A STUDY ON CRATE SIZING, INVENTORY AND PACKING PROBLEM

LEE SHIH JIA

(B.Sc., Cornell)

(M.Sc., NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously**.**

[illegible]

Lee Shih Jia

07 April 2015

Acknowledgements

This thesis is accomplished with tremendous help and guidance from both of my supervisors, A/Prof Chew Ek Peng and A/Prof Lee Loo Hay who provided relentless support and encouragement throughout the years.

To my family whom I could not spend more time with on many family occasions, I could not express more gratitude for their kind understanding and emotional support.

Lastly, we are also grateful to Company S Singapore Pte. Ltd. for the inspiration and data provided for the study of this thesis.

Table of Contents

Summary

This thesis is a formal study of an actual problem faced in the industry for crate sizing, inventory and packing. The problem is relevant because many manufacturers face the problem of proper planning, operations and evaluation of their product packaging and packing processes. Since most products will need to be packed before being distributed to customers, inefficient practices will lead to higher cost and time expended. In this final process, many aspects of the way the products are packed can be studied and improved. The industrial crate sizing problem addresses the problem of determining what are the optimal crate sizes and also how many types of crates are ideal. There is no formal study to scientifically investigate the crate sizing problem yet. Therefore, in this study, we first define and formalize the problem of crate length optimization faced by the industry, and represent it as an MIP model. The second problem is extended from the crate length optimization problem which considers the inventory and we formulate it as a non-linear MIP model. The tradeoff between inventory cost and wastage cost from fitting products into crates is considered in the objective function. The non-linear MIP model is generally difficult to solve, but by exploiting the structure of the problem, we are able to solve it using dynamic programming because the problem has the special property of Bellman's Principle of Optimality. We further extend the crate size optimization problem by considering the width and height dimensions of the crate in addition to the length dimension. In this problem, the products are in rolls; hence the crates are rectangular boxes with square cross section which means the crate width and height are the same. The problem is non-trivial and cannot be solved using any solvers for a reasonable size problem. Enumeration method can only be used to solve small size problems but is computationally intractable for larger problems. Therefore we propose using a Hungarian based genetic algorithm to solve the problem. Hungarian method is used to preserve the good neighbourhood structure which is required for genetic algorithm to perform well. When the parents are selected for crossover, it is treated as an assignment problem where the gene of a parent is matched to the closest gene of another parent before applying the crossover operations. In addition to the crate sizing and inventory problem, this study also looks into the packing of the crates into containers. After finding the crate size and crate types, we also need to pack the crates into shipping containers for distribution. We solve the problem of packing crates into containers by using a bin packing algorithm with an improvement heuristic. This approach utilizes the information of the solutions from the previous iteration to create good potential columns for the next iteration. Overall, this study has covered several of the important aspects which can be improved for a real industrial-based problem and also proposes different methods to tackle and solve the crate sizing, inventory and packing problem.

List of Tables

List of Figures

List of Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
BPP	bin packing problem
CV	coefficient of variance
GA	genetic algorithm
IP	integer programming
LP	linear programming
MSSCSP	multiple stock size cutting stock problem
MI	marginal improvement
MIBR	marginal improvement by random
MIBS	marginal improvement by sequence
MILP	mixed integer linear programming
MIP	mixed integer programming

List of Notations

Crate Sizing Problem

Roll

N	Number of rolls
w_i	Roll width *i*
d_i	Roll height (diameter) *i*
μ_i	Mean demand of roll *i*
σ_i	Standard deviation of demand of roll *i*

Crate

K	Number of crate types
L_k	Crate length *k*
W_k	Crate width (or height) *k*
L_{min}	Minimum crate length
L_{max}	Maximum crate length
W_{min}	Minimum crate width
W_{max}	Maximum crate width
x_{ik}	1, if roll *i* is assigned to crate length *k* and 0, otherwise
y_{ik}	Loss of length inside crate when roll *i* is assigned to crate length *k*; 0 otherwise

z_{ik}	1, if roll *i* is assigned to crate *k* of length L_k and width W_k; 0 otherwise
$s_{a,\ldots,b}$	Pooled risk of standard deviation (safety stock) of demand of roll widths *a* to *b*

Constants

p	Penalty cost factor
h	Inventory holding cost factor
P	Minimum padding requirement inside the crates
M	A very large integer number

Dynamic Program

n	Stage of the dynamic program
N	Total number of stages (roll width sizes to be considered)
x_n	State of stage *n*
a_n	Decision variable of stage *n*

Assignment problem

x_{ij}	1 if gene *i* of chromosome 1 is matched to gene *j* of chromosome 2, otherwise 0.
c_{ij}	Cost of matching gene *i* of chromosome 1 to gene *j* of chromosome 2

Bin packing problem

L	Container length

W	Container width
H	Container height
K	Total number of containers available
S	Size of the rectangular layer (the floor of the container)
l_i	Length of item i
w_i	Width of item i
s_i	Size of item i
n	Total number of items i
x_i	Geometrical location of item i (left-coordinate)
y_i	Geometrical location of item i (back-coordinate)
β	Bin (container) index
l_{ij}	1 if item i is in the left of item j; 0 otherwise
b_{ij}	1 if item i is at the back of item j; 0 otherwise
c_{ij}	1 if $\beta_i < \beta_j$; 0 otherwise
d_i	Demand of items i of size l_i and w_i to be packed
a_{ij}	Number of items i of size l_i and w_i packed in layer j
A_j	Packing pattern p_j
X_j	Number of layers packed with pattern p_j
J	Total number of distinct feasible cutting patterns p_j

h_j	Height of layer j
Y_k	Number of packed bins
e_{jk}	Number of layer type j in bin k

1 Introduction and Overview

Nearly every product has to be packed and transported in the course of its distribution process. Although this is typically the last operation in any manufacturing activity, it plays a vital role in ensuring that the product is delivered to the customer in sound condition. Packing and packaging serves several purposes such as protection, identification, transportation, storage and stacking. The packaging should be secure and able to protect the goods adequately during transportation at suitable cost. However, there are many challenges encountered in various stages such as planning and evaluation, packing materials, space utilization, warehouse and storage and freight issues in order to achieve minimum cost. Specifically, packaging-wise, decisions have to be made regarding what packaging types to design as well as which sizes to order and stock in order to cater to demand variability. Packing-wise, decisions also have to be made on how to pack into the shipping containers.

A research was conducted by Peerless Research Group on behalf of Logistics Management and Modern Materials Handling magazines for Packsize International in June 2013[1]. Referring to Figure 1.1, it is revealed that 38% of the companies noted that their packaging and shipping costs have increased by 5% to 20% in the past year while 53% saw no change and a small minority of 9% saw a decrease. In addition, almost all of the companies (94%) use different sizes of packaging and the top three expenses involved are packaging materials, labour and shipping costs. It can be seen that packaging and shipping costs are a cause of concern for many manufacturers.

[1]Source: http://www.mmh.com/images/site/Packsize_Brief_F.pdf

There are many aspects of packaging and packing that can be studied in order to improve the process and keep costs as low as possible. As such it is worthwhile to study the optimization of packaging and shipping processes to increase overall efficiency and reduce total cost.

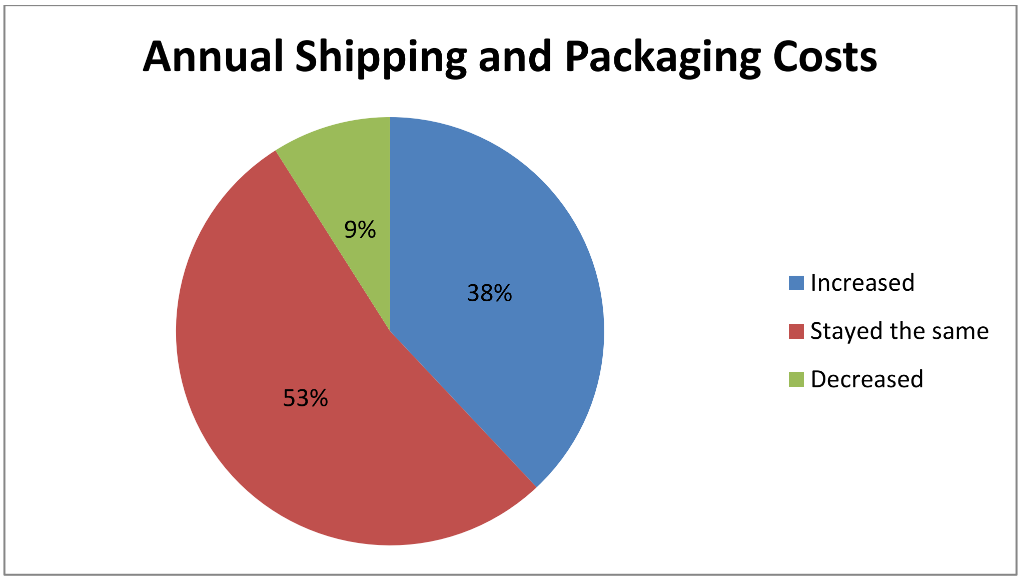

Figure 1.1 Survey on annual shipping and packaging costs in 2013

1.1 Background and Motivation

The research is based on a real industrial problem faced by Company S, a multinational corporation in the applied chemistry industry. Company S is the leading manufacturer in performance films which serve as interlayers for laminated glass, automobile and building window films, protective and conductive films and others used in a myriad of architectural and industrial applications. Their main products are polyvinyl butyral (PVB), ethylene vinyl acetate (EVA), and thermoplastic polyurethane (TPU) which are sold worldwide from their headquarters based in USA, Belgium, Brazil and Shanghai.

Due to the nature of the products, the chemical films are sold in cylindrical rolls of various lengths and thicknesses. The rolls are customized according to customer’s specifications. They are also heavy and long hence the rolls are packaged in big wooden crates which are expensive. The wooden crates serve as protection from damage during the transportation process. Besides protection, the crates enable easy identification, lifting by forklift trucks and storage and warehousing. Company S stocks and uses a number of standard crate sizes for roll packaging. Currently, the company has designated four types of crates to cater to the demand. Because there are only a few standard crate sizes compared to the number of actual demand of roll sizes, there is bound to be empty space inside the crates once the rolls are fitted into individual crates. Each roll is assigned to a standard crate size which can fit the roll with the least amount of space wastage. Inside the crates, the empty space between the roll length and the crate end is filled with Styrofoam paddings to disallow the roll from movement and to prevent damage during transportation.

When the rolls are finished packing into crates, they are then loaded into shipping containers ready for delivery to customers by sea. There are a few choices of shipping containers, namely the 20’ and 40’ containers for regular type of rolls. For rolls that require refrigeration, there are reefer containers.

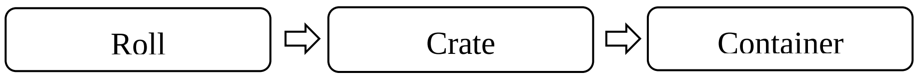

Figure 1.2 Product packing hierarchy

The product is actually a large sheet of thin film before it is rolled up. After rolling the film, the width of the film becomes the roll width whereas the length of the film makes up for the rolled up diameter or roll height. The length of the film can be customized to a few types of cut length. The product is sold and transported as cylindrical rolls. Together, both the roll width and roll diameter/height dimensions specify the roll type ordered by customers. Customers can order one or more types of rolls and the quantities needed for each type.

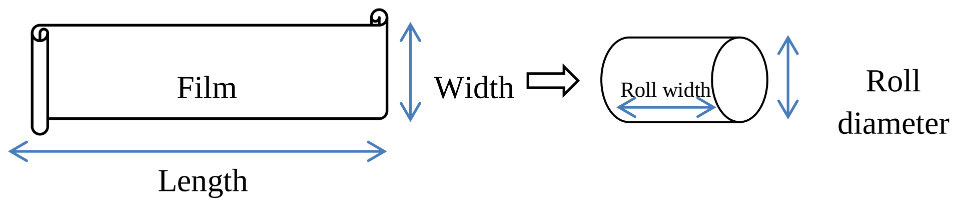

Figure 1.3 Product dimensions before and after rolling

When the rolls are fitted into the crates, it should be noted that the length of the roll corresponds to the rolled up diameter of the roll and the diameter of the roll depends on the thickness of the film type. Because the diameter of the roll is a circle, the cross sectional area of a crate is a square. It can be assumed that the crate height and width are equal to accommodate the cylindrical roll. Meanwhile, the roll width is the dimension that is parallel to the length of the crate.

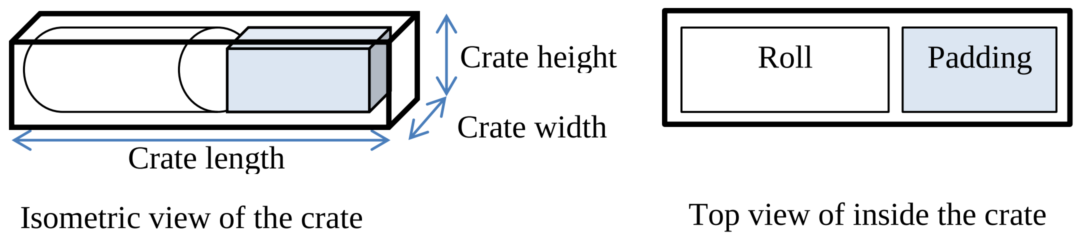

Figure 1.4 Packaging of roll in crates

The crates are packed into the container. Because most of the crates are very long and do not fit across the width of the container, they are packed along the length of the container. Depending on the dimensions of the crates, rotations can be allowed to maximize on space utilization. Empty space inside the container is filled with plastic air bags to cushion the impact from transportation so as to avoid damage to the wooden crates. Unutilized space and inefficient packing can lead to unnecessary wastage in total freight cost. Depending on the size and type of customer demand, each order is loaded into as few containers as possible to save on shipping cost.

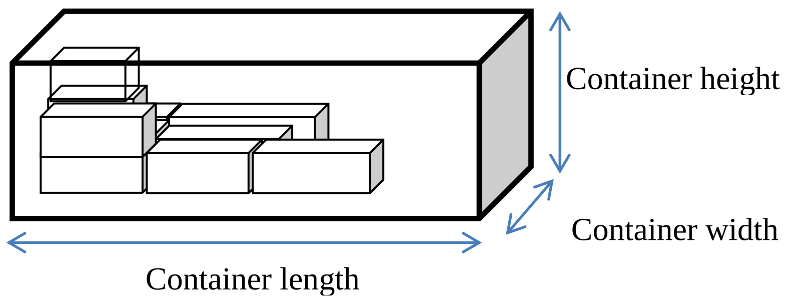

Figure 1.5 Packing of crates in shipping containers

From the abovementioned, the research is motivated to provide a more efficient solution to strategize packing problems. In the first packing step of packaging rolls into crates, there are decisions on planning the standard crate sizes and the number of crate types. If the crates are too big, there will be a waste of crate materials, space inside the crates and eventually in the shipping containers. Also, the number of crate types can have a huge impact on the wastage cost. If there are many types, the rolls will fit better but there will be higher stocking and inventory costs to cater to demand uncertainty. On the other hand, if there are few types, the rolls will fit worse but there will be savings in inventory cost. On the practical side, it makes sense to have a manageable number of crate types in order to reduce complexity and handling of operations.

Currently the usual industrial practice involves reviewing sizes from historical data and in many instances, experience and intuition by industry experts play an important role. One method is to determine the sizes with the highest demands and assign one size for each peak demand. Another method is to divide the demand sizes into a few equally spaced intervals. The current practices have certain limitations as they do not consider inventory cost. There is no formal study on investigating the choice of crate sizes in order to minimize total costs and also the ideal number of crate sizes. The optimization of the crate sizes is interesting enough to warrant a formal study to find a compromise between space wastage and inventory cost.

The problem is challenging because the crates have three dimensions i.e. the crate length, width and height. Fortunately, due to the constraints of the problem, the crate width and height can be treated as equal. Essentially, solving the two dimensions is analogous to solving all three crate's dimensions. Beyond the packing of rolls into crates, there is potential savings in the containerization process as well. Container loading can be improved to better pack the crates into the shipping containers.

The questions we will like to address in this research are as follows: Firstly, what type of crates will be suitable for packing the rolls in and what sizes they should be, secondly, how many types of crates would be optimal and thirdly, how to pack the crates to the containers so as to minimize total cost from the wastage cost of the packing of rolls into crates and subsequently into containers and also inventory and shipping cost.

The questions above are addressed to solve the overall problem. Initially, the optimization of crate size is built from the basic problem involving a deterministic problem of finding optimal crate lengths only (as crate length is the naturally the longest dimension of the three dimensions and highest contributor to the total loss) with a mixed integer programming problem formulation (MIP). However, given demand uncertainty, the MIP is not easy to solve and as such, dynamic programming approach is applied to the problem to solve both the crate lengths and crate types optimally. Thereafter, the problem is further extended to find the optimal crate dimensions for crate length, width and height simultaneously. Genetic algorithm approach is employed in this extended problem. Finally, an improvement method is applied to improve the packing process of crates into shipping containers for sea freight.

In this research thesis, we have made several contributions namely:

1. We are able to define and formalize the problem of crate length optimization faced by the industry, and represent it as an MIP model. Using the historical data, we are able to find the optimal crate lengths given the number of crate types.
2. We extend the problem by considering demand uncertainty and introduce the safety stock consideration into the problem. While the problem can be modelled as a non-linear MIP model, it has a good property that exhibits the Bellman's Principle of Optimality. This

allows the problem to be solved efficiently by using dynamic programming.

3. We further extend the crate size optimization problem by considering width and height dimensions of the crate in addition to the length dimension. As the width and the height are the same, the problem can be modelled as a two dimensional problem. The problem is non-trivial and cannot be solved using any solvers for a reasonable size problem. We propose a Hungarian-based genetic algorithm to solve the problem. Hungarian method is used to preserve the good neighbourhood structure which is required for genetic algorithm to perform well.
4. We solve the problem of packing the crates into containers by using a bin packing algorithm with an improvement-based heuristic approach. This approach utilizes the information of the solutions from the previous iteration to create good potential columns for the next iteration.

1.2 Organization of the Thesis

This thesis consists of 6 chapters. The rest of the thesis is organized as follows:

Chapter 2 first discusses related works and literature review of the crate sizing problem, also known as box sizing problem, and then the second part reviews the bin packing problem (BPP).

Chapter 3 describes the crate sizing problem in one dimension, i.e. the crate length with and without inventory cost consideration. The problem is defined and then solved using integer programming and dynamic programming approaches.

Chapter 4 extends the crate sizing problem from Chapter 3 where both crate length and crate width/height are now considered for optimization. In this extended problem, genetic algorithm approach is used to find the optimal solution for crate dimensions with inventory cost consideration.

Chapter 5 delineates a packing algorithm to pack the crates into shipping containers. The recommended approach is layer packing using packing heuristics with improvement-based approach for improvement.

Finally, Chapter 6 examines some potential future research directions and conclusions derived from the study of this work.

2 Literature Review

There are many aspects of logistics in the packaging, packing processes and delivery of products to customers ranging from packaging type and sizing, warehouse and storage, to bin packing or containerization into shipping containers. This segment is organized into two parts, where we first review the crate sizing problem and related problems then bin packing problem in 1D (one-dimensional), 2D (two-dimensional), 3D (three dimensional) and others.

2.1 Crate Sizing Problem

From the literature, there has been research on the packaging problem and related problem such as box sizing or crate sizing problem. Some related works in the literature include the size selection problem, standardization, and assortment or catalogue problem.

In the standardization problem, a standard size is smaller or larger than the desired size on the control dimension. If the dimension is not the same, there is an adaptation loss. The paper by (Bongers, 1982) and book by (Bongers, 1980) discussed many ways of tackling the standardization problem such as recursion formula for loss function and adaptation loss. In an applied garment industry problem, (Tryfos, 1985) tackled the issue of measurement of a given number of sizes to apparel in an effort to minimize discomfort and maximize expected sale. The author presented an algorithm to design for optimal sizing system based on normal distribution of the population sizes by developing the general necessary conditions for optimization via grouping in one controlled body dimension mainly in one dimension. However the result was not conclusive. (Pentico, 1986) authored a comment on Tryfos' paper where he noted that the problem of optimal sizing is not a new one, but rather it is a special case of the assortment and catalogue problem which has been researched. Thus, (Vidal, 1994) extended the study and presented an algorithm to determine the numbers and dimensions of sizes of apparels to maximise profit. The author developed an interactive one variable bisection search algorithm that solves the problem by giving the optimal solution.

Meanwhile, the assortment or catalogue problem is to decide a limited subset of a large discrete set of possible sizes to stock. Given a set of sizes of products and their demands, generally only a selected subset of box sizes will be stocked due to factors such as space and inventory cost. (Pentico, 2008) in his paper presented a review of assortment or catalogue problem works published over the last 50 years from 1957 to 2007. The author classified the studies into one dimensional and multi-dimensional where different methodologies are used. Many of the research works also used heuristics to solve the problem. Apart from that, the author in his paper also touched on some related problem such as standardization, substitution and revenue or yield management. (Hinxman, 1980) authored a survey paper on trim-loss and assortment problems.

(Kasimbeyli, Sarac, & Kasimbeyli, 2011) presented a one dimensional cutting stock and assortment problem where the total number of roll sizes to be stocked was determined using linear integer programming and then the cutting stock patterns required to satisfy the demands were determined. However, the problem differs from our problem because the rolls are cut into different sizes in their problem whereas we only assign one roll to each crate to find the loss in length in the crate length optimization model with inventory consideration. (Yanasse, 1994) proposed a search strategy for a 1 dimensional assortment problem. The strategy uses and updates a lower bound contour until a satisfactory solution is achieved. (Gasimov, Sipahioglu, & Saraç, 2007) presented a 1.5 dimensional cutting stock and assortment problem. A 1.5 dimensional cutting stock problem is where the length of a sheet is sufficiently large or considered infinite. The authors presented an MILP and new conic scalarization. (Li & Chang, 1998) proposed a new model to reformulate the assortment problem with less binary variables. (Li & Tsai, 2001) presented a fast algorithm to solve the two dimensional assortment problem and proved that it is computationally efficient. (Li, Chang, & Tsai, 2002) proposed a piecewise linearization technique to find the approximate global optimization for assortment problem. (Lin, 2006) presented a genetic algorithm for solving the two dimensional assortment problem. (Baker, 1999) proposed a spreadsheet model to determine which sizes to stock and formulated it as a shortest or longest path problem on a directed acylic network. (Gemmill, 1992) introduced a genetic algorithm to solve the assortment problem. In an industrial application, (Rajaram, 2001) considered the assortment problem in fashion planning to choose a mix of the merchandise to maximize expected profit and determine the inventory breadth and depth. Additionally, (Flapper, González–Velarde, Smith, & Escobar-Saldívar, 2010) discussed the assortment of products to stock if customers only order if the delivery is on time and maximize profit by considering inventory cost, setup cost and others. (Chen & Lin, 2007) approached the product assortment problem using a data mining method to decide on which products to display and their ideal shelf life.

