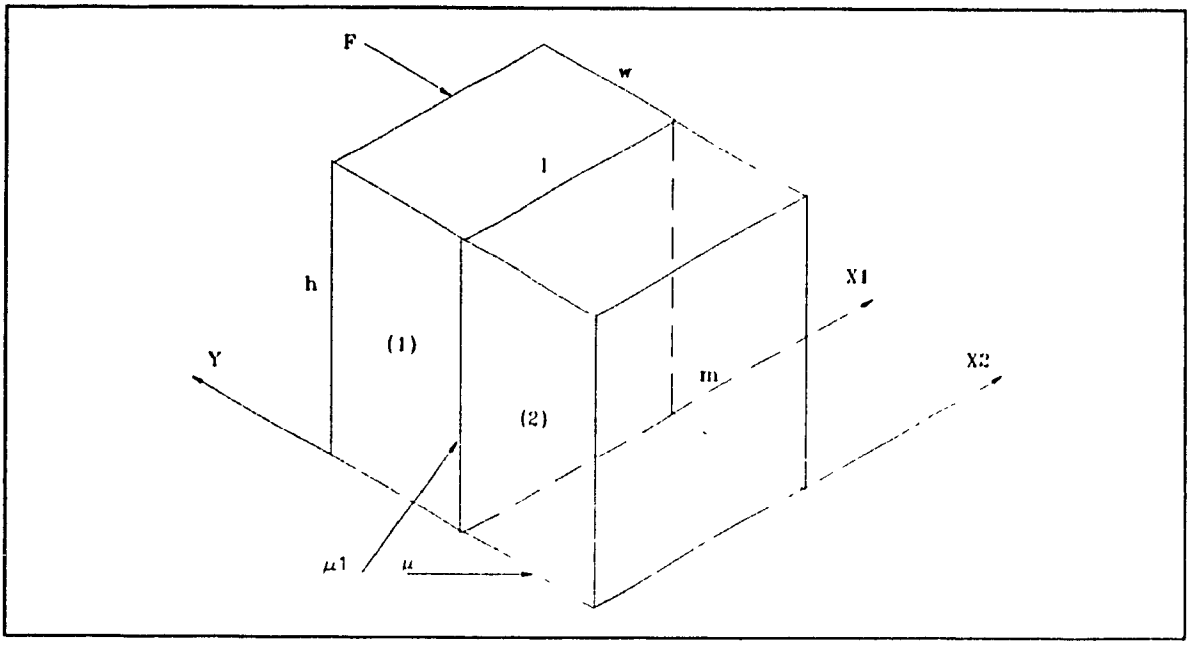


Figure F.2 Situation 1: Both Sliding

a Both sliding (Figure F 2)

$$F_{ss} = 2\mu mg \qquad (4.5)$$

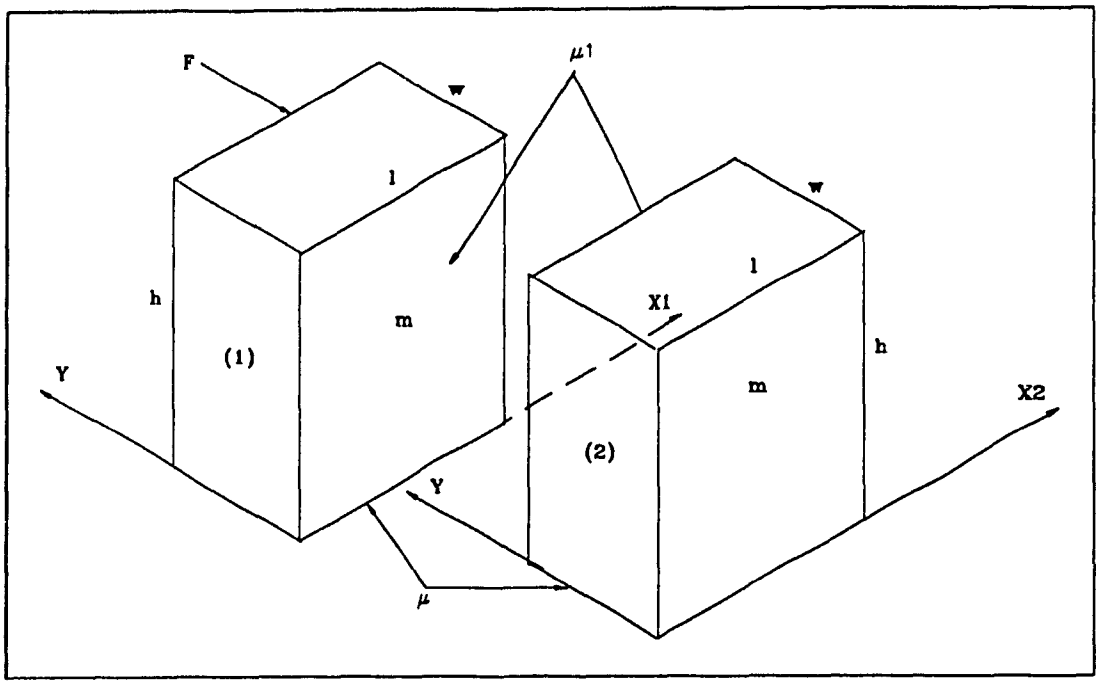


Figure F.3 Situation 1: Both Topping

b. Both toppling (Figure F.3)

Consider the situation where both boxes have edge contact with the supporting (pallet) surface at axis X1 and axis X2.