The crate sizing problem can also be related to the box sizing problem. One of the earliest works on box sizing is by (Wilson, 1965) who presented a paper with the objective to select the optimum number and sizes of boxes which can minimize the total system cost where an integer programming formulation was given. The author used heuristics to generate the box sizes. The paper by (Korchemkin, 1983) also presented a heuristic approach but by first dividing the problem into smaller sub-problems to solve a minimum-cost packaging problem. Using genetic algorithm, (Wang, Wang, Ni & Cheng, 2011) introduced a genetic search algorithm model named Multi-parameter Optimization Design System of Package Container Size to solve the packaging problem that reduces logistics cost by determining the optimum inside and outside of the packages. The authors used genetic algorithm to search the optimal inside and outside package sizes during the packaging process to efficiently reduce the waste of space for container vessels, rate of transportation and quantity of storage resources. In an industrial based problem (Leung, Wong, & Mok, 2008) the authors presented a box sizing problem whereby they used genetic algorithms to design make-to-order carton sizes to fit products of different sizes in the apparel industry with the objective to minimize total distribution and packing costs. The problem presented differs from this paper as they pack multiple items into each carton whilst there is only one roll to each crate in our problem. (Wong & Leung, 2006) presented a box sizing problem whereby they would like to search for the best box design, the optimal set of cartons for combined order which minimizes the unfilled as well as the number of carton types in the apparel industry with the objective to minimize total distribution and packing costs. Similarly, (Xu, Qin, Shen, & Shen, 2008) presented an optimization framework to the box sizing problem caused by supply chain strategy changes using the Cut/Pack/Select (CPS) framework which decomposes the problem into several sub problems for simplicity. The authors used a combination of the CPS framework and IP model to determine the box sizes before the demand of product is fixed with the use of an existing heuristic algorithm, the container loading problem. Given the historical demands, the framework uses a top down approach and determines the sizes of the inner and outer boxes, the matching of the products to their corresponding boxes, and uses container loading to load the boxes into the container for shipment. Their problem differs in that there are inner and outer boxes as the problem involves packing multiple small products into inner boxes and then packing these inner boxes into outer boxes before being consolidated for shipping in containers. From a different perspective, (Zhang, Yuan, & Yuan, 2012) presented an algorithm to generate a function to determine expected waste space versus box size for box optimization. In order to do so, the authors' research showed the correlation between average waste space per box and box sizes for online next fit bin packing by enlarging interval distribution.

2.2 Bin Packing Problem

The sizes of the packaging boxes will impact the packing patterns and utilization of shipping containers for shipping to customers worldwide. It is closely related to the bin packing problem where the objective is to lower shipping costs. Cutting and packing problems is an actively researched topic. The paper by (Dyckhoff, 1990) provided a good typology of the various cutting and packing problems. (Wäscher, Haußner, & Schumann, 2007) produced an improved typology and bibliography of research applications. (Oliveira & Wäscher, 2007) discussed the many ways how cutting and packing problems can be modelled in LP formulation. There are two closely related problems called the cutting stock and bin packing problems because the difference is that in cutting stock, there are unlimited stock (bin) sizes to cut from whereas in bin packing, there are limited bins to pack into. Each problem is the reverse of the other.

One of the more prominent cutting and packing problems is cutting stock. (Coverdale & Wharton, 1978) and (Haessler & Sweeney, 1991) covered on the cutting stock problems and ways of solving them. (Gilmore & Gomory, 1961) presented the one dimensional cutting stock problem solution with LP and column generation, then (Gilmore & Gomory, 1963) reformulated the LP, proposed a rapid algorithm for knapsack problem, and modelled a paper mill problem with constraints modified for different parent length rolls and cost. (Gilmore & Gomory, 1965) extended the LP for two or more dimensions in addition to the corrugated box problem and sequencing problem. (Dyckhoff, 1981) presented a new linear programming approach as compared to the classical model from Gilmore & Gomory. (Sinuany-Stern & Weiner, 1994) discussed the one dimensional cutting stock problem using two objectives. In addition, (Vance, Barnhart, Johnson, & Nemhauser, 1994) solved the binary cutting stock problem by column generation and branch-and-bound. (Cui & Zhou, 2002) discussed on the special case of generating optimal cutting patterns for single-size rectangles. (Alves & Carvalho, 2008) presented an exact algorithm to solve the ordered cutting stock problem.

Besides cutting stock approach, bin packing problem is also widely researched. Exact solutions can be obtained via branch-and-bound algorithm as in (Martello & Vigo, 1998) and (Martello, Pisinger, & Vigo, 2000) for 2D and 3D problems respectively. The former work performed worst case analysis and found new lower bounds for the NP hard problem. It also obtained exact solution for cases of up to 120 pieces. Extension of this work to 3D managed to solve cases of up to 90 pieces.

There are a variety of approaches to cutting and packing problems. Two of the earlier heuristics for packing include first fit decreasing (FFD) where items are first placed in order of non-increasing weight and best fit decreasing (BFD) where items are put into best-filled bin that can hold them. (Berkey & Wang, 1987) also discussed heuristics to solve the packing problem with finite next-fit, finite first-fit, finite best-strip, finite bottom-left and hybrid first-fit heuristics. Many authors also tackle the packing problems in layers, shelves and stages. (Caprara, Lodi, & Monaci, 2005) introduced the first approximation scheme APTAS for two-dimensional shelf bin packing.

Many approaches using different types of algorithm and heuristics were developed to solve one dimensional bin packing problems. (Abidi, Krichen, Alba, & Molina, 2013) developed a genetic algorithm for the one dimensional bin packing problem. By using greedy algorithm, the first fit heuristics and randomly, the algorithm generates an initial population of chromosomes and performs a series of perturbations to improve load of all bins sequentially. On the other hand, (Toledo Suarez, Gonzlez, & Rendon, 2006) introduced a heuristic approach using interactive algorithm for offline one dimensional bin packing problem. The authors' algorithm is successful with the design of the algorithm bounded by the performance of the point Jacobi method by taking the problems as a matrix. (Bhatia, Hazra, & Basu, 2009) however presented a study on better fit heuristics for one dimensional packing where an existing object from a bin is replaced when the object can fill the bin better than the object replaced. The proposed algorithm behaves as offline as well as online heuristics but performs better than offline best fit decreasing heuristics and also online best fit heuristics.

Other methods such as stochastic approach for one dimensional bin packing were also studied by (Berkey & Wang, 1991) who presented a systolic based parallel approximation algorithm that obtains solution for one dimension bin packing problem. The authors' algorithm has an asymptotic error bound of 1.5 and time complexity of $\Theta(n)$. From the author's experimental study, the heuristic offers improved packing and execution performance over parallelization of two well-known serial algorithms. Similarly, (Anika & Garg, 2014) presented packing problem solution by parallelizing generalized one dimensional bin using MapReduce. This optimization is attained by packing a set of items in as fewer bins as possible. The efforts have been put to parallelize the bin packing solution with the well-known programming model, MapReduce which is supportive for distributed computing over large cluster of computers. The authors have proposed two different algorithms using two different approaches, for parallelizing generalized bin packing problem. The results obtained were tested and it was found that by working on the problem set in parallel, significant time efficient solutions for bin packing problem were obtained. Aside from that, (Kao & Lin, 1992) introduced a new stochastic approach called annealing genetic algorithm for one dimensional bin packing problem where simulated annealing is used for exhaustive and parallel treatment of the problem and to increase the probability of finding global minimums. The results showed that the solution quality of this approach is equal if not better than first-fit-decreasing with no non-monotone anomaly found.

Using heuristics, many similar approaches for one dimensional bin packing have also been used for two dimensional bin packing problem for optimization. (Bansal, Lodi, & Sviridenko, 2005) presented a generalization of the classical bin packing problem with orthogonal packing without rotation using guillotine cuts. Guillotine cuts is a well-studied and frequently used constraint where every rectangle in the packing must be obtainable by recursively applying a sequence of edge to edge cuts parallel to the edges of the bin. The author proved that guillotine two dimensional bin packing problem admits an asymptotic polynomial time approximation scheme which is in sharp contrast with the fact that general two dimensional bin packing problem is APX-hard. The author was also able to show a structure of approximating general guillotine packing by simpler packing which could be of independent interest.

(Bekrar & Kacem, 2008) explored the use of two heuristics for two dimensional bin packing using best shelf and non-shelf heuristic filling. Using strip and bin packing with guillotine cuts by packing a set of rectangular bins on one strip of width *W* and infinite height or bins of width *W* and height *H*, the items are packed without overlapping and need to be extracted by a series of cuts that go from one edge to the opposite edge (guillotine constraint). The results obtained by the author shows that the two heuristic algorithms are complementary.

(Pargas & Jain, 1993) presented a stochastic optimization approach to a two dimensional bin packing problem for a rectangular area similar to genetic algorithm or simulated annealing algorithm. Using a parallel processing algorithm with processes of evaluating the length of layout; near perfect load balancing is achieved with a minimum of 80% efficiency or utilization based on bin length.

(Omar & Ramakrishnan, 2011) proposed evolutionary particle swarm optimization algorithm (EPSO) for solving non-oriented two dimensional bin packing problem. The author deals with a set of rectangular pieces that need to be packed into identical rectangular bins where the rectangular pieces are only allowed to rotate 90^{o} without overlapping. Although comprehensive testing methodology was presented, the results only indicated improved initial results and the author is currently working on improvement for the proposed EPSO.

On the other hand, (Cao & Kotov, 2011) presented a two dimensional bin packing problem to minimize the number of large rectangles for packing a set of small rectangles using best fit algorithm. The author was able to prove that this heuristic approach obtains better results and is faster compared to classical bin packing algorithm.

Three dimensional packing problem consists of packing a set of boxes into a minimum number of bins. To solve three dimensional bin packing problem, many methodologies using a hybrid approach were applied. (Lin, Foote, Pulat, Chang & Cheung, 1993) presented a layer by layer scheme that finds the appropriate boxes in the next layer using a hybrid genetic algorithm called SMILE to solve the three dimensional container packing problem. It is also a heuristic approach however the solution is augmented by simulated annealing to improve performance. The authors also presented an improvement of SMILE in the following year and proved that genetic algorithm is a good technique for optimization problems. (Yang & Shi, 2010) used a heuristic approach and introduced an algorithm for solving the three-dimensional bin packing problem, which is based on hybrid of caving degree algorithm from container loading problem and variable neighbourhood descent structure. Based on the computational experiments performed on standard benchmark problems, the algorithm show that the quality of the solutions is equal to or better than that obtained by the best existing algorithms in average. The authors applied the concept of genetic algorithm with multiple chromosomes to a three dimensional bin packing problem. From the results, the authors were able to prove that multiple chromosomes algorithm gives a better optimization solution. The authors were also able to show the multiple chromosomes algorithm created had better adaptability for large problem and near optimal solutions for small problems compared to a single chromosome algorithm. (Wang & Chen, 2010) likewise presented a hybrid genetic algorithm as well for a three dimension bin packing problem. The authors introduced in their hybrid algorithm a combination of a specially designed diploid representation scheme of individual and a heuristic packing method using fill packing method. With the above approach, the authors presented several genetic algorithms in their research and also found that the proposed hybrid algorithm presented using combination of chromosomes to be efficient in addressing three dimensional bin packing problem. Another example of hybrid algorithm for solving three dimensional bin packing problem was presented by (Jiang & Cao, 2012) with combination of simulated annealing. The authors combined the concept of block and batch to create a seven tuple algorithm and also increased the memory function for searching process. By doing so, the author's computational results were able to prove that the methodology used was very efficient to obtain near optimal solution within short duration.

(Pimpawat & Chaiyaratana, 2001) presented a heuristic rule which uses a co-operative co-evolutionary genetic algorithm (CCGA) in conjunction to solve three dimensional container loading or bin packing problem. The method differs from others by using proposed heuristics to partition the entire loading sequence into a number of shorter sequences. The authors proved that the methodology used is efficient in optimization of minimal number of containers required compared to standard genetic algorithm. The author was also proved that CCGA is suitable for use in a sequence based optimization problem use.

(Salma & Ahmed, 2011) considered a storage problem of a foam industry and introduced a heuristic by proposing an integer programming model for variable bin length storage problem. The problem is a variable sized bin packing where it involves allocating, without overlapping, a given set of rectangular items that cannot be rotated into the minimum number of three dimensional bins with different bin dimensions as input variables. Based on the proposed approach, the authors reduced the dimension of a given bin packing problem from three dimensional to a one dimensional.

On a more probabilistic and stochastic analysis note, (Akeda & Hori, 1976) performed Monte Carlo simulation and presented the confidence interval for mean random packing density and lower bound on limiting density comparison. (Ong, Magazine, & Wee, 1984) proved that the expected number of bins can be estimated as a function of number of elements and that the number of bins converges to expected value in probability. (Rhee & Talagrand, 1991) and (Rhee & Talagrand, 1993) dealt with stochastic packing with items of random sizes. In particular, the latter work showed that there exists an online algorithm that depends on the distribution of items. Other authors used different methods to solve the one dimensional bin packing problem such as genetic algorithm (Gómez & Fuente, 2000) use a cyclic crossover GA with fitness by area and variable mutation to minimize wastage of raw material. (Brusco, Thompson, & Jacobs, 1997) used simulated annealing with morphing process such that workload across all bins are evenly distributed. (Levine & Ducatelle, 2004) used a hybrid ant colony optimization (ACO) with local search whereas (Healy & Moll, 1996) used local optimization with rectangular layout in terms of holes and rectangles. (Van De Vel & Shijie, 1991) presented an algorithm which is non-polynomial as an application of bin packing technique to minimize makespan of a job scheduling problem. (Lins, Lins, & Morabito, 2003) considered a non-orthogonal 2D problem that seeks to maximize the number of items using the recursive partition of a rectangular or an L-shaped piece into two pieces, each of which is rectangular or an L-shaped piece. It is ideal for pallet loading and the L-approach always finds optimum packing of (ℓ, w)-

rectangles into rectangular piece even though it is a little time/memory consuming.

Related to the use of column generation to solve the 2D packing problem, (George, 1996) packed circles into rectangles using three approaches – a greedy heuristic, a pre-allocation method and integer programming related methods for no more than three pipe sizes in each container. (Puchinger & Raidl, 2007) developed an integer linear programming models for a 3-stage 2BP and used column generation in combination with greedy heuristics to improve the optimization process. (Vanderbeck, 1999), (Vanderbeck, 2000) and (Vanderbeck, 2001) did a computational study of a column generation algorithm for bin packing and cutting stock problems.

(Adelson, Norman, & Laporte, 1976) provided references on dynamic programming method used to solve the crate length optimization model. Other references include (Ji & Jeng, 1990), (Liu & Hsiao, 1997), (Mrad, Meftahi, & Haouari, 2013), (Savelsbergh, 1997), (Verma & Singh, 2010) and (Dowsland, 1996) which provided references on GA algorithm.

3 Crate Length Optimization

Orders come in various combinations of rolls from customers all over the world and each roll will be packaged into a crate. Due to process restraints, the crate width is a constant for all crates. With the crate width as a given constant, the roll lengths are calculated and adjusted according to the thickness of the material so as to have a consistent roll diameter. As such, the primary concern in the determination of the crate sizes is assumed to be the crate lengths. Since it is not possible to have a single crate type for every single roll size, it is inevitable that there will be some loss in the space inside the crates. As there can only be a few limited types of pre-determined crate sizes, the demand rolls will naturally be categorized into a few subsets of lengths which are packed accordingly into the best fit pre-determined crate length. Currently, Company S pre-determines the standard crate lengths from experience and there are four types of crate lengths in use. The company would like to determine the crate sizes given a fixed number of crate types to minimize overall loss and improve the efficiency of the transportation process.

3.1 Crate Length Optimization without Inventory Consideration

This section introduces a mixed integer linear programming model which is developed to solve the real world problem of finding the optimal crate lengths as described above. The model will find the optimal crate lengths with the objective of minimizing the total waste of space in the crates for a given number of crate types and demand distribution of the rolls of films.

3.1.1 Modelling Assumptions

The assumptions for crate length optimization model are as below:

(1) Each roll is assigned to one crate. This is a restriction due to the nature of the product. It is not possible to pack more than one roll in each crate as the rolls will be damaged from abrasion with one another during transportation.

(2) Demands of roll widths are given. The demands are generated based on historical data.

(3) The number of crate types is given as pre-determined input. The company would like to revisit the current practice of crate sizes and examine the consequences of having other number of crate types.

(4) The roll as placed into the rectangular crate will mean that the roll's width actually corresponds to the length of the crate whereas the roll's length is rolled up and contributes to the diameter of the roll.

The following parameters and decision variables are used for the crate length optimization model in this section:

Parameters

w_i	Roll width *i*
μ_i	Mean demand of roll width *i*
K	Number of crate types
N	Number of roll widths
L_{min}	Minimum crate length

L_{max}	Maximum crate length
P	Padding requirement inside the crates
M	A very large integer number

Decision Variables

L_k	Crate length *k*
x_{ik}	1, if roll *i* is assigned to crate length *k* and 0, otherwise
y_{ik}	Loss of length inside crate when roll *i* is assigned to crate length *k*, and 0, otherwise

3.1.2 Problem Formulation

The optimization model for Problem 1 has been formulated as follows:

$$Min \sum_{i=1}^{N} \sum_{k=1}^{K} \mu_i y_{ik} \tag{3.1}$$

s.t.

$$L_k - w_i \le y_{ik} + M(1 - x_{ik}) \text{ for } i=1,.., N,\ k=1,.., K \tag{3.2}$$

$$L_k + M(1 - x_{ik}) \ge w_i + P \text{ for } i=1,.., N,\ k=1,.., K \tag{3.3}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \text{ for } i=1,.., N \tag{3.4}$$

$$L_{\min} \le L_k \le L_{\max} \text{ for } k=1,.., K \tag{3.5}$$

$$y_{ik} \ge 0 \text{ for } i=1,.., N,\ k=1,.., K \tag{3.6}$$

$$x_{ik} \in \{0,1\} \text{ for } i=1,.., N,\ k=1,.., K \tag{3.7}$$

In the objective function (3.1), we minimize the total sum of space wastage in terms of length. The decision variable y_{ik} represents the extra length from the assigned crate length L_k minus the roll width w_i. This is multiplied by the corresponding mean demand of roll width μ_i to obtain the total sum of length loss inside all the crates assigned to all the rolls. Note that we have included P in the computation of the total loss. However this will not affect the optimal solution since P is a constant value, and hence will not affect the decision variables. Constraint (3.2) enforces the constraint that each roll must be assigned to a crate length that is bigger or equal to its width when x_{ik} is 1. Constraint (3.3) implies that the assigned crate length should have a minimum allowance of P inside the crates for each roll. Constraint (3.4) ensures that each roll is assigned to one crate type only. Constraint (3.5) states that all the decision variables of optimal crate lengths must be within the range of specified minimum crate length L_{min} and maximum crate length L_{max}. Constraint (3.6) dictates that the decision variables y_{ik} must be positive and lastly, constraint (3.7) states that the decision variables x_{ik} are 0-1 binary variables.

3.1.3 Computational Results

Figure 3.1 shows the input parameter for the roll width of Company S's demand distribution and we observe that it is highly scattered with a few obvious peaks. There are 80 types of roll widths in total. The few peaks are due to strongly dominant industry sizes, for example those for architectural and automotive use. The other input parameters are set such that the minimum padding length, P is 8cm and K varies from 2 to 10 types.

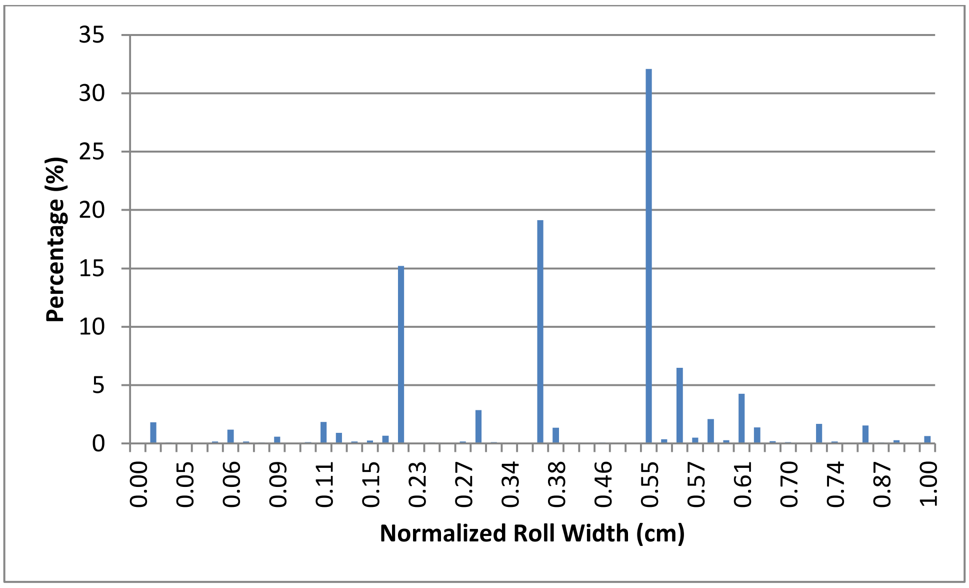

Figure 3.1 Roll Width Demand Distribution

Figure 3.2 shows the computational results obtained by implementing the mixed integer linear programming model for the crate length optimization problem using ILOG CPLEX11.2 for a distribution of Company S's demand of film rolls. We observe that as expected, the objective value will decrease with increasing specified number of crate types. There are diminishing returns of reducing loss of empty space inside the crates with increasing number of crate types.

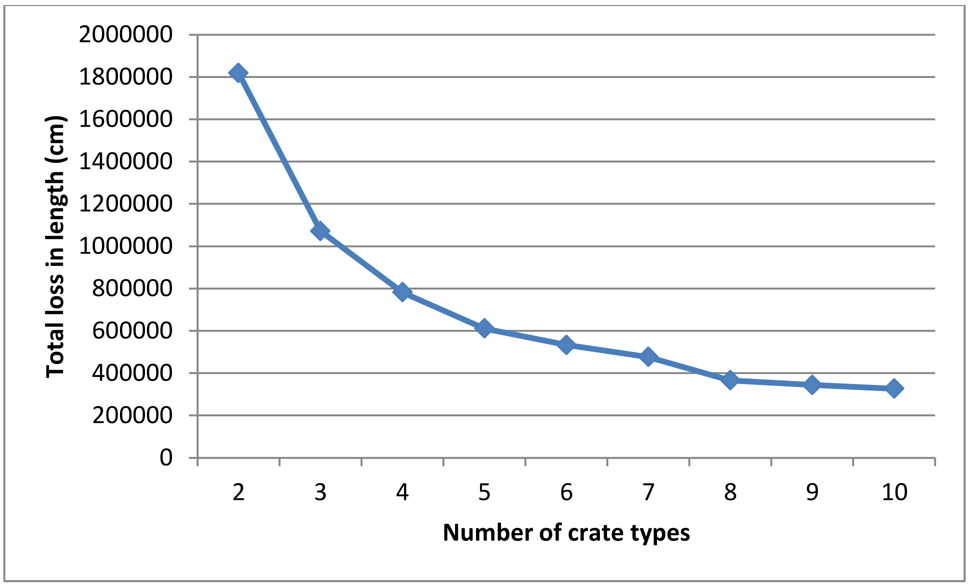

Figure 3.2 Objective Value with Number of Crate Types

Figure 3.3 shows the resulting optimal crate lengths for specified number of crate types from 2 to 10. The x-axis is the index *k* for the optimal lengths while y-axis is the optimal lengths L_k for the ten cases. For example, the case for two types has a line with two points at (1.00, 0.56) and (2.00, 1.00) which mean that the two optimal lengths in sequence are 0.56 and 1.00. For each result, the crate length 1.00 is a must because it has to accommodate for the biggest roll width in the demand with 8cm padding length. It can be seen that there are several crate lengths present in a few number of crate types hovering around 0.20, 0.38 and 0.56 due to the peaks of these demands in the input distribution (Figure 3.1). The first crate length for all cases generally is either 0.20 or 0.07, with the exception of the case for two types whereby it is 0.56.

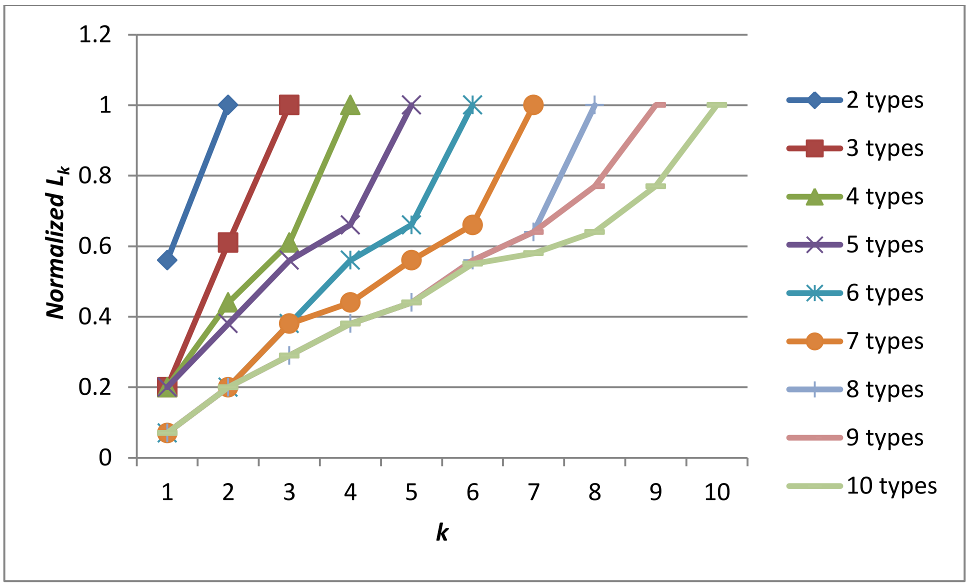

Figure 3.3 Optimal Crate Lengths for Given Number of Crate Types

3.2 Crate Length Optimization with Inventory Cost Consideration

The model presented in section 3.1 solves the real industrial problem by Company S to revise the optimal crate lengths used in transportation of the rolls. However, it did not consider demand uncertainty. The fluctuation in demand affects the amount of total inventory costs of the packaging crate materials. For each crate type, it is necessary to keep a certain level of safety stocks in each distribution centre to deal with the uncertainty of demands from customers. If the number of fixed crate types is too large, then it is unavoidable that more safety stocks need to be held and this adds to the complexity of handling. In addition, there is limited space in the warehouse to hold all types of crates. The motivation of the problem can also be more simply described as follows: if crate types increase, wastage will decrease due to better fit of the rolls in the crates but inventory cost will also increase due to more safety stocks, and so more crate types also translates to higher cost and complexity of handling.

Therefore the model in section 3.1 is extended with the consideration to find the optimal number of crate types so as to save on total inventory costs associated with the handling, storing and warehousing of all crate types. With the purpose of determining both the optimal crate types and lengths, the trade-off between loss of space inside the crates and inventory holding cost of the safety stock of crate types will be presented as an objective function in this section. In this extended problem, there is incentive to limit the number of crate types because the resulting objective value will decrease with increasing crate types, unlike the previous problem definition where inventory cost was not considered and there was no trade-off between the two variables. The two key decisions are choosing the optimal number of crate types and their lengths such that the associated inventory cost of having more number of box types' safety stock is balanced with minimizing the waste of space in the crates for all items. This is because if more crate types are decided, there is certainty that the extra space inside the crates will be less. However, this also leads to more costs to keep safety stocks of each crate type. On the other hand, a decision to have less crate types will result in more extra space but keep safety stock costs lower. In this section, the model will find both the optimal number of crate types and the optimal lengths to use.

3.2.1 Modelling Assumptions

The assumptions for the second problem are outlined below:

(1) Each roll is assigned to one crate. This is a continuation of the assumption from the model in section 3.1.

(2) Demands of the roll width are independently distributed.

(3) Periodic review policy is assumed. Hence the safety stock is computed using this policy based on the demand variability over the lead time and the review period.

The parameters and decision variables from the crate length optimization problem in Section 3.1 are used along with the additional variables as follows, with the exception of K, number of crate types now not being an input but a decision variable along with optimal crate lengths L_k.

Inputs and Parameters

σ_i	Standard deviation of demand of roll width i
p	Penalty cost
h	Inventory holding cost

The parameters p and h are assumed to be constant, independent of the number of crate types chosen. The assumption is based on that both factors are estimated from the average of historical data. However, it is possible that the parameter p may change considering that p is the penalty which encapsulates both the cost of extra wood or materials, and additional padding per cm of extra length in the crate. In addition, due to bulk ordering, it might be more expensive if more crate types are ordered from the packaging supplier. This is because the supplier might charge more for customizing a large number of varied crate sizes in lower quantities.

Decision Variables

K	Number of crate types
L_k	Crate length k

3.2.2 Problem Formulation

The second problem definition with inventory cost considerations can be formulated mathematically as below in conjunction with constraints (3.2) to (3.7) from the problem definition in Section 3.1.

$$Min \sum_{k=1}^{K} hS_k + \sum_{i=1}^{N} \sum_{k=1}^{K} p\mu_i y_{ik} \tag{3.8}$$

s.t.

$$S_k = \sqrt{\sum_i \sigma_i^{\ 2} x_{ik}} \text{ for } k=1,.., K \tag{3.9}$$

$$x_{i-1,k} \geq x_{ik} \text{ for } i=2,.., N, k=1,.., K \tag{3.10}$$

and constraints (3.2) to (3.7) as in the first problem in Section 3.1.

The objective function (3.8) reflects the dynamics of trade-off between having less crate types to have smaller value of the first term to compensate for the rise of value in the second term. The first term in the objective function is the product of the sum of S_k as the safety stock of each crate type k with the corresponding factor h to convert to equivalent in dollars of the inventory costs. The second term is the product of the sum of total space wastage in terms of length for all demands with a penalty cost factor p to convert to equivalent loss in dollars. Constraint (3.9) introduces safety stock of crate size *k*, S_k^2 which is the sum of σ_i^2 if w_i is assigned to crate type *k*. It can also be viewed as the risk pooling term for all rolls which are clustered into groups for optimal crate types. Lastly, constraint (3.10) forbids a roll *i*-1 from being assigned to a crate *k* unless its adjacent (and larger) roll *i* is assigned to crate *k* for all rolls *i*. Because of the different variability of demand, a situation may arise where it will be more desirable to assign a particular roll width to a larger crate type available. In order to restrict this situation, the constraint is introduced. The other constraints (3.2) to (3.7) are as described in Section 3.1. The formulation of the model is a non-linear mixed integer programming problem. As it is not a straightforward problem to solve, this leads to the use of a dynamic programming approach in Section 3.3.

3.3 Dynamic Programming Approach

Although the crate length optimization model with inventory consideration has been formulated in the preceding section, the integer programming model does not exploit the special sequential structure of the problem. As such, a dynamic programming approach is presented as an alternative to solve the problem.

The dynamic programming approach is based on the development of a recursive optimization process. There are several characteristics of a dynamic program: stages, states and the recursive property. The state of a stage reflects enough information to evaluate the optimal value function of a stage. The recursive property links the current stage to the next stage for all stages.

There are several structures of the problem that make the dynamic programming approach viable. The first is that the problem has optimal substructures which satisfy Bellman's Principle of Optimality wherein regardless of the initial state and decision, the subsequent decisions must constitute an optimal policy which is a consequence of the initial state and decision. Secondly, there is a finite choice of the crate length, which is one of the lengths of demand. In this problem, the solution space is most definitely limited to the set of the rolls widths considered. Therefore the solution space is discrete and finite. Seeing that the roll widths can be portrayed as an increasing array of variables, naturally the rolls will also be guaranteed an assignment of a crate type that is the smallest crate type that the rolls are able to fit in, or more simply put as the adjacent crate length. In addition, the objective function (3.9) is an additive function of the non-decreasing costs of space penalty and inventory holding of the safety stock. The decision at each stage depends on minimizing this total cost function which reflects the returns of the current path.

The following notation is used to formulate the problem as a dynamic program.

Inputs and Parameters

$w(i)$	Roll width i
$s_{a,\ldots,b}$	Pooled risk of standard deviation (safety stock) of demand of roll widths a to b
p	Penalty cost per unit loss of length
h	Inventory holding rate
n	Stage of the dynamic program

N Total number of stages (roll width sizes to be considered)

x_n State of stage n

Decision variable

a_n Decision variable at stage n where a_n represents the previous crate type width

3.3.1 Dynamic Programming Formulation

We formulate the problem as a forward induction dynamic program. In a forward induction process, the first stage is the initial stage of the problem. Then, the subproblems are solved moving forward one at a time until all stages are included. Suppose n is the stage of the dynamic program and N is the total number of roll width types considered, then there are N number of stages. Calculating $F(n)$ for n=1,…,N, where N is the total number of roll width types considered, we obtain the final optimal value $F(N)$ given that $F(0)$ is initialized to zero.

The problem is formulated as a forward induction dynamic program as shown below:

$$F_n(x_n) = \left\{ \begin{array}{l} F_{n-1}(x_n) = F(x_n), 0 \le x_n \le n-1 \\ \min\limits_{a_n} g_n(x_n, a_n) + F_{n-1}(x_n, a_n) = \min\limits_{a_n} g_n(x_n, a_n) + F(a_n), x_n = n \end{array} \right\} \quad (3.11)$$

$$F_0(x_0 = 0) = F(0) = 0 \quad (3.12)$$

$F_n(x_n)$ is the minimum cost function for state x_n at stage n, $F_{n-1}(x_n)$ is the cost to go and $g_n(x_n, a_n)$ is the one period cost. When $0 \le x_n \le n-1$, $F_n(x_n)$ is equal to $F_{n-1}(x_n)$. When x_n=n, then $a_n = \{0,1,...,n-1\}$ and $F_n(x_n)$ is the best decision

a_n which gives the minimum cost which is a sum of the one period cost at stage n and the cost to go, where the one period cost is evaluated as $p\sum_{i=a_n+1}^{n}[w(n)-w(i)]\mu_i + s_{a_n+1,\ldots,n}h$.

Using this method, it will divide the demands into groups through enumeration of all the stages and each group's set of rolls will be assigned to the adjacent crate length. The adjacent crate length is none other than the last or biggest roll width which is assigned to the cluster, plus an additional given pre-set of padding allowance, a constant *P*. The term $F(a_n)$ in (3.11) is the minimum cost up to stage a_n and one period cost refers to the additional possible costs of the possible states x_n, where we can see that it is the possible grouping of the roll widths into clusters resulting in cost of penalty from loss of length inside the crates, compounded by the penalty cost factor *p*, and also the cost of safety stocks in the latter part, compounded by the holding cost factor *h*. The difference between the two problem definitions in Sections 3.1 and 3.2 as described above therein lies in the consideration of the cost function to be minimized. If the inventory cost is removed, the dynamic programming approach reduces to minimizing only the penalty cost and the solution will be a crate type for every roll as there is no motivation to risk pool the fluctuations of the roll demands to drive down the inventory cost of holding safety stock of crate types. The formulation leads to several deductions of the defined problem, first of which is that the optimal solution set is equal to or is a subset of the set of roll widths considered. Following which, it can be deduced that the optimal decision assigns the roll widths to their adjacent length that is longer. Specifically this refers to the term $p\sum_{i=a_n+1}^{n}[w(n)-w(i)]\mu_i$ in the optimal decision whereby the *w*(n)* chosen is the largest of all the *w(i)* for all states of possible widths within the cluster starting for the index of i from a_n+1 up to n as the cost function is an additive function of two non-separable non-decreasing costs.

3.3.2 Computational Results

In this section, the proposed method of dynamic programming is applied to a set of demands using MATLAB version R2012a. The dynamic programming implementation has a time complexity with two 'for' loops that executes n times, where n is the input size of the number of crate widths. Therefore the order of complexity of the algorithm is quadratic complexity, $O(n^2)$. The complexity is of a polynomial algorithm.

Firstly, it is applied to Company S's actual demand data whereby the input parameters of p and h are 0.066 and 2.50 respectively. Both values are estimated from historical data. The result obtained is 10 optimal types of crates.

Next, the method is tested against a range of p and h values. Figure 3.4 shows the number of optimal crate types at varying values of h from 1.0 to 3.5 when p is fixed at 0.066. The number of optimal crate types decreases with increasing h. The decreasing pattern is expected because with higher inventory cost, there is more incentive to group into fewer types so that each type has the advantage to buffer the uncertainty in demand.

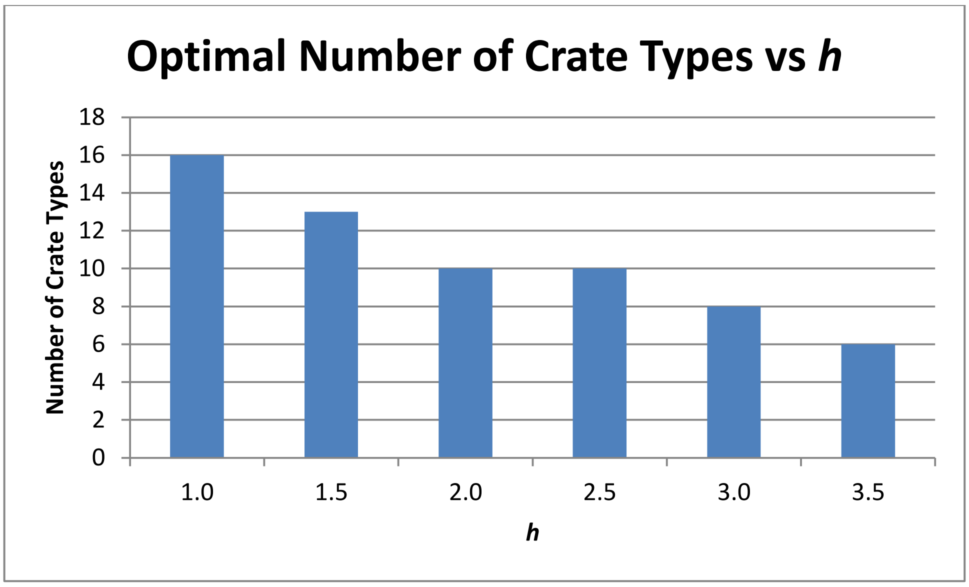

Figure 3.4 Optimal Number of Crate Types at Varying Values of *h*

On the other hand, Figure 3.5 shows the number of optimal crate types at varying values of *p* from 0.02 to 0.10 when *h* is fixed at 2.50. The number of optimal crate types increases with increasing *p*. The number of crate types is very sensitive to the changes in the values of *p*. The sensitivity is due to the fact that the factor *p*, although small in magnitude relative to *h*, is multiplied with the μ_i, mean demand of roll width *i* in the first term in (3.12). Its contribution to the total cost is compounded and is therefore more sensitive compared to *h*.

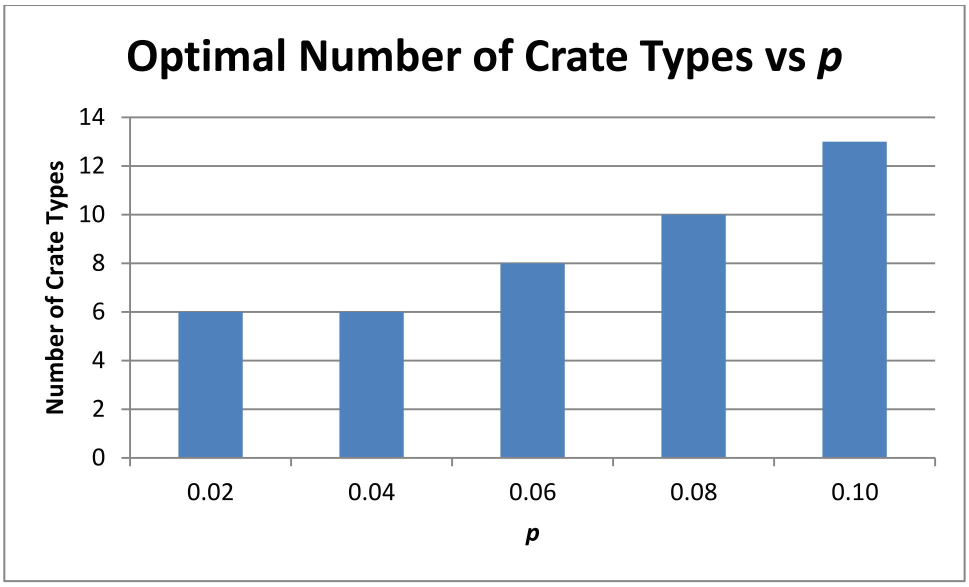

Figure 3.5 Optimal Number of Crate Types at Varying Values of *p*

Thirdly, the method is tested against different demand patterns of the roll width to see the effects of varying the ratio of *p/h* on the total cost and number of optimal types. Three types of demand pattern for a range of 20 types of roll width sizes are created. Figures 3.6 to 3.8 depict the demand patterns generated for testing. They are a) uniform, b) normal and c) right skewed pattern. For these three figures, the *x*-axis represents the roll width size from 100 to 290 while the *y*-axis represents the mean demand value. The normal demand refers to a generated demand in which the average demands for the mid-size crate types are generally higher than the smaller and larger crate types. Meanwhile, right skewed demand refers to a generated demand in which the average demands for the small-size crate types are generally higher than the other crate types.

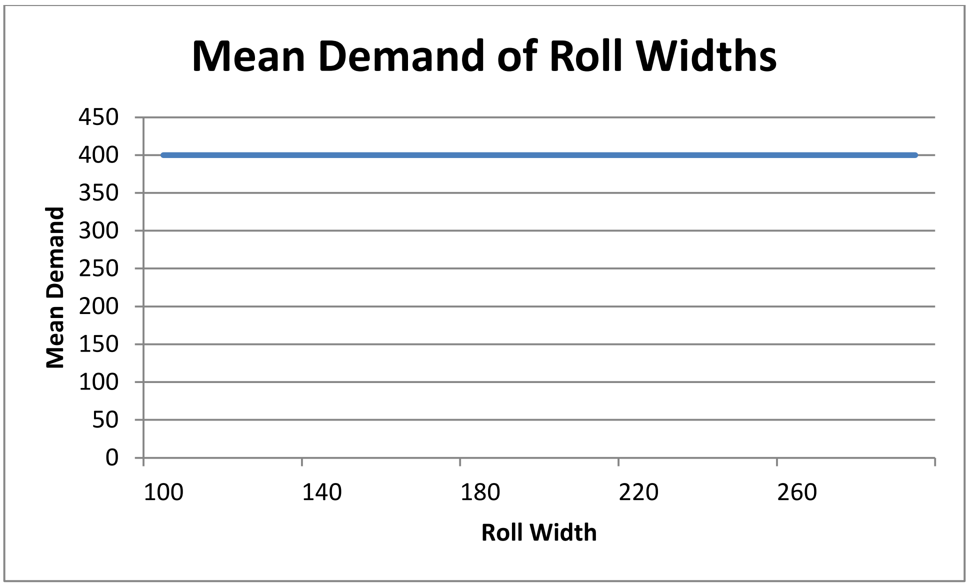

Figure 3.6 Uniform Pattern of Mean Demand of Roll Widths

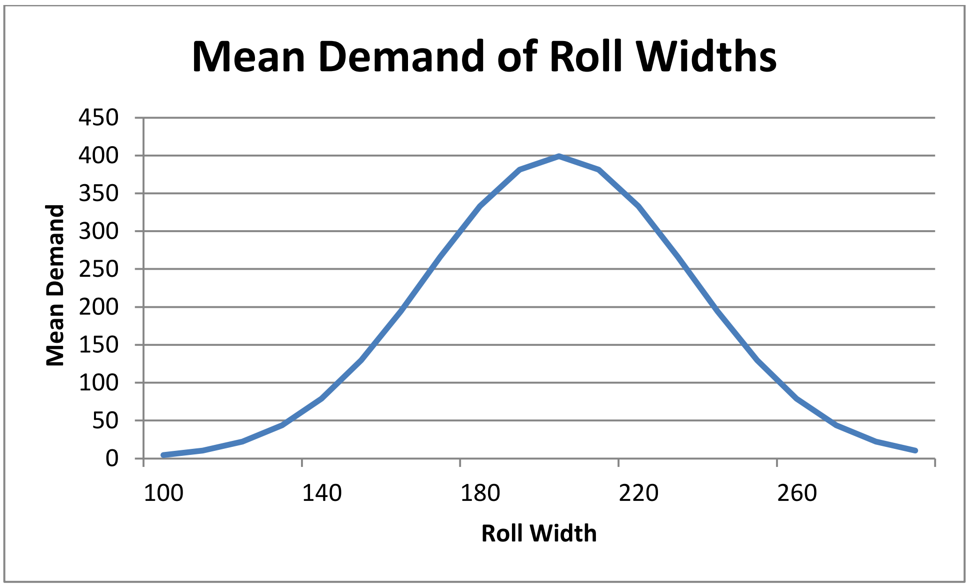

Figure 3.7 Normal Pattern of Mean Demand of Roll Widths

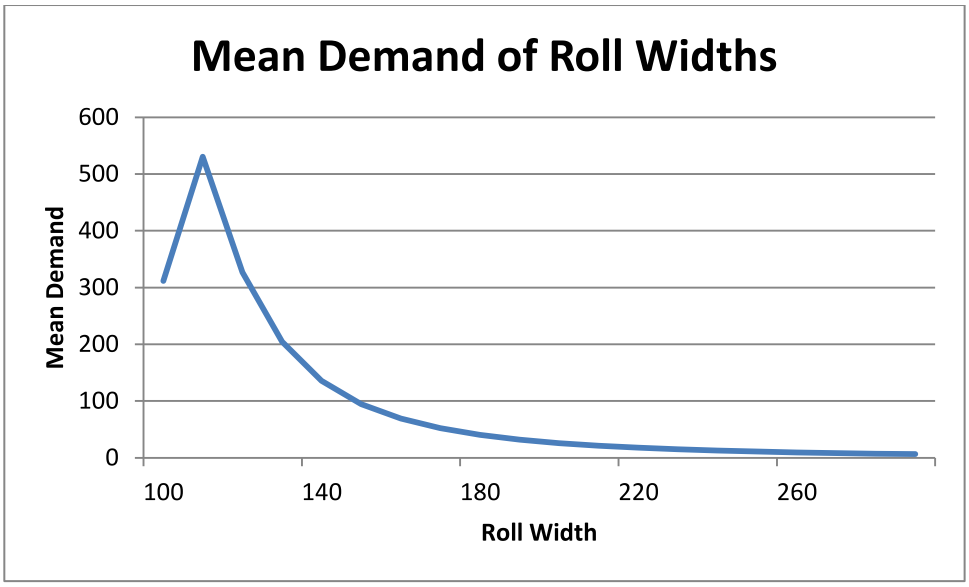

Figure 3.8 Right Skewed Pattern of Mean Demand of Roll Widths

In Figures 3.9, 3.11 and 3.13, the results are shown below whereby the *x*-axis shows the variance of demand varying from 0 to 4 for all cases of *p/h* from 0.005 to 0.05 and the *y*-axis shows the total cost in dollars for each scenario. In Figures 3.10, 3.12 and 3.14, the *y*-axis shows the number of optimal types.