About X1: $F_a * h = mg * w/2 + F_1 * h$

$$F_1 = F_a - mg * w/2h \qquad (a)$$

About X2. $F_1 * h = mg * w/2 + F_1 * \mu_1 * w \qquad (b)$

Substitute (a) into (b):

$$F_a = mg * (w/2h + w/(2h - 2\mu_1 w)) \qquad (F.3)$$

c. First toppling and second sliding

First box: $F_a * h = mg * w/2 + F_1 * h$

Second box: $F_1 = f = \mu * N = (F_1 * \mu_1 + mg) * \mu$

$$F_1 = mg * \mu/(1 - \mu * \mu_1)$$

$$F_a = mg * (w/2h + \mu/(1 - \mu * \mu_1) \qquad (F.4)$$

Situation 2. Force is along X axis (Figure F.4)

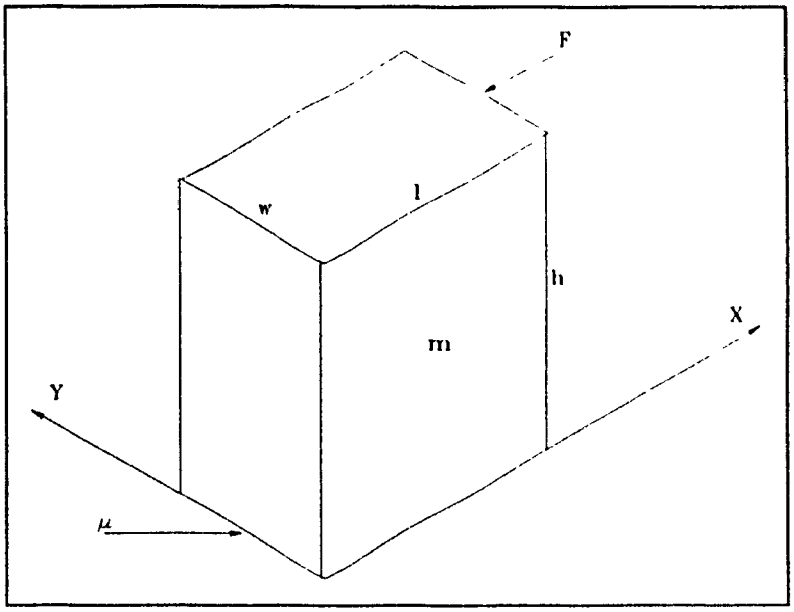


Figure F.4 Situation 2

a. Sliding Situation

$$F_s = \mu * mg$$

b. Toppling Situation

$$F_t = mg * l/2$$

Example $\mu = 0.5,\ \mu_1 = 0.5,\ w = 12,\ h = 30,\ l = 20$

$F_{n1} = mg,\ F_{n2} = 0.45mg,\ F_{n3} = 0.87\ mg$

$F_1 = 0.5mg$

$F_f = 0.33mg$

The minimum force is find as: $F_{min} = F_f = 0{,}33mg$ in Situation 2. Therefore, the box is more likely to topple by Y axis.

Case 3. Two Boxes on Top of One (Figure F.5)