For the case of uniform mean demand of roll widths, Figures 3.9 and 3.10 illustrate the effects of variance level on both optimal cost and number of optimal crate types. As expected, when variance increases, for all different ratios of *p/h*, the total cost increases uniformly. As for the number of optimal types, for values of *p/h* that are more or equal to 0.005, the optimal is always 20 types, one type for each roll width. The ratio has to be low enough before it starts triggering a change in the number of optimal types. When *p/h*=0.001, the number decreases rapidly from 20 to 10 then 7. When *p/h*=0.0005, the number drops slowly from 10 to 4 types. Lastly, for *p/h*=0.0001, the number decreases gradually from 4 to 2 types.

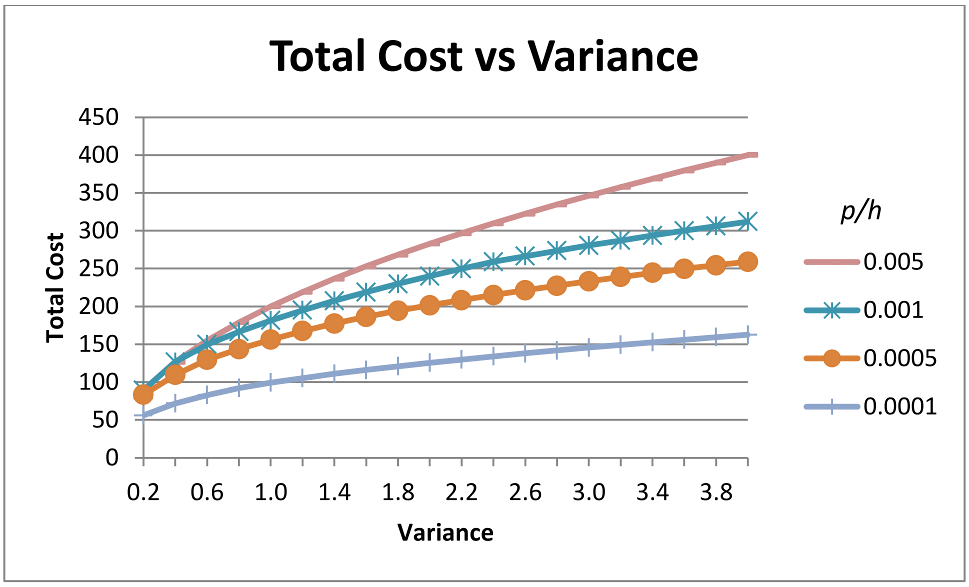

Figure 3.9 Total Cost vs Variance for a Uniform Pattern

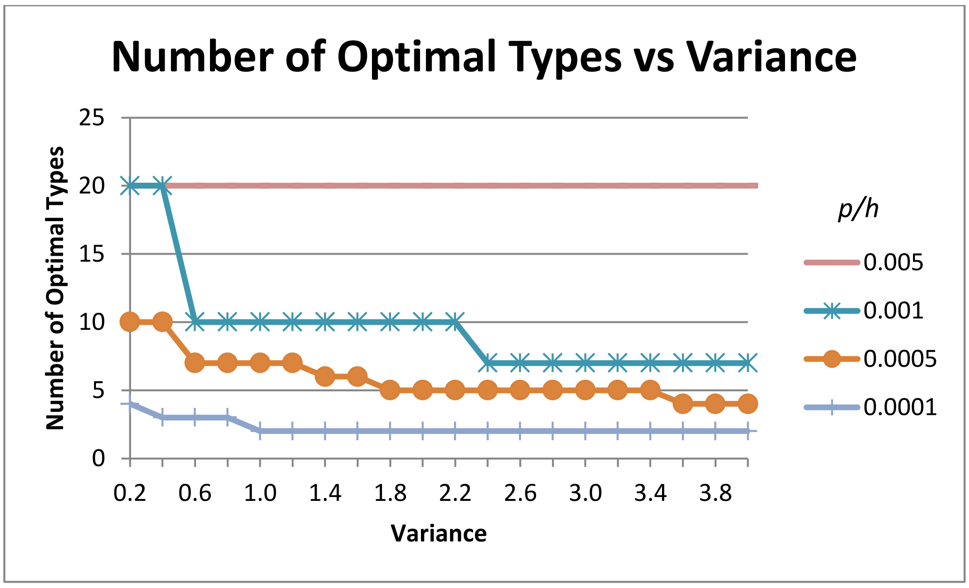

Figure 3.10 Number of Optimal Types vs Variance for a Uniform Pattern

On the other hand, for the case of normal mean pattern of roll widths, the increase in total cost appears to be more sensitive for lower ratios of *p/h*. It is also evident that at higher ratios of *p/h*, there appears to be more types of crates chosen as there is not much to gain from risk pooling the demands into clusters. At lower ratios of *p/h* of 0.01, 0.005 and 0.001, there are more distinct changes in the number of optimal types as the variance increases. The higher holding cost and variance levels increase the tendency to have less crate types as evidenced by Figure 3.12. This is because when the ratio of *p/h* is lower, the risk pooling effect has a more significant contribution to the total cost with the tendency to choose fewer types of crates resulting in higher savings from inventory cost.

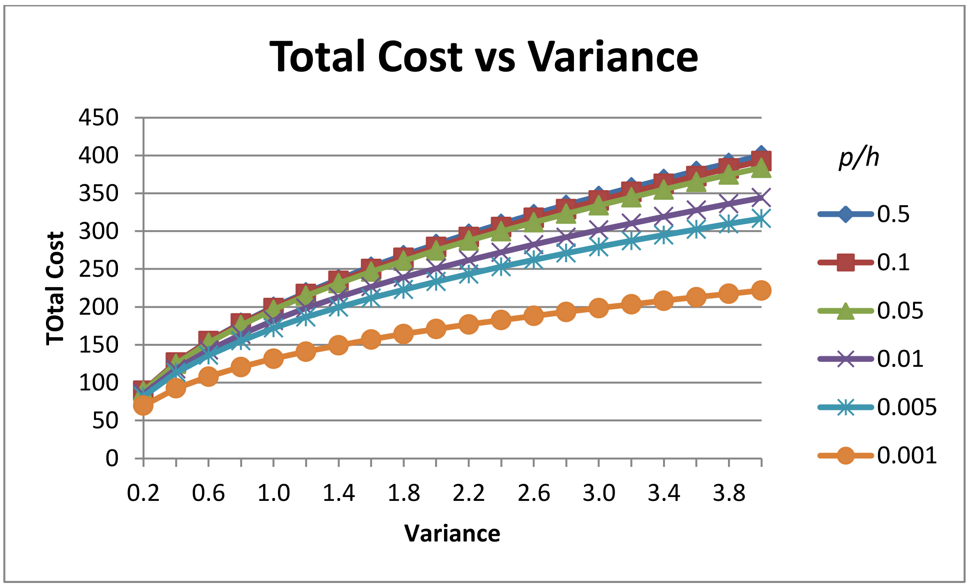

Figure 3.11 Total Cost vs Variance for a Normal Pattern

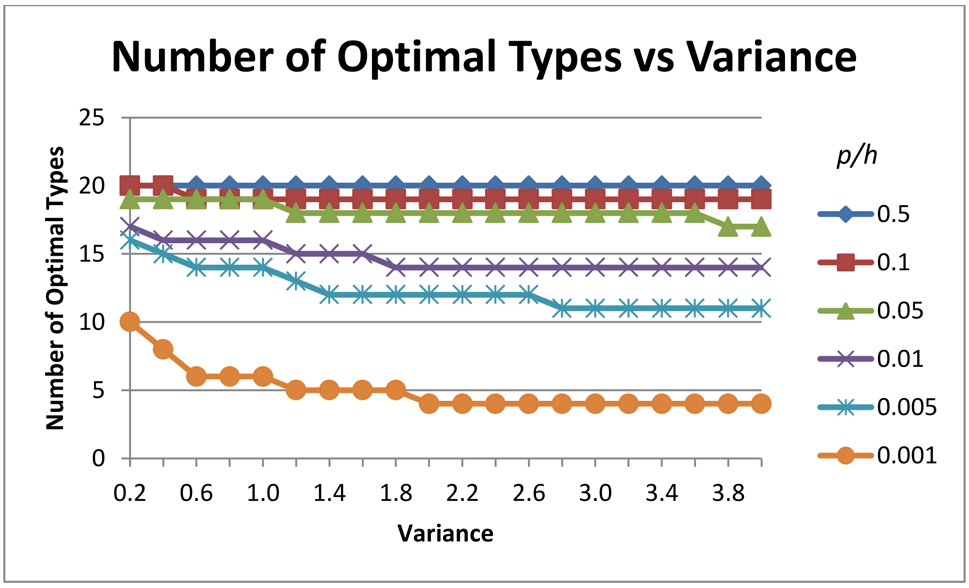

Figure 3.12 Number of Optimal Types vs Variance for a Normal Pattern

The same effect is investigated for a right skewed pattern of mean demand of roll width. Figure 3.13 highlights the same trend as before with total cost increasing with variance while Figure 3.14 shows the effect with increasing number of types.

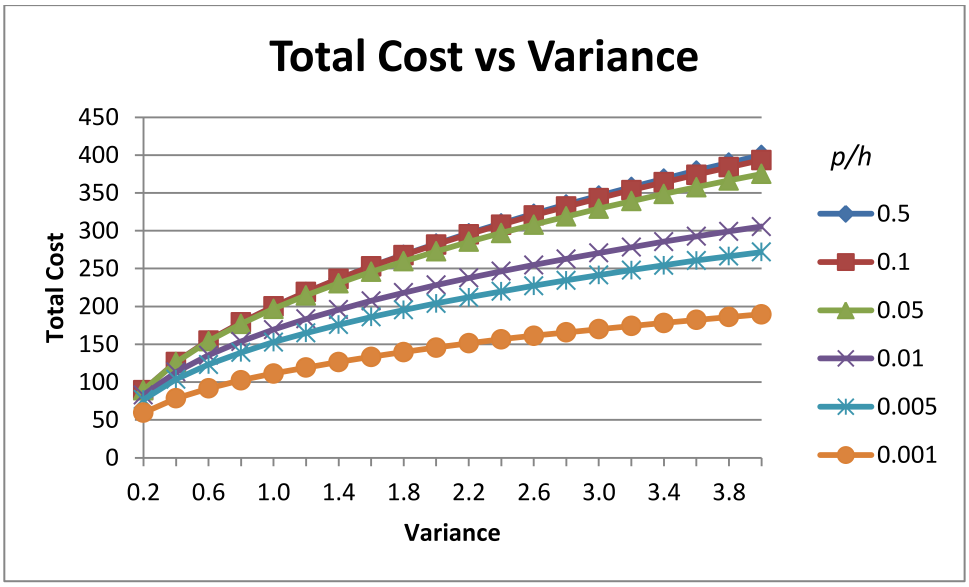

Figure 3.13 Total Cost vs Variance for a Right Skewed Pattern

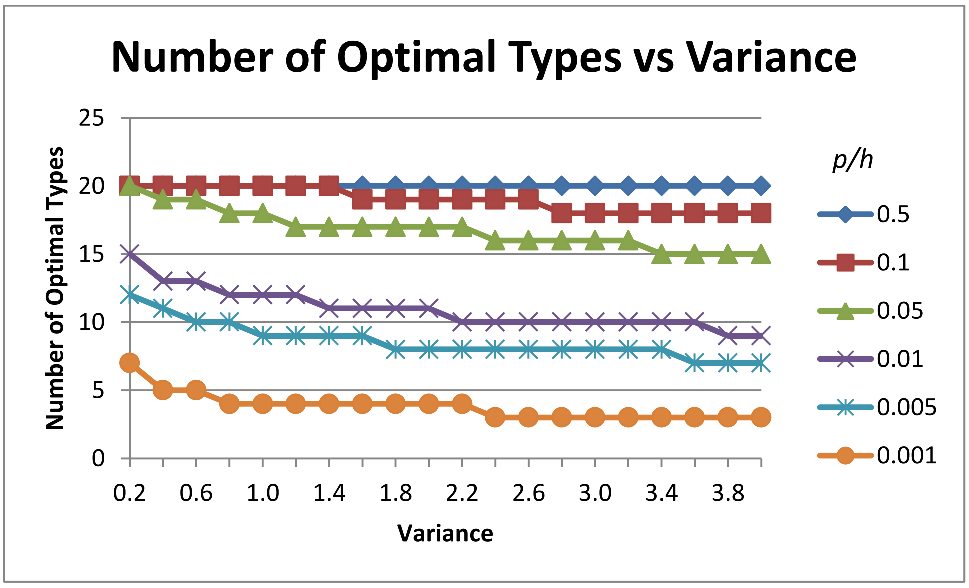

Figure 3.14 Number of Optimal Types vs Variance for a Right Skewed Pattern

Finally, Figures 3.15 and 3.16 are shown below whereby the *x*-axis shows the coefficient of variance of demand varying from 0 to 0.48 for all cases of *p/h* from 0.01 to 0.0005 and the *y*-axis shows the total cost in dollars for each scenario. As predicted, Figures 3.15 and 3.16 can be viewed that as CV (coefficient of variance) increases, for all different ratios of *p/h*, the total cost increases. The increase appears to be even more sensitive for lower ratios of *p/h* compared to increase in variance.

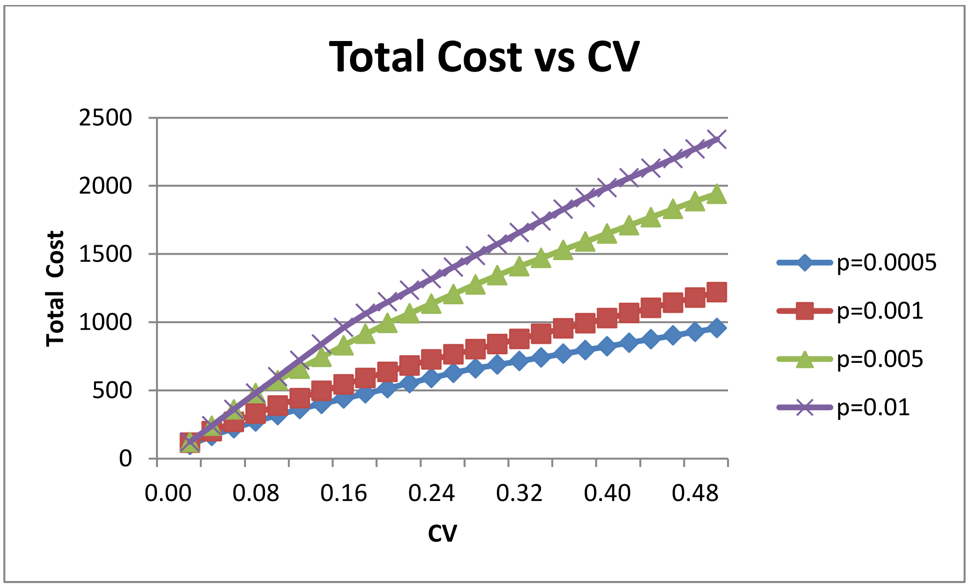

Figure 3.15 Total Cost at Different Levels of CV for a Uniform Demand

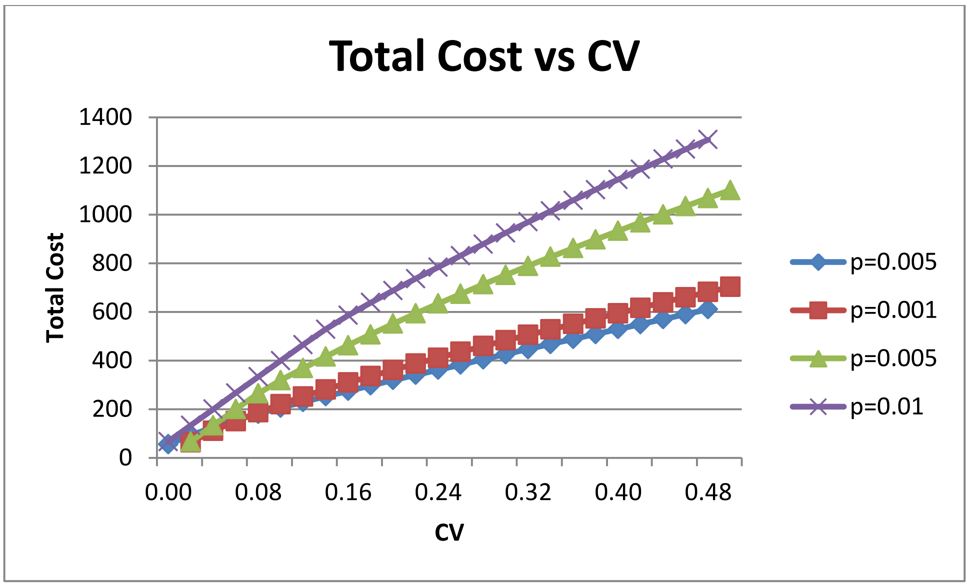

Figure 3.16 Total Cost at Different Levels of CV for a Normal Demand

Lastly, a demand distribution which is scattered and has several peaks which closely resembles the actual distribution of Company S as shown in Figure 3.1 is used to investigate the effect of varying the ratio of *p/h*. Figures 3.17 and 3.18 below show the results when a demand distribution with several peaks is used.

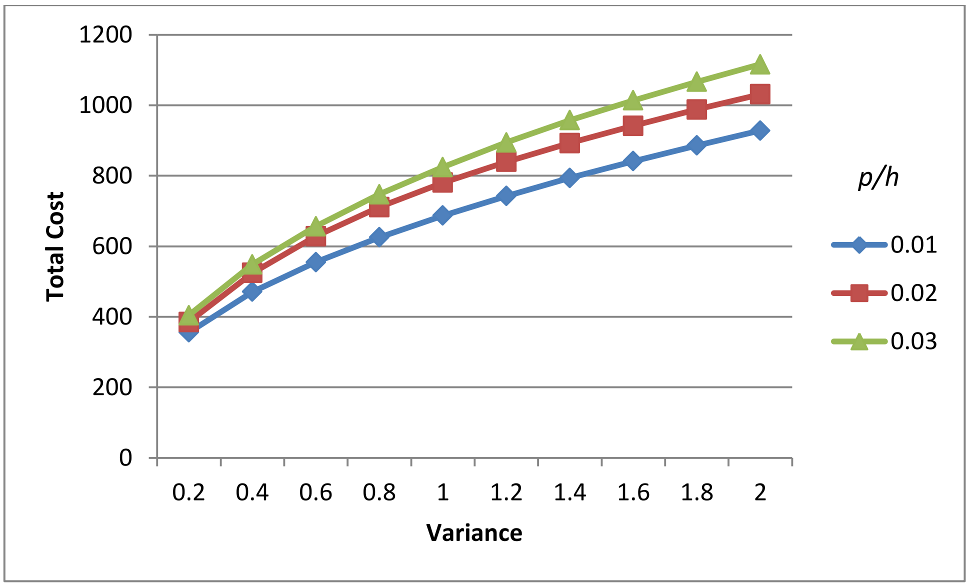

Figure 3.17 Total Cost at Different Levels of CV for a Demand Pattern Similar to Company S's Actual Demand

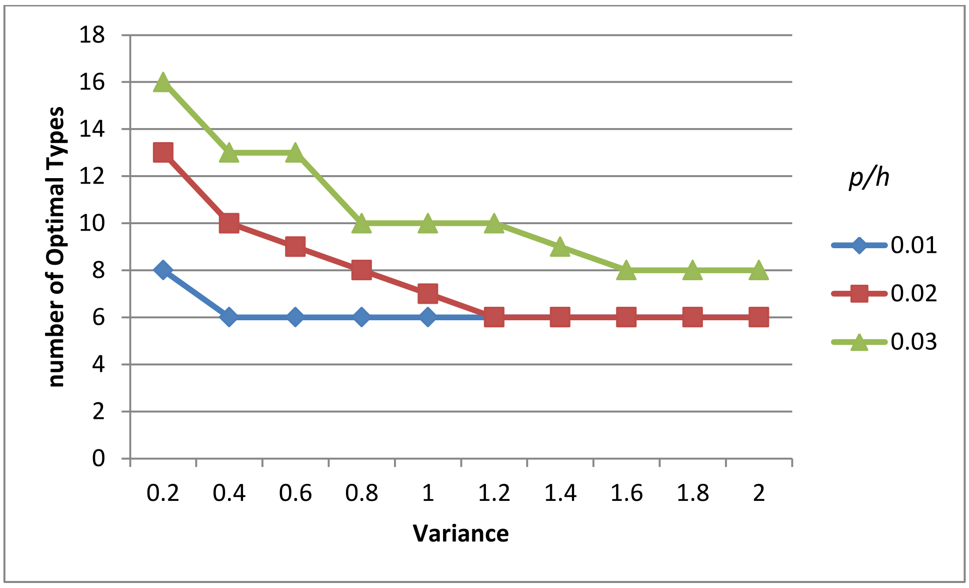

Figure 3.18 Total Cost at Different Levels of CV for a Demand Pattern Similar to Company S's Actual Demand

4 Generalized Crate Sizing Problem

In the previous section, the crate sizing problem deals with one crate dimension i.e. the crate length only. Realistically, a crate has three dimensions which can be taken into account for optimization. By including all three dimensions and thus calculating the total loss in terms of volume, this gives a more accurate grasp of the real problem.

Thus this section is dedicated to the generalized crate sizing problem. The problem can be viewed as an extension of the crate length optimization problem where now the optimization problem is extended to solve for both optimal crate length and crate width/crate height simultaneously. The crate width and height are treated as equal. Customers order a combination of roll sizes. A single customer demand has information on roll dimensions (roll length and diameter) and corresponding quantity (mean demand and variance). Each roll is to be packaged into one wooden rectangular crate. There are numerous possible types of rolls that can be customized by a customer. Because of the large number of combinations, it is not feasible to have one crate size for each roll type. The usual practice is to decide on a few crate sizes and stock them on hand to address the variety of the customer demand types. When packing a customer demand, each roll is packaged into a feasible crate size with minimum loss of volume. The total loss of volume of each customer demand is the total loss of volume inside the crates which is the sum of the differences between a chosen crate size and its roll for all rolls/crates. Because all demands have to be packed, this is equivalent to minimizing total volume of crates assigned. Meanwhile, the total inventory cost is the cost of safety stock from risk pooling all demands of the same-size crates for all crate types. The objective is two-fold, one is to decide the optimal crate sizes and the other is to assign and pack all rolls in a demand into crates with minimum total volume and minimum inventory cost.

However, the generalized crate sizing problem differs from crate length optimization problem in the earlier section in that the number of optimal crate sizes is fixed and given as input for optimization. The number of optimal crate types is a pre-determined input (but we can always vary the number of crate types to find the optimal number of crate types). In this problem, the inputs are the customer demand with various lengths and widths. Since there are only finite number of customer demand types (with various length and width dimensions), the choice of the optimal crate sizes will be finite and they should fall into the dimensions of the customer demand type. Without loss of generality, we assume that the requirement of padding has been considered in the dimension of the customer demand type.

As mentioned in the background problem, the crate has a unique characteristic of being very long and having a square cross sectional area to accommodate the cylindrical shape of the rolls. With a square cross section, this means that the crate width and height are of equal dimensions. Due to this property, solving the crate sizing problem can be viewed as solving a 2D problem. However, computational results will depict the total loss of volume (3D) for the generalized crate sizing problem.

Although dynamic programming was used in the earlier problem, in this extended problem, dynamic programming cannot be used as it does not exhibit the property of the Bellman's Principle of Optimality. The principle dictates that the optimal solution of the problem must constitute the optimal solution of earlier stages or smaller sub-problems. But in this case, the optimal solution of the sub-problem may change in the optimal solution when there are two dimensions to be considered instead of just one dimension in the section. As such, the recursive method cannot be applied here.

We will define the problem in sections 4.1-4.2. Three methods are proposed to solve this problem, namely enumeration method, marginal improvement method and genetic algorithm. They will be discussed in Sections 4.3-4.5.

4.1 Modelling Assumptions

The following assumptions will be used in this problem:

(1) The roll height or diameter is assumed to be always smaller than the roll width. As such, rotation is not possible. The crate length and crate width are not interchangeable.

(2) From this section onwards, the crate length L_k and crate width W_k are assumed to be inclusive of minimum padding P.

The following parameters are used for the generalized crate sizing problem in this section:

Parameters

w_i	Roll width *i*
d_i	Roll height (diameter) *i*
μ_i	Mean demand of roll type *i*
σ_i	Standard deviation of demand of roll type *i*
K	Number of crate types
L_{min}	Minimum crate length
L_{max}	Maximum crate length
W_{min}	Minimum crate width
W_{max}	Maximum crate width

Decision variables

L_k	Crate length *k*
W_k	Crate width (or height) *k*

z_{ik} 1, if roll *i* is assigned to crate *k* of length L_k and width W_k; 0 otherwise

The inputs required are demand of roll types *i* and the number of optimal types *N*. Each roll type has μ_i mean demand and σ_i standard deviation of demand. If roll type *i* is selected to the current group of pooled risk, $\sqrt{\sum_i \sigma_i^2 z_{ik}}$ represents the pooled risk of the variances of the rolls in the group. Specifically, it is the square root sum of the variances of the rolls in the group. Finally, the total cost or objective value in this problem is the total loss of volume and the total cost of inventory. It is the combined cost of crate volume from packing the rolls into crates and the cost of holding safety stock of pooled risks for all crate types *k*.

4.2 Problem Formulation

$$min \sum_{k=1}^{K} h \sqrt{\sum_i \sigma_i^2 z_{ik}} + \sum_{i=1}^{N} \sum_{k=1}^{K} p \mu_i L_k W_k^2 z_{ik} \tag{4.1}$$

s.t.

$$w_i z_{ik} + P \le L_k \text{ for } i=1,.., N, k=1,.., K \tag{4.2}$$

$$d_i z_{ik} + P \le W_k \text{ for } i=1,.., N, k=1,.., K \tag{4.3}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \text{ for } i=1,.., N \tag{4.4}$$

$$L_{\min} \le L_k \le L_{\max} \text{ for } k=1,.., K \tag{4.5}$$

$$W_{\min} \le W_k \le W_{\max} \text{ for } k=1,.., K \tag{4.6}$$

$$z_{ik} \in \{0,1\} \text{ for } i=1,.., N, k=1,.., K \tag{4.7}$$

Objective function (4.1) minimizes the sum of total inventory holding cost and sum of crate volume $\mu_i L_k W_k^2$ multiplied by the factor p for items i that are assigned to crate k. Constraints (4.2) and (4.3) dictate that the dimensions of the roll width w_i and roll diameter d_i are smaller than the crate lengths and crate widths for a feasible fit into the crates. Constraint (4.4) guarantees that each roll i is assigned to only one crate type of length L_k and width W_k. Constraints (4.5) and (4.6) confine the solutions of crate lengths and crate widths to their minimum and maximum values allowed. Lastly, constraint (4.7) prescribes that the variable z_{ik} is either 0 or 1.

To put into visual perspective, a table portrays all the options of crate sizes. Figure 4.1 shows the pictorial representation of all sizes and demand. The columns represent crate width whereas the rows represent crate length, both sorted in ascending order. Each cell represents a demand of crate size in the two dimensions. The crate length and crate width are discrete and do not need to be equally spaced sizes.

For a demand of *u* crate lengths and *v* crate widths, the table is an array of size *u*x*v*. As the optimal solution will only lie on the dimensions given by the customer orders, every cell in the table can be regarded as a potential candidate for the optimal size. Naturally, the cell with the biggest size $x_u y_v$ is part of the optimal size solution because all demands must be assigned. For any given number of optimal crate types *K*, there can be ${}^{uv-1}C_{(K-1)}$ ways of choosing the optimal solution. To give an insight into the magnitude of the problem, a problem with just 10 crate lengths and 10 crate widths with 4 optimal sizes has ${}^{99}C_3$ that is 156,849.

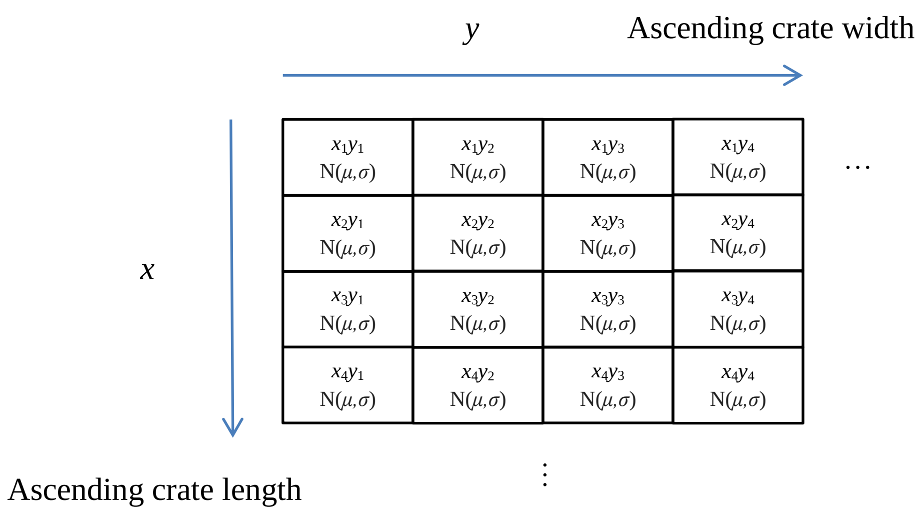
y
Ascending crate width
x
Ascending crate length
x_1y_1 N(μ,σ)
x_1y_2 N(μ,σ)
x_1y_3 N(μ,σ)
x_1y_4 N(μ,σ)
x_2y_1 N(μ,σ)
x_2y_2 N(μ,σ)
x_2y_3 N(μ,σ)
x_2y_4 N(μ,σ)
x_3y_1 N(μ,σ)
x_3y_2 N(μ,σ)
x_3y_3 N(μ,σ)
x_3y_4 N(μ,σ)
x_4y_1 N(μ,σ)
x_4y_2 N(μ,σ)
x_4y_3 N(μ,σ)
x_4y_4 N(μ,σ)
…
⋮

Figure 4.1 Pictorial representation of sizes and demand

4.3 Enumeration Method

The first approach of enumeration method is the most direct method. It enumerates all possible optimal sizes given a specified number of crate types and demand and then assigns all the demand accordingly to calculate the minimum total cost. Each demand is assigned to the smallest feasible crate type.

The method has been implemented using MATLAB R2012a program. It is simple in execution and works well for small size problems. However the brute force method slows down considerably when applied to larger problems. Even though the program can be extended to accommodate larger size problems, running time is a cause of concern. Even so, the enumeration method is useful to yield optimal solutions for small size problems and serves as an appropriate comparison basis and benchmark for other subsequent methods.

4.4 Marginal Improvement Method

Due to the limited capability of the enumeration method, a second method is introduced in this section. The second method involves using marginal improvement to find and improve a given solution.

Given a solution, i.e. a set of crate sizes, we would like to know the marginal improvement in the overall objectives values if we change one of the crate sizes to one of its neighbouring sizes while keeping the rest unchanged. This can be illustrated in Figure 4.2. Note that the highlighted cell is the crate size x_iy_j (the length is L_i, and the width is W_j) that we would like to change, and its neighbours are $x_{i-1}y_{j-1}$, $x_{i-1}y_j$, $x_{i-1}y_{j+1}$, x_iy_{j-1}, x_iy_{j+1}, $x_{i+1}y_{j-1}$, $x_{i+1}y_j$ and $x_{i+1}y_{j+1}$.

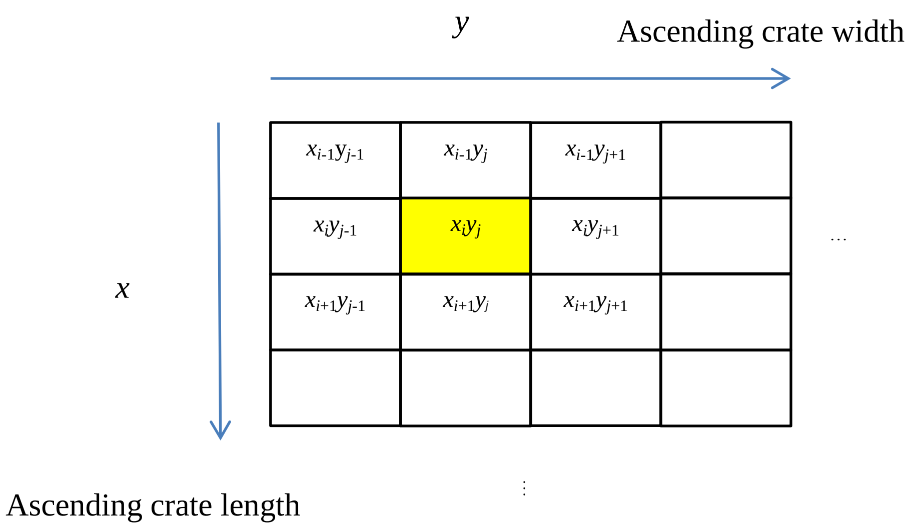

Figure 4.2 Neighbours for marginal improvement

Cx_iy_j is defined as the objective value for the overall solution, while Cx_iy_{j+1} is the objective value for the overall solution when crate size x_iy_j is changed to x_iy_{j+1} while the rest of crate sizes remained the same.

The table also displays the potential neighbours for marginal improvement for a point that has 8 adjacent neighbours. For a point that is on the perimeter, there will be fewer neighbouring points. Figure 4.3 shows the possible directions for marginal improvement for a cell that is not on the perimeter and has eight neighbours. The number of neighbours is equal to the number of possible directions.

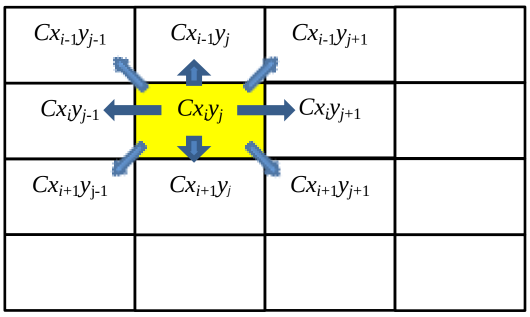

Figure 4.3 Directions for marginal improvement

The algorithm for the marginal improvement method is as follows:

1. Set StopFlag=0
2. For any crate size we would like to change, while StopFlag =0, do the following steps 3-8 ; else exit
3. Set current point PointofConsideration
4. Calculate and set total cost of current solution CurrentTotalCost
5. Calculate change in CurrentTotalCost when the PointofConsideration is changed to a neighbouring cell as neighbourCost for all neigbours
6. Find minimum neighbourCost and set LowestCost
7. If LowestCost=CurrentTotalCost then StopFlag=1 and current point is not improving the solution any more, exit; else replace current point to

PointofConsideration as the neighbour with lowest neighbourCost and repeat steps 3 to 7

8. Output LowestCost and new crate sizes

The marginal improvement method is based on one-crate-size-at-a-time. While keeping the rest of the solution, it only improves the current input size of consideration. The quality of the rest of the other sizes greatly affects the quality of the solution. If the other sizes are very far off, then the improved one size does not help in minimizing total cost and the solution will also be far from optimal. Nevertheless, the marginal improvement is a useful tool when one is in the vicinity of optimal or good solutions. On top of that, it can be used as a local search tool to generate better solutions in a short amount of time because the marginal improvement method at any one time only needs to calculate at most eight neighbouring cells to consider. There will be savings in time for a quick improved solution and computational power because it does not need to evaluate all the cells which can certainly slow down the process.

The marginal improvement algorithm improves based on changing one crate size at a time. In order to change all the crate sizes, there are two ways of choosing which crate size to improve. The former is by sequence (marginal improvement by sequence- MIBS) while the latter is by random (marginal improvement by random- MIBR). For example, for a three size problem, MIBS will change crate size type 1 and then crate size type 2. Crate size type 3 cannot be changed because it is the biggest crate size that must accommodate itself. Meanwhile, MIBR may either change crate size type 1 first and then crate size type 2, or crate size type 2 first and then crate size type 1. In addition, the starting values for the crate sizes can either be pre-set or randomly generated.

4.4.1 Numerical Experiments

Numerical experiments were conducted to evaluate the marginal improvement method. Without loss of generality, in all the experiments, we assume that the demand size is 1 and variance is 0. To compare to the enumeration method, the following tests were conducted. Table 4.1 shows the comparison between marginal improvement (MI) and enumeration method for the problem with two sizes. Note that in this problem, as we can only vary one crate size, MIBR reduces to MIBS. The starting value of (x_1^*, y_1^*) is (1,1) and the value of (x_2^*, y_2^*) is (n,n). The size of the search space is n^2.

The results indicate that the MI methods are able to find the global optimal solution.

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	**(Enumeration) Cost**	**(MI) Cost**
2	**2**	2	1	2	2	20	20
	3	3	2	3	3	153	153
	4	4	2	4	4	640	640
	5	5	3	5	5	1925	1925
	6	6	3	6	6	4860	4860
	7	7	4	7	7	10339	10339
	8	8	5	8	8	20288	20288
	9	9	5	9	9	36369	36369
	10	10	6	10	10	61600	61600
	11	11	6	11	11	99341	99341
	12	12	7	12	12	153072	153072
	13	13	7	13	13	229333	229333
	14	14	8	14	14	330848	330848
	15	15	9	15	15	467775	467775
	16	16	9	16	16	645376	645376
	17	17	10	17	17	873647	873647
	18	18	10	18	18	1163808	1163808
	19	19	11	19	19	1523059	1523059
	20	20	12	20	20	1971200	1971200

Table 4.1 Comparison between MI and enumeration method for two sizes

Table 4.2 shows the comparison between MIBS, MIBR and enumeration method for the problem with three sizes. The starting values of (x_1^*, y_1^*) and (x_2^*, y_2^*) are randomly generated and the value of (x_3^*, y_3^*) is *(n,n)*. Because the starting values are randomly generated, the results depend on the initial values. The experiment is run 20 times and the lowest cost is obtained with the corresponding optimal sizes. The results for both MIBS and MIBR indicate that the MI methods are able to find the global optimal solution.

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	**(Enumeration) Cost**	**(MIBS) Cost**	**(MIBR) Cost**
3	**3**	3	1	3	2	126	126	126
	4	2	4	4	2	512	512	512
	5	4	4	5	2	1587	1587	1587
	6	3	6	6	3	3888	3888	3888
	7	5	6	7	3	8482	8482	8482
	8	4	8	8	4	16384	16384	16384
	9	6	8	9	4	29709	29709	29709
	10	5	10	10	5	50000	50000	50000
	11	7	10	11	5	80886	80886	80886
	12	6	12	12	6	124416	124416	124416
	13	8	12	13	6	186271	186271	186271
	14	7	14	14	7	268912	268912	268912
	15	9	14	15	7	380682	380682	380682
	16	8	16	16	8	524288	524288	524288
	17	10	16	17	8	711417	711417	711417
	18	9	18	18	9	944784	944784	944784
	19	11	18	19	9	1240174	1240174	1240174
	20	10	20	20	10	1600000	1600000	1600000

Table 4.2 Comparison between MIBS, MIBR and enumeration method for three sizes

Table 4.3 shows the comparison between MIBS and enumeration method for the problem with four sizes. The starting values of (x_1*, y_1^*), (x_2^*, y_2^*) and (x_3^*, y_3^*) are randomly generated and the value of (x_4^*, y_4^*) is (*n*,*n*). The experiment is run 20 times and the lowest cost is obtained with the corresponding optimal sizes and cost as shown in MIBS I Cost column. With the exception of *n*=8, it is able to find the global optimal solution.

When we increase the number of runs from 20 to 100, the MIBS II Cost column is obtained.

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	x_3^*	y_3^*	**(Enumeration) Cost**	**(MIBS I) Cost**	**(MIBS II) Cost**
4	**4**	4	2	2	4	4	3	456	456	456
	5	5	2	2	5	5	4	1355	1355	1355
	6	6	2	2	6	6	4	3360	3360	3360
	7	7	3	3	7	7	5	7231	7231	7231
	8	8	4	3	8	8	6	14048	14400	14048
	9	9	3	3	9	9	6	25515	25515	25515
	10	10	4	4	10	10	7	42820	42820	42820
	11	11	4	4	11	11	8	68959	68959	68959
	12	12	5	5	12	12	9	106704	106704	106704
	13	13	5	5	13	13	9	-	158925	159757
	14	14	5	5	14	14	10	-	230384	230720
	15	15	6	5	15	15	11	-	324975	324675
	16	16	6	6	16	16	11	-	449056	449056
	17	17	6	17	11	8	17	-	613547	608022
	18	18	8	7	18	18	13	-	812718	812718
	19	19	7	19	12	10	19	-	1068560	1059079
	20	20	8	7	20	20	15	-	1369500	1372720

Table 4.3 Comparison between MIBS and enumeration method for four sizes

Table 4.4 shows the comparison between MIBR and enumeration method for four sizes. Note that the MIBR method is run for 20 times (MIBR I) and 100 times (MIBR II) and both runs reach global optimal solution for *n* ranging from 4 to 12.

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	x_3^*	y_3^*	**(Enumeration) Cost**	**(MIBR I) Cost**	**(MIBR II) Cost**
4	**4**	4	2	2	4	4	3	456	456	456
	5	5	2	2	5	5	4	1355	1355	1355
	6	6	2	2	6	6	4	3360	3360	3360
	7	7	3	3	7	7	5	7231	7231	7231
	8	8	3	3	8	8	6	14048	14048	14048
	9	9	4	3	9	9	7	25515	25515	25515
	10	10	4	4	10	10	7	42820	42820	42820
	11	11	4	4	11	11	8	68959	68959	68959
	12	12	5	4	12	12	9	106704	106704	106704
	13	13	5	5	13	13	9	-	158925	158925
	14	14	5	5	14	14	10	-	230384	230384
	15	15	6	6	15	15	11	-	324675	324675
	16	16	6	6	16	16	11	-	449056	449056
	17	17	7	6	17	17	12	-	608022	608022
	18	18	7	7	18	18	13	-	807300	807300
	19	19	8	7	19	19	14	-	1059079	1059079
	20	20	9	8	20	20	15	-	1377200	1377200

Table 4.4 Comparison between MIBR and enumeration method for four sizes

From the results for the marginal improvement methods, it can be seen that they can be used to find optimal solutions for small problems of two, three and four sizes. For the same-scale problem, the marginal improvement method was able to produce the optimal solution with a much shorter time compared to the enumeration method which took very long. The running time of the marginal improvement method took minutes whereas the enumeration method took hours. However, when the problem gets bigger, it becomes more difficult to get optimal solutions, and so we introduce another method in Section 4.5.

4.5 Genetic Algorithm Method

The genetic algorithm (GA) method is a well-known evolutionary algorithm that is used to handle a multitude of optimization problems. Based on the idea of ‘survival of the fittest’, GA begins with an initial population which comprises of randomly generated individuals. Every individual is evaluated and given a fitness score/measure. At each generation, the individuals undergo mutation, crossover and the fittest individuals are selected to remain and survive for the next generation.

The GA has many good properties that can be used to solve different types of problems. It has good mechanism to consider the trade-off between exploitation and exploration. By using the appropriate selection, crossover and mutation mechanism, we can achieve good results. In general, selection helps to keep elitism in the solutions, crossover performs exploitation and mutation does the exploration. In order for the GA to perform well, it is important to have the right solution representation (chromosome representation) which can work well with the crossover operation. If such a representation is not present, when we do crossover, we might not be able to exploit the neighbourhood to obtain good solution. In our problem, the chromosome is represented by the crate sizes which will be discussed later, and if we do naïve crossover, we might destroy the neighbourhood structure. Hence we propose a Hungarian method which aims to match the genes for crossover to ensure the offspring will lie within the neighbourhood of the parents.

The GA approach is introduced in this section because the enumeration method and marginal improvement methods are local search methods that are not as efficient for solving larger size problems. The GA approach is able to obtain improved results over the other two methods.

The proposed GA algorithm can be viewed as a general framework for a generalized crate sizing problem. It can be used to find the optimal 2D(+1) sizes of crates given a demand of crate sizes. The first requirement for this framework is the crate length and crate width is not interchangeable. The second requirement is that the second and third dimensions are equal and treated as the same. As such, this can be applied to any problem which has the same properties. In essence, the GA can be used for a problem that finds the optimal sizes of long and rectangular type of boxes/packaging/crates. The demand of the sizes can be normal or of other distributions. This does not affect the suitability of the GA algorithm. However, this changes the fitness function of the GA where the evaluation function is currently proposed for a normally distributed demand. For other distributions, the fitness function should be modified accordingly.

4.5.1 Chromosome Representation

Each chromosome is an individual and represents a solution. The chromosome has several genes and each pair of genes represents a crate size for a customer demand type. The chromosomes are of fixed length. Note that if there are K crate types, the chromosome only needs 2(K-1) genes. This is because the largest customer demand type is always a required crate size in the optimal solution. Hence it is not necessary to include it in the chromosome.

Figure 4.4 depicts an example of chromosome representation of crate sizes. This is a chromosome for six crate types. The first pair of genes (L_3,W_2) is a crate size for the customer demand type with crate length L_3 and crate width W_2. Each pair of genes refers to the crate sizes for the customer demand type respectively.

(L_3,W_2)
(L_3,W_5)
(L_{12},W_4)
(L_9,W_7)
(L_4,W_{12})

Figure 4.4 Chromosome representation

4.5.2 Creation of initial population

The creation of the initial population P_0 comprises of crate sizes chosen from the set of possible customer demand types. If the number of available customer demand types is N, and the number of crate sizes of the problem is K, where K is less than N, each chromosome is created by choosing K-1 individuals from N-1 types. For the initial population, 100 chromosomes are generated.