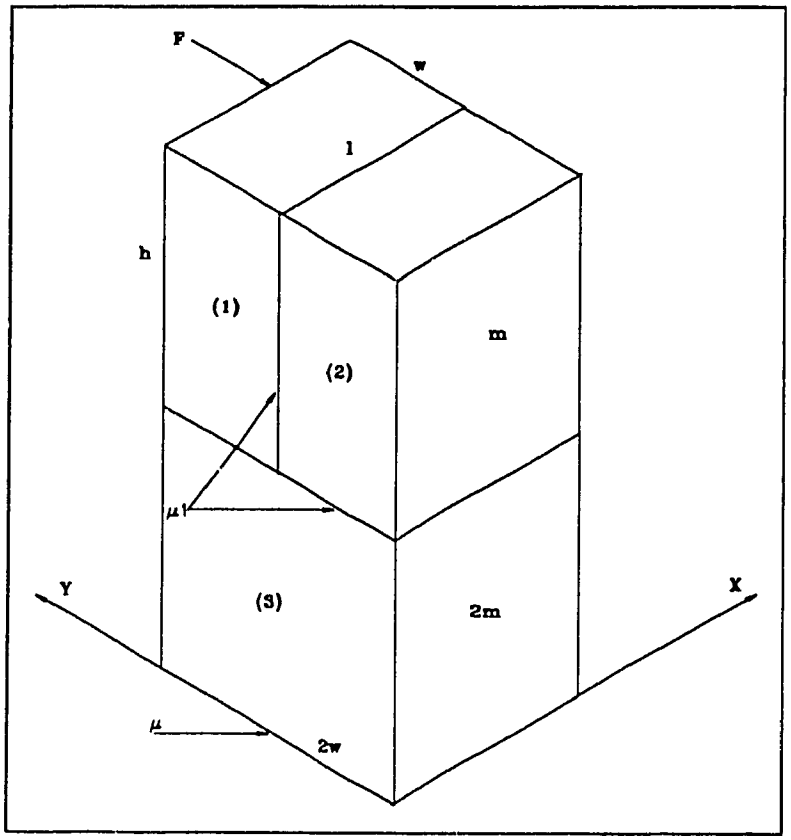


Figure F.5 Case 3

For the two boxes on the top, it will be the same as Case 2. $F_{min} = 0.33mg$

For the box on the bottom:

Sliding: $F_s = 4\mu mg$ *(F.5)*

Toppling: $F_t * h = 4mg * w$

$F_t = 4mg * w/h$ *(F.6)*

Example: $\mu = 0.5,\ \mu_1 = 0.5,\ w = 12,\ h = 30,\ l = 20$

$F_s = 2mg,\ F_t = 1.6mg$

Since the maximum friction between the two boxes $f_{max} = 2\mu mg = mg$, less than either F_t or F_s, the bottom box will never topple nor slid

$F_{min} = 0.33$

Case 4. One Box on Top of Two (Figure F.6)

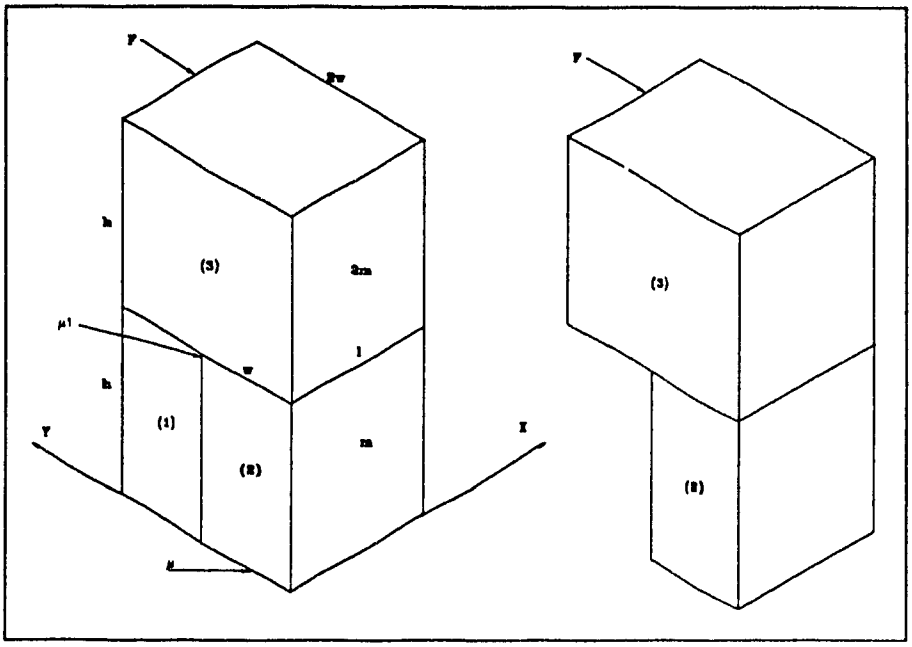


Figure F.6 Case 4

For the box on the top (box 3) :

Sliding: $F_s = 2\mu mg$

Toppling: $F_t = 2mg*w/h$

For the box at the right corner (box 2): This box will topple or slid with box 3.

Sliding: $F_s = 3\mu mg$

Toppling: $F_t*2h = 2mg*w + mg*w/2$

$F_t = 1.25mg*w/h$

Example $\mu = 0.5$, $\mu_1 = 0.5$, $w = 12$, $h = 30$, $l = 20$:

$F_{3t} = 0.8mg$

$F_{3s} = mg.$

$F_{2t} = 0.5mg$

$F_{2s} = 1.5mg$

The minimum force $F_{min}=0.5mg$

Compare these four cases, the followings are found:

• Case 1 and Case 2 show that when two pieces are put together, the overall stability is increased because of the friction force between the boxes.

• Case 3 and Case 4 show that when there are two layers, to put the one with higher stability on the one with lower stability will give better overall stability.

• Factors affect stacking stability:

a density of the box material

b. width and length of the box

c. friction coefficients between boxes, box and pallet

d. box height only has effect on sliding, not on toppling.

Based on the conclusion, a rule of thumb determining the loading pattern during palletization can be derived: when a pallet is loaded by several layers of boxes, the layer with better stability should be placed on the top of others. In most cases, the layer contains smaller boxes has worse stability.