4.5.3 Selection Mechanism

After a population of individuals are generated, we need to have a selection mechanism to select parents for reproduction. For the criteria of selection, a fitness value *F(x)* has to be assigned to all individuals in the population. The total cost of an individual is the evaluation of the objective function as described in the problem formulation (4.1). The fitness of an individual *F(x)* is then measured by using a function to calculate total cost of the individual over the mean of the total cost of all the individuals in the population.

In this algorithm, tournament selection is used. The winning pair of individuals is selected as parents for mating.

4.5.4 Reproduction – crossover operation

After a pair of individuals is selected as parents, crossover is usually performed. In our problem, we will use the arithmetic crossover. Let β^{f_1} and β^{f_2} be the genes matched for crossover, and the offspring is β^{s_1} after crossover. It is defined as $\beta^{s_1} = \alpha.\beta^{f_1} + (1-\alpha).\beta^{f_2}$. After performing crossover, the child might not belong to any of the customer types, and so some repair needs to be carried out. For repair, the child is modified to the closest customer type from its neighbours.

Note that it is important to find the matching pair of genes to perform the crossover. If we naively match the genes by their order in the chromosome, the offspring generated might be far off from their parents, which will be undesirable for crossover operations. The concept of distance is introduced to measure the similarity between the two gene pairs. In our case, we use rectilinear distance as the measure.

Figure 4.5 illustrates a naïve crossover between parent A and parent B and the rectilinear distance between the gene pairs. Figure 4.6 shows the relationship of the crossover in a graph. It can be seen that the offspring resulting from the crossover can be far away from their parents which would destroy the neighbourhood structure.

Parent A	Parent B	Rectilinear Distance
(187, 92)	(166, 69)	44
(145, 85)	(173, 78)	35
(187, 99)	(147, 36)	103
(113, 71)	(174, 96)	86
(171, 28)	(120, 77)	100

Figure 4.5 A naïve crossover example

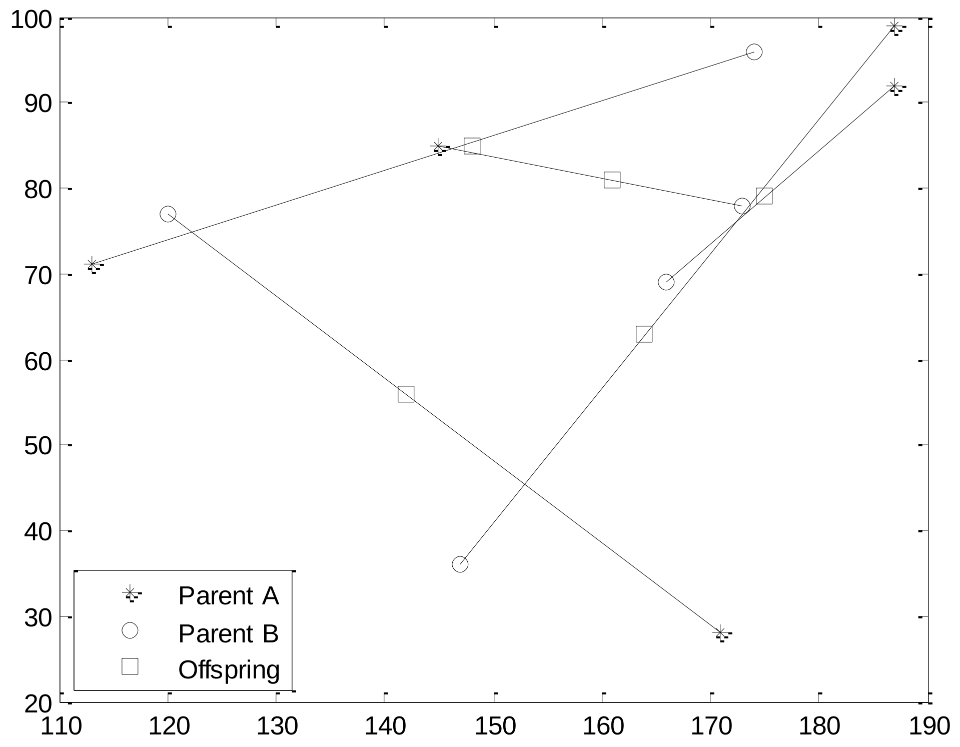

Figure 4.6 Naïve crossover example in a graph

In our problem, a naïve crossover is not very effective. This is because the optimal crate size is not likely to deviate far from its neighbours. Therefore the neighbourhood structure is essential here. When we preserve the neighbourhood structure, the crossover is done over a smaller region and produces an offspring in the vicinity. By taking advantage of this special property, it is able to have more exploitation. If we match the pairs of genes to the closest neighbour, this can be modelled as a 1-to-1 assignment problem. The objective of the assignment problem is to pair off all genes between the two parents at minimum matching cost. Therefore we propose the Hungarian algorithm to solve the assignment problem here.

Figure 4.7 illustrates a modified crossover between parent A and parent B and the rectilinear distance between the gene pairs. Figure 4.8 shows the relationship of the crossover in a graph.

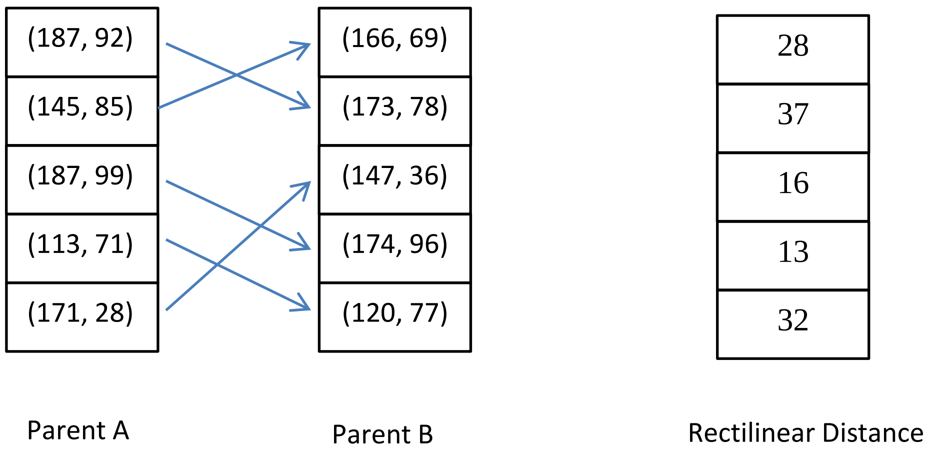

Figure 4.7 Hungarian match crossover pairing

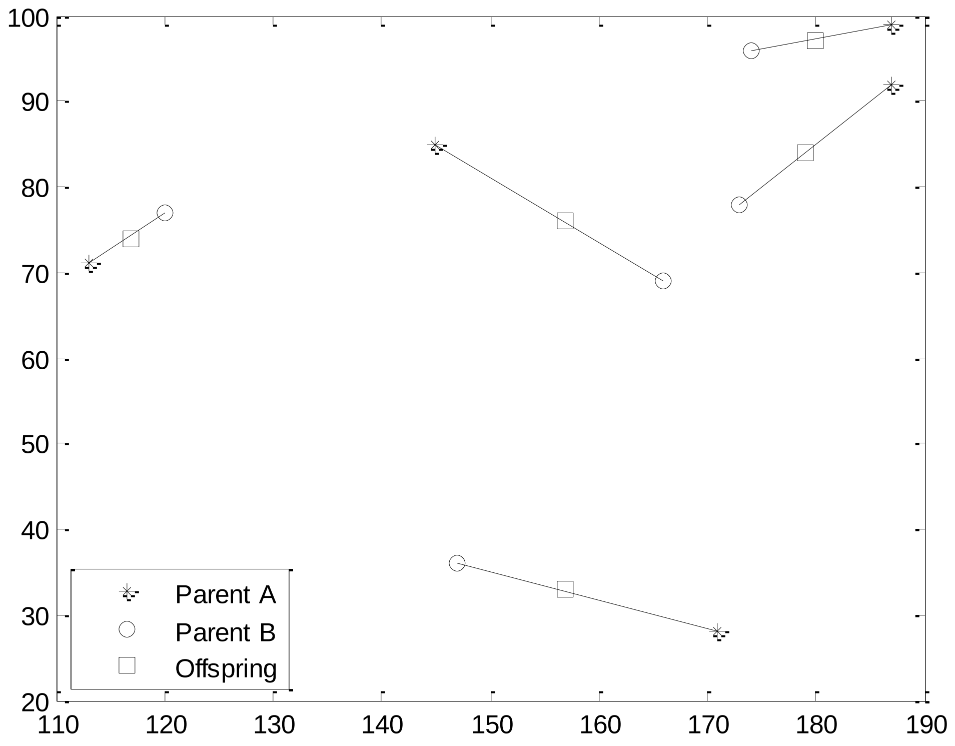

Figure 4.8 Hungarian match crossover pairing in a graph

The formulation for the Hungarian model to find the matching genes so that we can process the crossover operation is presented below. Assume that we have two parent chromosomes 1 and 2, a_{i1} is the crate length and b_{i1} is the crate width of gene *i* for parent chromosome 1, while a_{j2} is the crate length and b_{j2} is the crate width of gene *j* for parent chromosome 2. Then $c_{ij} = (|a_{i1} - a_{j2}| + |b_{i1} - b_{j2}|)$ is the cost of matching gene *i* of chromosome 1 to gene *j* of chromosome 2. *n* is the number of pairs of genes in the chromosome. The problem of finding matching genes can be modelled as an assignment problem shown below:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.8}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for } i=1,.., n \tag{4.9}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } j=1,.., n \tag{4.10}$$

$$x_{ij} \in \{0,1\} \text{ for } i=1,.., n, j=1,.., n \tag{4.11}$$

x_{ij} is the binary variable where it is 1 if gene *i* of chromosome 1 is matched to gene *j* of chromosome 2, otherwise it is 0.

After the above assignment problem is solved using the Hungarian algorithm, crossover is performed.

4.5.5 Mutation Operator

The purpose of mutation is to bring in random traits and increase variability of the population to allow for exploration. Mutation is applied using a random number and compared with mutation probability to decide if mutation should be performed. The mutation rate is set and if the random number is less than mutation rate, a gene of the offspring is randomly selected to be mutated. It is randomly mutated to another customer demand type among its neighbours.

4.5.6 GA Algorithm

The algorithm for the GA implemented follows the steps outlined below:

1. Initial population - Generate initial population.
2. Parent selection - Based on tournament selection whereby a random set of tournament size individuals is selected for tournament. The individual with lower *F(x)* wins (minimization). Two winning individuals are selected as parents for reproduction.
3. Reproduction - A new offspring is produced using the following steps:
    a. A cost matrix c_{ij} is constructed using the pairing of the selected pair of parents where the cost is calculated using rectilinear distance
    b. Hungarian match is used to find the nearest neighbour to preserve the neighbourhood property.
    c. Once matches are found, whole arithmetic crossover $\beta^{s_1} = \alpha.\beta^{f_1} + (1-\alpha).\beta^{f_2}$ is applied; if not feasible, repair and modify to nearest neighbour.

    d. Mutation is applied using a random number and compared with mutation probability to decide if mutation should be performed.

4. The cycle of parent selection, crossover, and mutation is repeated to generate new individuals.
5. Fitness function-All individuals in the population are evaluated for *F(x),* the fitness value based on total cost of individual.
    a. Individuals with lower *F(x)* are preferred (minimization of fitness value)
6. Elitism is the preservation of best solutions of the population pool for the next generation.
    a. At the end of each generation, elite individuals with best fitness values are selected to remain and copied into the next generation's population
    b. Individuals with unsatisfactory fitness values are discarded
    c. The best elite individuals are selected to remain for the next generation's population P_{t+1}
7. The new population P_{t+1} replaces the current population P_t
8. Exit when set number of generations G is reached
9. The best solution is found

The algorithm for the GA can be illustrated using the flowchart as shown in Figure 4.9 below. After the best chromosomes are chosen for copying into the next generation, marginal improvement as discussed in Section 4.4 is applied to a small percentage of individuals. The marginal improvement step is intended to speed up the process of finding better solutions. After this is done, one generation is complete and progresses to the next generation. The GA algorithm is run for a set number of generations before terminating. The output is the minimum value when the set number of generations is reached.

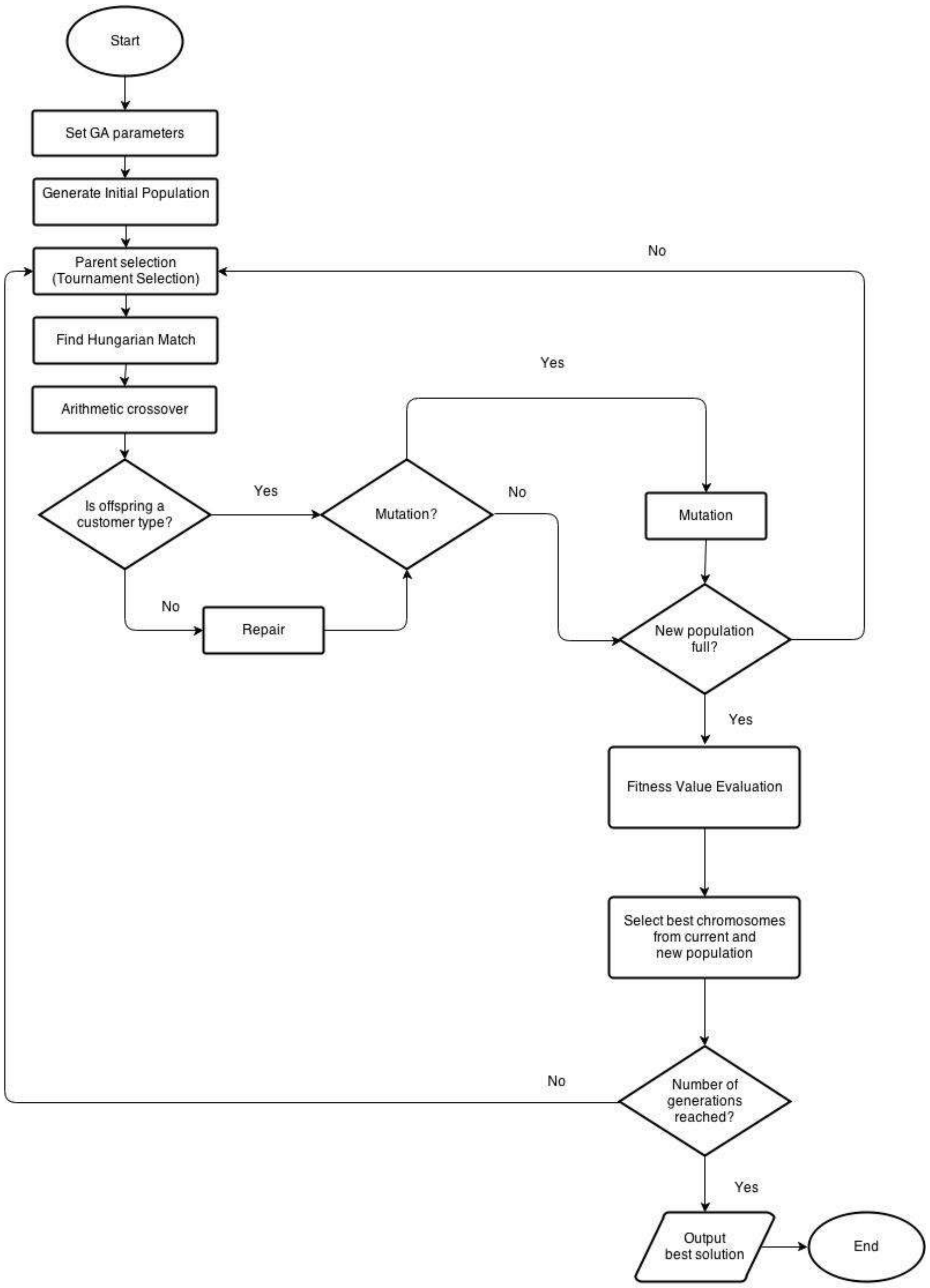
Start
Set GA parameters
Generate Initial Population
Parent selection (Tournament Selection)
No
Find Hungarian Match
Yes
Arithmetic crossover
Is offspring a customer type?
Yes
Mutation?
No
Mutation
No
Repair
New population full?
Yes
Fitness Value Evaluation
Select best chromosomes from current and new population
No
Number of generations reached?
Yes
Output best solution
End

Figure 4.9 Flowchart of GA algorithm

4.5.7 Numerical Experiments

Numerical experiments were conducted using the proposed GA algorithm.

4.5.7.1 Comparison with Enumeration Method

In a standard GA, parameter tuning is crucial to the evolutionary computation of the problem. The efficiency of a GA is greatly dependent on its tuning parameters. The parameters include population size, tournament size, probability of mutation, α value in the crossover operator and the number of generations. Design of experiments can be applied to find the optimal settings for all the parameters. On the other hand, the paramaters can be tuned one at a time although this may produce suboptimal solutions because the parameters may interact in a complex way. Despite the disadvantage, many researchers opt to tune the parameters "by hand" which is testing different values and selecting the value with the best results due to time constraints. When building a GA, there is a need to guesstimate what the optimal values are for a lot of parameters. Mostly there is a lot of trial and error. In this thesis, the same approach is adopted to tune the parameters by experimentation.

In a GA algorithm, increasing the population size will increase the accuracy of the GA. Basically, the bigger the population the better, but realistically there is a need to make compromises in order to run the algorithm in a reasonable amount of time. Meanwhile, on the other end of the spectrum, if a population size is too small, it is possible that the GA will converge to a local optimum value as there is a lack of diversity as weak values are generally "pushed out" to make space for the population size. Generally, the rule of thunb for a population size is in the range of 30-100. It must be noted that increasing the population size will also increase the time needed to converge. In view of this, we would like to offset the accuracy with the time it takes to converge. In order to determine the parameter of population size, the population size is first set to an empirical value for a run with input size *n*. Then the population size is observed to see whether it holds and remains constant as the size of *n* is increased from 2 to 20. For this problem, because the results can be compared to the enumeration method, the convergent values can be evaluated as whether they are optimal or not. For all sizes *n* from 3 to 20, the population size is 10 and the number of generations it took to converge to the optimum value is always either 2 or 3. Hence it can be concluded that a small population size of 10 is sufficient for this small-scale problem of determining two sizes for *n* ranging from 2 to 20.

Tournament size is the parameter which determines the selective pressure of a tournament selection. The size of the tournament selection is relatively small compared to the population size. The ratio is indicative of the selective pressure. Due to the coding implementation, the population size must be divisible by the tournament size. Experimenting with different values of the tournament size from 2 to 10, the same convergent values were obtained in the same number of generations. Hence the tournament size can be set to any value in this range.

Besides population and tournament size, mutation probability is another parameter that is important as the nature of genetic algorithm is randomization. There is some bias inherent in the mutation effect where the larger the current value is, the larger the mutation will be. Hence, the mutation probability is generally set low. The probability of mutation is tuned by comparing the results obtained when it is changed from 0.01 to 0.10 in steps of 0.01. The same optimal results were obtained for each setting thus the probability of mutation can be set to 0.01.

Next, crossover operator is also considered for tuning. The α value is a random weighting factor chosen before each arithmetic crossover operation. It is a random number generated from the uniform distribution on the interval [a,b], usually between 0 and 1. This has the advantage of producing feasible offspring within the solution space. However, if the optimum lies near the solution space boundary, then it has the disadvantage of producing offspring toward the interior of the solution space. The value of α is initially set to [0,1].

Lastly, the number of generations is tested from 100 to 10 in steps of 10. The same optimal results were obtained for each calibration. Since a higher number of generations takes a longer time, the number of generations is set to the lowest value of 10 only.

Similarly, this approach was used to calibrate the parameters for the three-size problem and four-size problems and it was found that the same parameters as shown in Table 4.5 are still valid.

Parameters

Type of coding	Real value
Initialization	Random
Population size	10
Tournament size	2
Probability of mutation	0.01
α	[0,1]
Number of generations	10

Table 4.5 Parameters of GA experiment I

Tables 4.6-4.8 show the results for comparison with the enumeration method for two-size, three-size and four-size problems. Without loss of generality, in these experiments, we assume that the demand size is 1 and variance is 0. The results show that the GA algorithm is able to converge to the global optimal solution for the following problems when compared to the enumeration results.

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	**(Enumeration) Cost**	**(GA) Cost**
2	**2**	2	1	2	2	20	20
	3	3	2	3	3	153	153
	4	4	2	4	4	640	640
	5	5	3	5	5	1925	1925
	6	6	3	6	6	4860	4860
	7	7	4	7	7	10339	10339
	8	8	5	8	8	20288	20288
	9	9	5	9	9	36369	36369
	10	10	6	10	10	61600	61600
	11	11	6	11	11	99341	99341
	12	12	7	12	12	153072	153072
	13	13	7	13	13	229333	229333
	14	14	8	14	14	330848	330848
	15	15	9	15	15	467775	467775
	16	16	9	16	16	645376	645376
	17	17	10	17	17	873647	873647
	18	18	10	18	18	1163808	1163808
	19	19	11	19	19	1523059	1523059
	20	20	12	20	20	1971200	1971200

Table 4.6 Comparison between GA and enumeration for two-size problem

K	***n***	x_1^*	y_1^*	x_2^*	y_2^*	**(Enumeration) Cost**	**(GA) Cost**
3	**3**	3	1	3	2	126	126
	4	2	4	4	2	512	512
	5	4	4	5	2	1587	1587
	6	3	6	6	3	3888	3888
	7	5	6	7	3	8482	8482
	8	4	8	8	4	16384	16384
	9	6	8	9	4	29709	29709
	10	5	10	10	5	50000	50000
	11	7	10	11	5	80886	80886
	12	6	12	12	6	124416	124416
	13	8	12	13	6	186271	186271
	14	7	14	14	7	268912	268912
	15	9	14	15	7	380682	380682
	16	8	16	16	8	524288	524288
	17	10	16	17	8	711417	711417
	18	9	18	18	9	944784	944784
	19	11	18	19	9	1240174	1240174
	20	10	20	20	10	1600000	1600000

Table 4.7 Comparison between GA and enumeration for three-size problem

Table 4.8 shows the comparison between GA with enumeration, MIBS I, MIBS II and MIBR costs for the four-size problem. The results show that the GA performs better than the marginal improvement methods.

K	*n*	x_1^*	y_1^*	x_2^*	y_2^*	x_3^*	y_3^*	**(Enumeration) Cost**	**(MIBS I) Cost**	**(MIBS II) Cost**	**(MIBR) Cost**	**(GA) Cost**
4	**4**	4	2	2	4	4	3	456	456	456	456	456
	5	5	2	2	5	5	4	1355	1355	1355	1355	1355
	6	6	2	2	6	6	4	3360	3360	3360	3360	3360
	7	7	3	3	7	7	5	7231	7231	7231	7231	7231
	8	8	3	3	8	8	6	14048	14400	14048	14048	14048
	9	9	4	3	9	9	7	25515	25515	25515	25515	25515
	10	10	4	4	10	10	7	42820	42820	42820	42820	42820
	11	11	4	4	11	11	8	68959	68959	68959	68959	68959
	12	12	5	4	12	12	9	106704	106704	106704	106704	106704
	13	13	5	5	13	13	9	-	158925	159757	158925	158925
	14	14	5	5	14	14	10	-	230384	230720	230384	230384
	15	15	6	6	15	15	11	-	324975	324675	324675	324675
	16	16	6	6	16	16	11	-	449056	449056	449056	449056
	17	17	7	6	17	17	12	-	613547	608022	608022	608022
	18	18	7	7	18	18	13	-	812718	812718	807300	807300
	19	19	8	7	19	19	14	-	1068560	1059079	1059079	1059079
	20	20	8	7	20	20	15	-	1369500	1372720	1377200	1369500

Table 4.8 Comparison of GA to enumeration and MIBS I, MIBS II and MIBR for four-size problem

Note: For N=9, the optimal sizes from enumeration are (9,3) and (3,9), (9,6) and (9,9) with the same total cost of 25515.

4.5.7.2 Numerical experiment of a medium size problem

After comparing the GA to the enumeration and marginal improvement methods for the trivial problems, GA is applied to explore for a medium size problem based on 50 different types of crate lengths and 50 different types of crate widths. The GA finds for five and ten sizes out of these customer demand sizes.

In order to determine the parameter of population size for the medium-scale problem, the population size is tested by increasing the size from 10 to 100 in increments of 10. It was observed that a population size of below 50 did not always produce a value that is lower. There was a tendency to get stuck in local minima. Thus the population size is set to 50.

Another parameter is the tournament size. Experimenting with different tournament sizes ranging from 2 to 25, it was observed that a tournament size of 2 did not always produce the lowest value while a tournament size of 5 did not have the same issue. Therefore the tournament size is set to 5.

In addition, the probability of mutation is tuned by comparing the results obtained when it is changed between 0.01 and 0.25. However, it was found that mutation rate of 0.15 and above did not always give the lowest value so it should be lower than 0.15. As such the mutation rate is maintained at 0.01.

The value of α is initially set to [0,1]. The range is gradually increased to [-0.25,1.25] to allow more diversity. A broader range of the α value was found to avoid the pitfall of falling into local minima.

Then, the number of generations is tested from 1000 to 100 in steps of 100. Next, it is tested from 100 to 10 in steps of 10. Because the algorithm converges quickly, the number is generations is set at a conservative estimate of 50.

Parameters

Type of coding	Real value
Initialization	Random
Population size	50
Tournament size	5
Probability of mutation	0.01
α	[-0.25,1.25]
Number of generations	50

Table 4.9 Parameters of GA experiment II

In order to increase the efficiency of the algorithm, at the end of each generation, the top 10% of the population is processed with marginal improvement method. Because of this, the first generation of the GA includes a solution from the marginal improvement method. The percentages reflect the improvement of the value from the initial generation. Both figures 4.10 and 4.11 show the evolution of the minimum value in each generation. GA is able to improve on the marginal improvement method by 13.62% for five sizes and 11.68% for ten sizes.

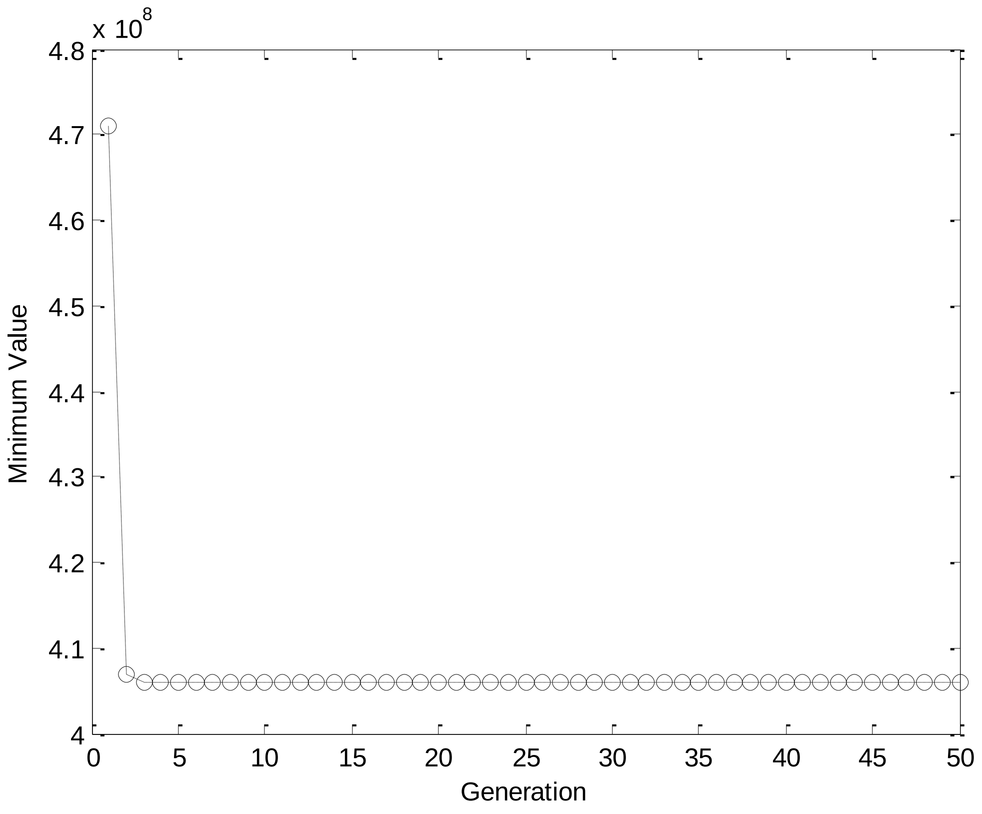

Figure 4.10 Convergence for a medium problem GA (5 sizes)

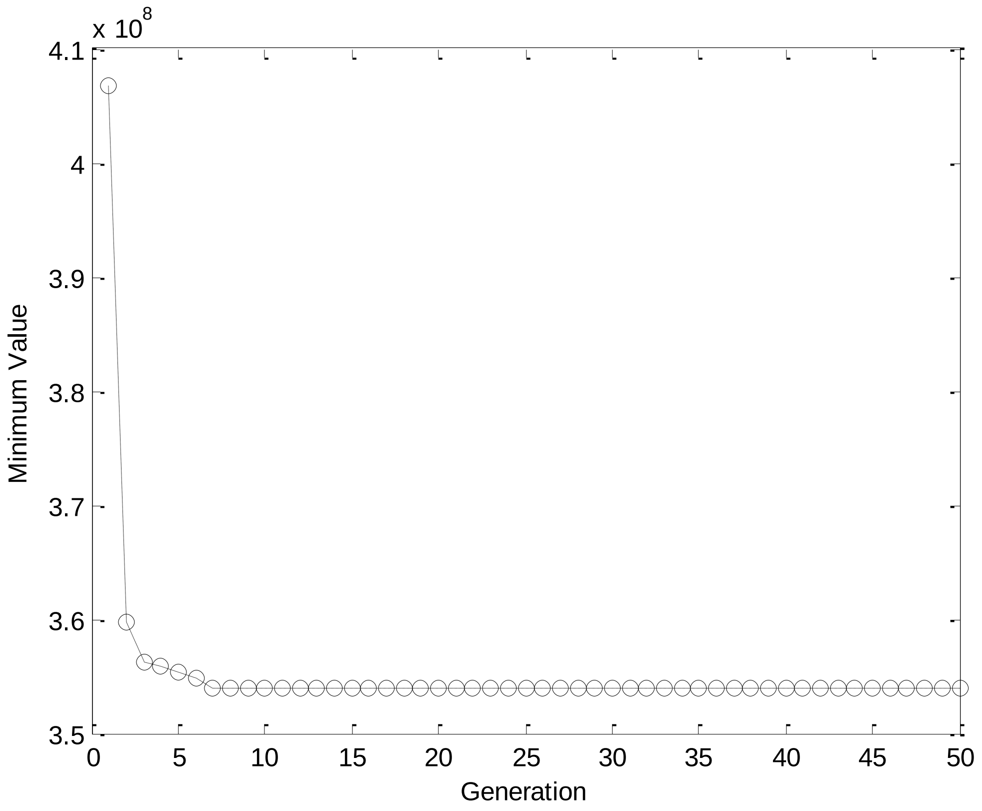

Figure 4.11 Convergence for a medium problem GA (10 sizes)

4.5.7.3 Numerical experiment of a large size problem

GA is then applied to explore for a large size problem on the scale of 100 types of crate lengths and 100 types of crate widths. There are in total 10000 customer demand sizes that are in the search space. The GA finds for ten ~~optimal~~ sizes out of these demand sizes. There are ${}^{9999}C_9 = 2.746 \times 10^{30}$ possibilities in this large size problem.

When building the large size problem, the algorithm is more sensitive to the GA parameters and tuning is necessary. The parameter tuning follows the trial and error approach as described previously. Firstly, the population size is tested by increasing the size from 10 to 100 in increments of 10. The population size and the tournament size were calibrated together. When the tournament size is 5, the results did not show lowest results for population sizes of 10, 20, 30, 40, 70, and 80. Meanwhile, when the tournament size is increased to 10, the results showed that the results were better for population size of 100. After investigating different scenarios, the mutation rate is maintained to 0.01. As for the crossover operation, the value of α is initially set to [-0.25,1.25]. The range is then broadened to [-0.50,1.50]. Finally, the number of generations is tested from 1000 to 100 in steps of 100. Because the algorithm converges quickly, the number is generations is likewise maintained at 50.

Parameters

Type of coding	Real value
Initialization	Random
Population size	100
Tournament size	10
Probability of mutation	0.01
α	[-0.50,1.50]
Number of generations	50

Table 4.10 Parameters of GA experiment III

Figure 4.12 shows the convergence of the objective value in 50 generations. As described for the medium size problem, the top 10% of the population is directed to the marginal improvement method. GA is then able to improve on the marginal improvement method by 37.85%.

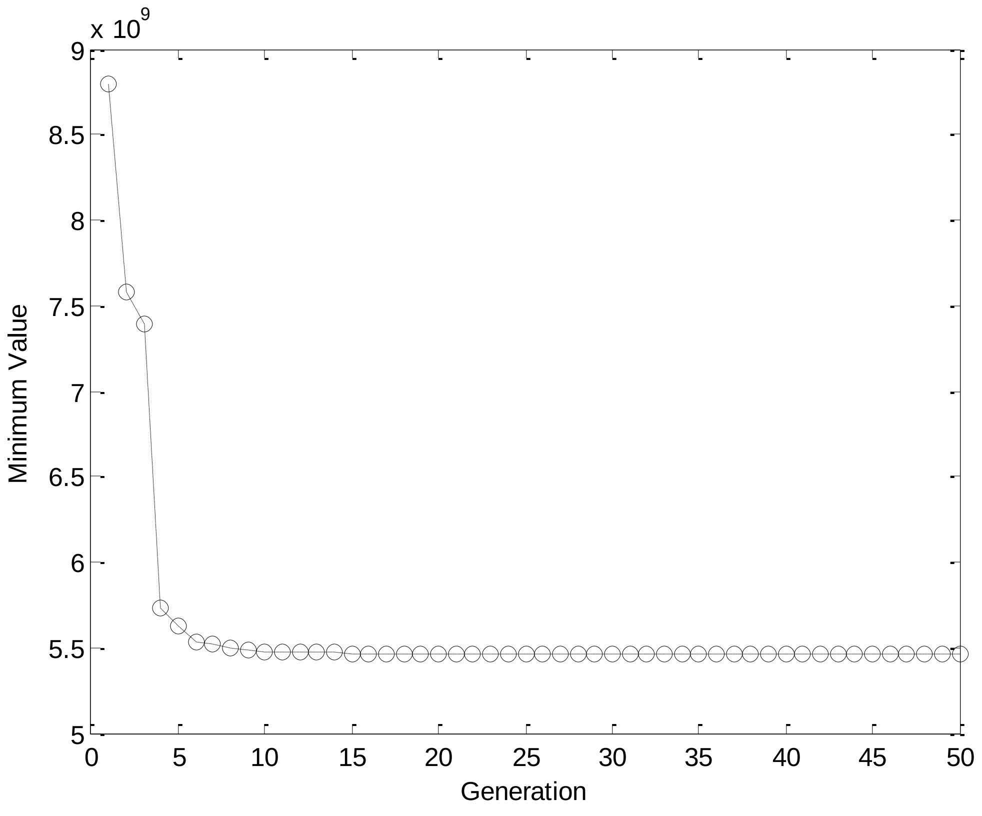

Figure 4.12 Convergence for a large problem GA

Figure 4.13 shows the decrease in objective value when the number of types is increased from 2 to 10 for the same problem. This is due to the better fit of the rolls into the crates and hence the cost of the total volume decreases and the effect is lower total cost. Meanwhile Figure 4.14 shows the increase in objective value for the above problem when variance level is calibrated between 2 to 10 for ten types. This is due to the effect of higher cost of safety stock which leads to higher overall cost. From here, it can be expected that there is a balance of trade-off between total volume cost and inventory cost which can help determine the number of ideal types in Section 4.5.2.

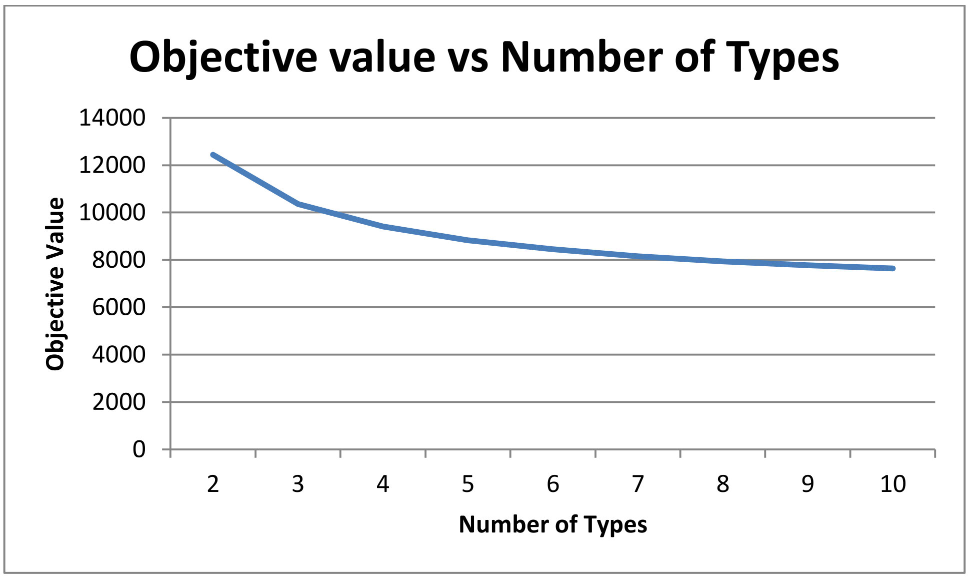

Figure 4.13 Objective value vs. increasing number of crate types

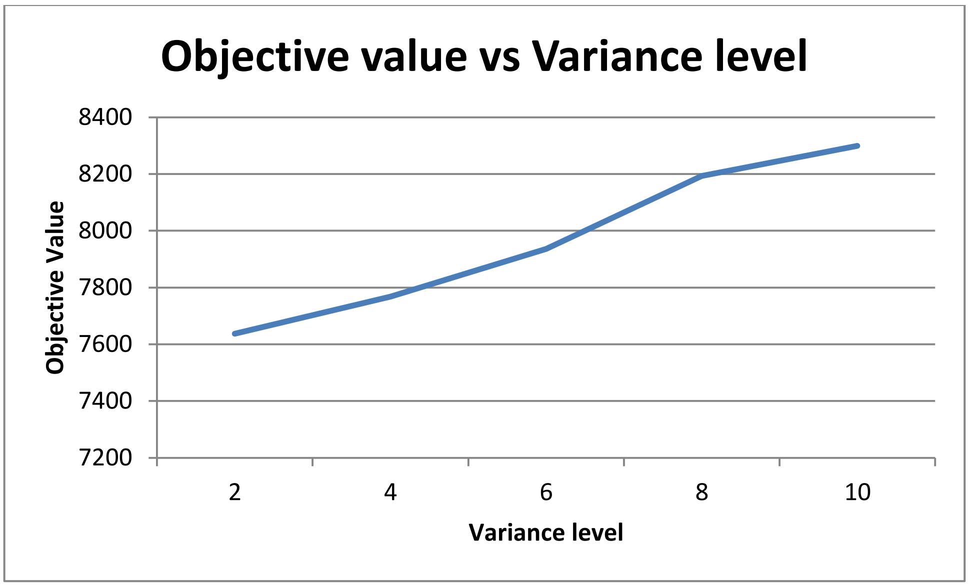

Figure 4.14 Objective value vs. increasing variance level

4.5.8 Determining the number of types

We can determine the suitable number of types by using the proposed GA algorithm by conducting the GA iteratively. For example, GA is used to explore for a small size problem on the scale of 50 types of crate lengths and 50 types of crate widths. The GA is used to determine the number of optimal sizes from 5 to 20 for $\frac{p}{h}$ ratio of 10^{-6} and we assume that the demand size and variance are 1. The scale of the $\frac{p}{h}$ ratio is important as this determines the trade-off between the inventory cost and the total packing volume cost. Figure 4.15 shows that the number of crate types for this problem is 12 where it is at the minimum and increasing the number of types beyond 12 does not bring in cost savings from fitting the rolls into crates more effectively because this is countered by the higher safety stock cost of additional crate types. However, the values for 9, 12 and 14 types are very close and so depending on the situation, it may be more favourable to opt for the smaller or larger number of crate types instead. This approach of iteratively conducting the GA algorithm to find the number of crate types and their sizes ties back to the original problem described in Section 3.2 for the 1D problem. The Hungarian-based GA algorithm is able to minimize total volume of crates and inventory stock for a 2D problem instead of loss of length and inventory stock only.

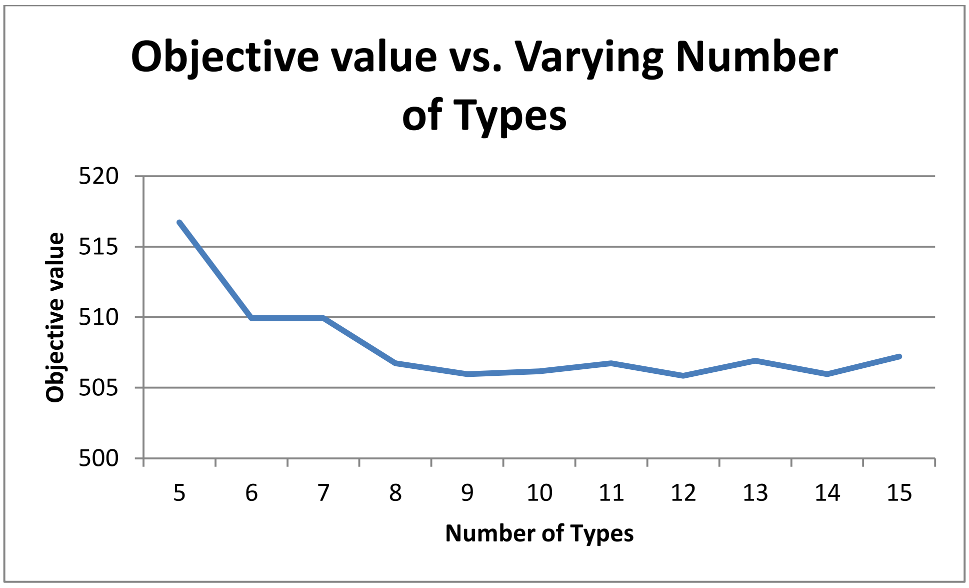

Figure 4.15 Objective value vs. Varying number of types

5 Bin packing (Rectangular)

Packing problems are common in the industry of shipping and logistics. Some examples of issues that arise are the choice of product shapes and sizes, choice of containers, packing layout and sequence, fluctuations in demand, rotations and space limitations. There are many varieties of products that come in a lot of different shapes and sizes, depending on the industry and type of business. Packing problems usually consist of packing products of different sizes. Compared to packing products that come in a single size only, packing products that come in different sizes is not as direct. In terms of product shapes, there are regular shapes such as circles, squares and rectangles while on the other hand, there are irregular shapes which make it harder to visualize generally. However, it is worth noting that although products come in many shapes, it is a usual practice to package the products in regular shape boxes for shipping purposes. Thus, in those cases, the packing problem of irregular shapes can be reduced to a packing problem of regular shapes for tractability and ease of handling. Aside from product shape and sizes, the choice of container is also important, as there are20-ft and 40-ft, hi-top, reefer and other types of containers available. The choice of containers has direct implication on the final cost. To add on to the problem, fluctuations in demand make it harder to predict when and how to pack the products to be shipped to customers therefore robust decisions are preferred and considered for long-term cost savings. Lastly, sometimes products may either have fixed orientations or can be rotated. The many factors that weigh in the packing process make it a difficult decision to manage. As such, inefficient packing has resulted in many partially-filled containers and unnecessary expenses. Owing to the increasing cost of shipping, many companies desire to improve on their packing process to reduce the number of containers required and total shipping cost.

In this section, the packing problem is inspired by a company that sells and ships products that are in the shape of cylindrical rolls. The rolls have a few types of thicknesses available as each unique thickness caters to a customer’s industrial purpose. The rolls are also cut to any roll length depending on customer specifications. The rolls are each packaged in rectangular box sizes before they are packed in shipping containers for shipping worldwide. The company stocks and uses a few known, standard-size and regular-shape boxes to contain each roll as packaging. With this decision, it is inevitable that there will be some wastage of space inside the boxes even before the actual packing problem of the boxes into containers commences. This makes it even more worthwhile to ensure that the next step of containerization packing problem is considered for the improvement of shipping cost.

Following on the choice of boxes, the company also must decide on the type of shipping container to use. If a smaller container is chosen, the packing may be denser but more containers will be needed. On the other hand, larger containers are preferred when the orders are large but more space wastage may occur. Afterwards, the packing problem starts with how to load the boxed products into the containers to maximize on space. Here, there are many practical difficulties encountered in the loading of the containers due to the amount of physical labour involved. Firstly, the orientations of the products must be correct, and the layout and sequence must be such that it is possible to load starting from the back of the container and extending to the front of the container for the worker to proceed smoothly. Not only that, the workers will stack items from the bottom to top and hence layer packing is also more practical. Finally, because in this case the products are fairly fragile, all empty space must be padded with airbags for maximum protection of the products. Otherwise, the boxes will move around during transport and may cause the products to spoil upon arrival. If the boxes are packed dense enough, this extra protection is not necessary and the cost of the paddings can be reduced. Thus it is imperative that the packing solution is made more effective to maximize on space and minimize cost based on the above mentioned limitations.

In this operational packing problem, a demand realization which consists of a set of products in different dimensions is to be packed into as few identical containers as possible for shipping to customers. The products are individually packaged and transported in a known number of standard rectangular box sizes. The problem is reduced to a rectangular packing problem by pre-sorting the demand according to the height of the boxes. The container is then packed layer by layer whereby each layer comprises of boxes sharing the same height. In each case, the layer is packed as a rectangular packing problem and a relaxed LP with improvement is proposed to minimize the number of layers. In order to generate feasible packing patterns, rectangular packing heuristics from available literature such as the steplike stacking heuristic and maximal rectangle heuristic are applied. Following that, new columns are generated to both find improved patterns and explore new ones for the future layers. Finally, in the case of multiple heights, the container is packed as a 1D packing problem to minimize the number of containers.

Many conventional packing heuristics tend to use a greedy approach. Items are placed and then never considered again in subsequent steps. Items are packed densely with minimum waste to the first few containers (due to a large assortment of items available for choice), which are then followed by last few vastly sparse containers to accommodate the leftovers. However, as all items must be packed, it is advantageous to consider repacking the items in different patterns so as to average out the load. In this thesis, the approach to the abovementioned packing problem comprises of a mathematical model to solve the rectangular packing problem. With the use of the relaxed LP, we can gather information on the patterns which contribute to the high surplus of demand. Then using this information, improvement method is used where solutions whose patterns contribute to high surplus of demand in the optimal solution are re-evaluated. New and improved patterns are obtained and the method is repeated until the surplus is reduced to a satisfactory low value. With this method, current packing patterns can be improved and new ones can be explored for the next demand realization. After a given satisfactory number of runs, we can find a good set of packing patterns. In this section the problem is defined in Section 5.1 where the rectangular packing problem is described and difficulties in solving the problem are highlighted. Section 5.2 presents the formulation of the problem. In Section 5.3, the solution methodology is presented. The third section solves the case of single height with improvement method. In the improvement method, how the initial column is first obtained is described in detail here. Section 5.4 illustrates the case of different heights of boxes with cutting stock approach. Finally the numerical results for different scenarios of random demand realizations are finally presented in Section 5.5.

5.1 Problem Description

The packing problem's main objective is to minimize the total number of containers and ultimately reduce shipping cost. The input to the problem is a set of rolls which have been individually packed into rectangular boxes. As mentioned above, there are a few standard-size, rectangular boxes where the dimensions of each type of box are given. Therefore, the processed customer demand is a set of box sizes in given dimensions. In a single customer demand realization, all boxes must be packed into the containers, leaving no boxes behind. The output of the problem is the number of containers used. We assume using only one type of container and that there is unlimited number of containers available, i.e. there is no restriction on the number of containers. The dimensions of the container are also given. Henceforth, the container is also referred to as bin.

The overall packing problem described is a 3D bin packing problem which is NP-hard. This problem has many assorted small items and a single type of bin size. The number of bins available is not restricted. The packing is packed by layer, and it can be assumed that the layers can be feasibly arranged such that the denser solutions are placed bottom first before sparse solutions if any. Based on the characteristics of the problem and from the typology in literature (Dyckhoff, 1990) and (Wäscher et al., 2007), our packing problem can be categorized as a type of 2D dimensionality, input (value) minimization, weakly heterogeneous assortment and dimensions fixed for the small objects, weakly heterogeneous and dimensions fixed for the large objects, multiple stock size cutting stock problem MSSCSP.

The problem can be solved using various methods from the literature with well-studied 3D bin packing algorithms like the branch-and-bound and other bin packing methods. Despite that, the problem here has a special characteristic that justifies a different approach. In particular, the problem has a low number of types of boxes and their given dimensions. Because there are only a few types of boxes, it warrants an approach to pack the boxes by layer as there are only a few standard heights. As such the boxes are pre-sorted into groups of the same height; the container packing problem can be reduced into a 2D rectangular packing problem whereby the shipping container is assumed to be composed of several layers of rectangular area. In this approach, the packing problem has two cases, where the first case is assuming that there is only one height of boxes (which describes the packing of boxes into layers) and the second case is there are several heights of boxes (which describes the packing of layers into bins). The rectangular area of the layer corresponds to the floor area of the shipping container. The objective of the reduced problem is to minimize the number of packing layers which is assumed to unequivocally reduce the number of the containers in the original container packing problem subsequently.

Although the pre-sorting results in suboptimal solution, the benefits outweigh the reduced optimality. There is a very practical reason to reducing the original 3D bin packing problem (3D-BPP) into 2D rectangular packing problem because as mentioned previously, there are a known number of standard boxes and hence, there are only a few heights of boxes to tackle. Thus the problem is reduced by pre-sorting the original demand into a few groups of boxes of the same height. Then each group is treated as packing by layers. Since the demand of each group is generally large enough to comprise of several layers, the objective is to minimize the total number of packing layers. After the rectangular packing stage, the layers are then packed into containers (bins) using the heights of the layers versus the height of the containers (bins). Even though reducing the problem from 3D to 2D by pre-sorting into shared heights makes the problem sub-optimal, this approach yields practical results. The actual packing problem requires manual labour and packing by layer is advantageous as it is simpler to understand and easier to manoeuvre with the heavy equipment and products. For the worker, it is easier to visualize and to arrange for movement in logical sequence to complete the packing. A packing instruction in original 3D bin packing solution is not easy to read and follow for the layman. It might require more skilled workers who demand higher wages. Another possibility is that this will slow down the packing process making the trade-off for space optimization unpredictable.

This section is organized as follows; for each type of height, the 2D bin packing problem is solved using the rectangular packing problem with improvement method as described in Section 5.3. When all the heights are packed into layers, the layers are then packed into the containers (bins) using 1D packing to minimize the number of bins. This cutting stock approach completes the overall 3D bin packing process after reducing the problem into 2D. The cutting stock method is described in Section 5.4.

5.2 Problem Formulation

There are many ways of representing a packing problem, for example arch flow model, set covering model, convexity model, position-oriented model etc. In a paper, the 3D-BPP is modelled by (Hifi, Kacem, Nègre, & Wu, 2010) as a mixed integer linear programming model using inequalities to describe the spatial constraints and minimize the total number of identical containers m with fixed dimensions length L, width W and height H. There are n rectangular items i to be packed which consist of several types of boxes with different lengths $l_{i,}$ widths w_i and heights $h_{i.}$ The coordinate (x_i, y_i, z_i) is used to describe the *left-bottom-back* coordinate of item i and that the coordinate of *left-bottom-back* corner of the container (bin) is (0, 0, 0). γ_i is defined as the label of the bin to which item i is assigned (i = 1, ..., n). The aim is to minimize the greatest label of the used bin γ = $\max_{1\leq i\leq n}\{\gamma i\}$. From the 3D-BPP model described above, the reduced 2D-BPP can be formulated as follows:

Parameters

L	Container length
W	Container width
l_i	Length of item i
w_i	Width of item i
n	Total number of items i

Decision variables

x_i	Geometrical location of item i (left-coordinate)
y_i	Geometrical location of item i (back-coordinate)
β	Bin (container) index
l_{ij}	1 if item i is in the left of item j; 0 otherwise
b_{ij}	1 if item i is at the back of item j; 0 otherwise
c_{ij}	1 if $\beta_i < \beta_j$; 0 otherwise

$$min\ \beta$$

s.t.

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} = 1, i < j = 1,...,n \qquad (5.1)$$

$$x_i - x_j + L\ (l_{ij} - c_{ij} - c_{ji}) \leq L - l_i, i \neq j = 1,...,n \qquad (5.2)$$

$$y_i - y_j + W(b_{ij} - c_{ij} - c_{ji}) \le W - w_i, i \ne j = 1,...,n \quad (5.3)$$

$$\left(\overline{\beta} - 1\right)(l_{ij} + l_{ji} + b_{ij} + b_{ji}) + \beta_i - \beta_j + \overline{\beta} c_{ij} \le \left(\overline{\beta} - 1\right), i \ne j = 1,...,n \quad (5.4)$$

$$l_{ij}, b_{ij}, c_{ij} \in \{0,1\}, i \ne j = 1,...,n \quad (5.5)$$

$$0 \le x_i \le L - l_i, i = 1,...,n \quad (5.6)$$

$$0 \le y_i \le W - w_i, i = 1,...,n \quad (5.7)$$

$$0 \le \beta_i \le \beta \le \overline{\beta}, i = 1,...,n \quad (5.8)$$

$l_{ij} = 1$ if item *i* is in the left of item *j*, $b_{ij} = 1$ if item *i* is in the back of item *j* and $c_{ij} = 1$ if $\beta_i < \beta_j$. The first three constraints ensure that no overlap exists between two packed items. The parameter $\overline{\beta}$ is a valid upper bound on *β*. Constraint (5.4) implies that when $c_{ij} = 1$ or $c_{ji} = 1$ the items *i*, *j* are located in different layers and when one of l_{ij}, l_{ji}, b_{ij}, b_{ji} is equal to 1, items *i* and *j* are necessarily located in the same layer. There are two dimensions, namely along the *x*-axis and the *y*-axis, parallel to the length of the container and the width of the container respectively. The layer is the horizontal layer of the container which is the floor area of the container.

The model illustrates the problem of packing same-height crates into a rectangular bin with known dimensions. While the formulation can give optimal results for the packing for the 2D-BPP problem, the scale of the MILP increases greatly with the size of the problem. For example, the number of decision variables grows exponentially with problems of large number of items. In daily practice, solving the MILP will demand higher complexity and computation. Therefore instead of using MILP to solve, there are many authors who prefer the use of heuristics as the packing problem is an NP-hard problem. In this section, the problem is also solved using heuristics and the approach is documented in Section 5.3. The methodology uses packing heuristics for rectangular packing of the bin by layer. The 2D bin packing problem is solved for each group of height of the boxes.

The assumptions for the model are:

1. The layers of the containers are independent and it is assumed that the layers can be feasibly placed on one another. Weight and possible symmetrical placements are not considered.
2. All demands must be satisfied, i.e. all items must be packed for a particular demand realization.
3. The height of boxes is a constant in each demand realization.

5.3 (2D-BPP) Layer Packing

An enumeration of all the packing patterns is not feasible for a reasonably large-sized problem. However, by using the idea of generating new columns, this allows us to build the set of packing patterns by starting with a smaller set of solutions and then improving the solution by generating new columns.

5.3.1 Layer Packing with Column Generation

In this section, the column generation approach is investigated. The integer programming model below describes the 2D (rectangular) layer packing problem using the cutting stock approach. The area of the layer is constrained by the floor of the container with dimensions of L (length of the container) and W (width of the container). There are n items i, each of which has an associated size of l_i, (length of item i), w_i (width of item i) and demand d_i. We use a column vector A_j to represent a packing pattern p_j. The elements of A_j, a_{ij} then corresponds to the number of pieces of item i in the pattern p_j. The elements of A_j must be all non-negative integers. Each pattern p_j must be a feasible 2D packing pattern for the rectangular area of L by W. Let J be the total number of distinct feasible cutting patterns which is the number of vectors A_j satisfying the constraints. All demands of item i, d_i must be satisfied. The objective is to minimize the total number of layers packed with pattern p_j, X_j. The integer programming model presented below can be relaxed to give a lower bound. Since there are n demand constraints, there are at most n non-zero variables.

Inputs and Parameters

L	Container length
W	Container width
S	Size of the rectangular layer (the floor of the container)
l_i	Length of item i
w_i	Width of item i
s_i	Size of item i
n	Total number of items i

d_i	Demand of items i of size l_i and w_i to be packed
a_{ij}	Number of items i of size l_i and w_i packed in layer j
A_j	Packing pattern p_j
J	Total number of distinct feasible cutting patterns p_j

Decision Variables

X_j	Number of layers packed with pattern p_j

min $$\sum_{j=1}^{J} X_j \tag{5.9}$$

subject to

$$\sum_{j=1}^{J} a_{ij} X_j \geq d_i \quad \text{for } i=1,.., n \tag{5.10}$$

$$X_j \geq 0 \quad \text{for } j=1,.., J \tag{5.11}$$

$$X_j \text{ integer} \quad \text{for } j=1,.., J \tag{5.12}$$

In the objective function (5.9), we minimize X_j. Therefore the number of containers can be obtained by dividing the objective value by the number of layers allowed in a container. Constraint (5.10) dictates that the decision variable on the packing layers X_j has to at least satisfy all the demand of items to be packed. Constraint (5.11) is straightforward, restricting the decision variable X_j to be positive and constraint (5.12) additionally restricts it to integers only.

The idea of column generation is to start with a few patterns and generate new ones as needed. Starting with an initial basis, we then determine if all non-basic columns have reduced cost ≥ 0 and if not, then find a column with negative reduced cost. In order to find an initial basic feasible solution, it is important to verify that the packing pattern is feasible. The patterns will be naïve ones where for every item *i*, we try to fit the maximum number of each into the rectangular area of *L* by *W*. The reduced cost of a cutting pattern p_j is $1-\overline{y}A_j$. Then new patterns are generated by solving the integer knapsack problem:

$$\max \qquad \sum_{i=1}^{n} \overline{y}_i a_i \tag{5.13}$$

subject to

$$\sum_{i=1}^{n} s_i a_i \leq S \tag{5.14}$$

$$a_i \geq 0 \text{ for } i=1,..,n \tag{5.15}$$

$$a_i \text{ integer for } i=1,..,n \tag{5.16}$$

This problem is equivalent to a two-dimensional knapsack problem. The problem consists of determining a cutting pattern which maximizes the sum of the profits of the cut items. In constraint (5.14) s_i refers to the size of item *i* and *S* refers to the size capacity of the layer. This constraint describes that the packing of all items with size s_i and quantity of a_i into *S*. For example, if *S* is a rectangle size with length 25 and height 40, while s_{sm} is the unique smallest rectangle size with length 2 and width 4, solving the knapsack problem in the first cut gives a solution of 125 for a_{sm} to maximize the objective value. However, this is not feasible as we can only fit a maximum of 120 rectangles of s_{sm} in rectangle S. An intuitive bound for a_i is then taking the integer part of the ratio of dimensions of S and rectangles s_i, $\left\lfloor \frac{(L*W)}{(l_i*w_i)} \right\rfloor$ whereas another bound is by multiplying the maximum number of rectangles s_i in a horizontal row and the maximum number of rectangles s_i in a vertical column, $\max\left\{\lfloor L/l_i \rfloor * \lfloor W/w_i \rfloor, \lfloor W/l_i \rfloor * \lfloor L/w_i \rfloor\right\}$.

The column generation approach of solving a rectangular packing problem involves solving a dual problem of a two-dimensional knapsack problem. As it can be seen, constraint (5.14) is not straightforward to evaluate for different sizes of s_i due to the two-dimensionality. It is difficult to determine the feasibility of the 2D packing pattern using linear programming without verifying with a packing heuristic. Therefore, the column generation approach is not ideal for this problem. We are motivated to find some modification to the method in order to solve the problem. In the next section, an improvement heuristic is introduced below as an alternative method.

5.3.2 Layer Packing with Improvement Heuristic

In this section, a heuristic method of generating new and improved columns is proposed. The approach borrows from the idea of column generation for the rectangular bin packing problem. In Section 5.3.1, new columns can be created from solving the integer knapsack problem. However, in this section, new columns are created by solving the integer programming model and then using the information found to find improved columns for the next iteration. This is because solving the model gives enough information to pursue a better solution.

By solving the model above, we can obtain information on the following:

1. The objective value is a measure of the performance of the packing solution where a lower number of layers is desirable
2. Secondly, the results give the surplus of types from the difference of left and right hand sides of constraint (5.10) $\sum_{j=1}^{J} a_{ij} X_j - d_i$
3. Besides that, X_j shows which of the feasible packing patterns are used in the packing of the demand realization. Only the picked packing patterns are used for the next step of improvement.

There are several steps in solving the rectangle packing problem in this section. Firstly a random demand realization of items to be packed is generated. A packing algorithm such as the steplike stacking heuristic or the maximal rectangle packing is used to obtain initial feasible packing patterns as described in Section 5.3.2.1. Then the second step is solving the integer programming model for each demand realization in order to utilize information regarding the surplus of types. The last step is the generation of new columns by improving columns as described in Section 5.3.2.2 with an improvement method.

5.3.2.1 Initial Column

The initial column is generated based on the a rectangular packing algorithm such as steplike stacking heuristic as described in detail in the paper by (Shi & Xue, 2009). The heuristic finds feasible solutions for the rectangular packing problem of fixed container size with minimum waste as the objective. The packing is orthogonal and no overlap is allowed. The authors use the terms "Steplike Line" L_t to demarcate the boundary between packed and unpacked regions, and "Step Positions" P_j as the available corners to place items at the position. All step positions are corners closing to the left and bottom corner of the line L_t. At any time t, all combinations of step position and items i are evaluated using a score of "F_{ij}" to indicate closeness to match the step sizes and items with perfect matches are packed first, then followed by items with lower matches and ordered by the amount of area wastage induced. The heuristic is implemented using MATLAB R2012a.

The inputs are:

a. Bin/container length and width
b. Items to be packed in a matrix R with the columns of length l_i and width w_i. The first item to be packed is the item with the largest area. This is subsequently changed to picking the items randomly so as to induce more randomness.

At any time t, the following steps are performed:

a. The information on packed items are stored in a matrix Q where the first two columns are the packed length and width and the next two columns are the packed positions.

b. F_{ij} scores are assigned for each item i and step position j combination at time t. The adjacent distance D_{ij} which is the borders shared by L_t and item i at step position j is used to evaluate the score. W_{ij} is used to evaluate the total loss incurred when assigning an item that is larger than the size of the step position.

c. F_{ij} and W_{ij} are used to indicate the type of fit of the item and step position combination. There are 8 types of scenarios:

   - F_{ij}=2: Item fits on both length and width of P_j perfectly (W_{ij}=0)
   - F_{ij}=1: Item fits on either length or width and are smaller on both dimensions (W_{ij}=0)
   - F_{ij}=0: Item does not fit on either dimension and are smaller on both dimensions (W_{ij}=0)
   - F_{ij}=1: Item fits on the length of P_j perfectly and has larger width than the width of *Pj* (W_{ij} is nonzero)
   - F_{ij}=1: Item fits on the width of P_j perfectly and has larger length than the length of *Pj* (W_{ij} is nonzero)
   - F_{ij}=0: Item does not fit the size of P_j and has larger width than the width of *Pj* (W_{ij} is nonzero)
   - F_{ij}=0: Item does not fit the size of P_j and has larger length than the length of *Pj* (W_{ij} is nonzero)
   - F_{ij}=0: Item does not fit the size of P_j and has larger length than the length of *Pj* and larger width than the width of P_j (W_{ij} is nonzero)

d. The hierarchy of choosing the best combination of item-step position is based on the value of F_{ij}=2 and W_{ij}=0, then F_{ij}=1 and W_{ij}=0 and F_{ij}=0 and W_{ij}=0 followed by all combinations with nonzero loss $W_{ij.}$ For combinations that have the same value of F_{ij} and W_{ij}=0, the combination with highest value of D_{ij} is selected to be packed. For combinations with non-zero loss, the pairing with lowest value of W_{ij} is selected to be packed.

e. After assigning an item to a step position, P_j and L_t are updated depending on the type of scenario at the end of time *t*. Only one item is packed at each time *t*.
f. Packed item *i* is removed from matrix *R* and inserted into the matrix *Q*.
g. The heuristic terminates either when matrix *R* is empty, bin is full or no more items can be feasibly packed.

The outputs are:

a. Matrix *Q* with packed items and positions. The sequence of *Q* represents the sequence of packing of items into bin.
b. Utilization is computed by comparing total area of packed items with the bin area.

Initial column is generated by applying the heuristic to a demand realization of items. From the heuristic, feasible packing patterns are obtained. Patterns that have low utilization are discarded. Low utilization refers to utilizations which are lower than a pre-determined threshold value. This is because we only want columns with efficient packing utilization. All the remaining items in columns with low utilizations are regrouped as one demand and repacked again. This step provides more new columns for the initial column.

5.3.2.2 Generation of New Columns

The method is used to improve the current solution and also to get new columns for the master problem. The columns are improved using the information obtained from the master problem. Generally the idea is that we try to replace loose demand types with tight demand types to increase efficiency. This is a one to one exchange of types.

*i** refers to the row index *i* of element a_{ij} that makes the expression $[(a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{in}X_J) - d_i]$ the largest. This is finding the *i*th demand constraint that has the largest surplus of demand. Meanwhile *j** refers to the column index *j* of element a_{ij} that makes the term $a_{i^*j}X_j$ in *i**-th demand constraint the largest. This is finding the *j*-th pattern that contributes the largest to the surplus of item type *i**.

From constraint (5.10),

$$\begin{aligned}
&a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_J \geq d_1\\
&a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_J \geq d_2\\
&\vdots\\
&a_{n1}X_1 + a_{n2}X_2 + \cdots + a_{nn}X_J \geq d_n\\
&\sum_{j=1}^{J} a_{ij}X_j - d_i = (a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{in}X_J) - d_i \forall i \in [1,n]\\
&i^* = \arg\max_i [(a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{in}X_J) - d_i] \forall i \in [1,n]\\
&j^* = \arg\max_j [(a_{i^*1}X_1, a_{i^*2}X_2, ..., a_{i^*n}X_J)] \forall j \in [1,J]
\end{aligned}$$

1. Let $i^* = \arg\max_i (\sum_j a_{ij}X_j - d_i)$ and $j^* = \arg\max_j (a_{i^*j}X_j)$
2. Looking at pattern *j**, we try to improve the columns by replacing patterns which have loosest demand type with tightest demand types. The reason behind this is that tight demand types are tight because the options for a pattern with lower quantity are not available to choose from the available patterns. Not only that, the loose demand types are not as much required in the demand and so their space can be given up to substitute with tight

demand types. The actions reduce the total amount of surplus which means there is a closer match to the demand quantities required.

3. Let $i' = \arg\min_i (\sum_j a_{ij} X_j - d_i)$
4. Remove *i**, and put in as many *i*’ as possible. This is achieved by calculating the actual area occupied by *i** and replaced with *i*’.
5. Verify the new pattern by using a rectangular bin packing algorithm. This step is important to make sure the new pattern is physically feasible in a rectangular bin.
6. After verification, the master problem is re-solved with the new column.
7. If a tolerance of surplus is exceeded for any type *i*, repeat steps 1 to 6 to improve the columns. Otherwise, end.

5.4 Multiple Height Packing

Because the crates were pre-sorted into same heights for packing into layers in the previous section, the final stage of packing is packing the layers into containers or bins. The layers have a few types of heights and are packed into as few containers as possible to minimize the number of bins used. The problem is formulated and as shown below.

Inputs and Parameters

H	Container height
K	Total number of containers available
h_j	Height of layer j
J	Total number of distinct feasible cutting patterns p_j

Decision Variables

Y_k	Number of packed bins
e_{jk}	Number of layer type *j* in bin *k*

5.4.1 Problem formulation

Minimize $$\sum_{k=1}^{K} Y_k \qquad (5.17)$$

subject to

$$\sum_{j=1}^{J} h_j e_{jk} \le HY_k \text{ for } k=1,.., K \qquad (5.18)$$

$$\sum_{k=1}^{K} e_{jk} = 1 \text{ for } j=1,.., J \qquad (5.19)$$

$$e_{jk} \text{ integer for } j=1,.., J, k=1,.., K \qquad (5.20)$$

$$Y_k = \{0,1\} \text{ for } k=1,.., K \qquad (5.21)$$

In the objective function (5.17), we minimize the total number of bins. The term $\sum_k Y_k$ represents the sum of the number of bins *k* required. The variable e_{jk} represents the number of layer type *j* in bin *k*. The constraint (5.18) dictates that the sum of chosen layers of Y_k has to be less than or equal to the height *H* of the container. Meanwhile (5.19) assigns each layer to one bin only. The constraints (5.20) and (5.21), restricting the decision variable Y_k to be binary and e_{jk} to be integer.

5.5 Numerical Experiments

We evaluate the rectangular packing algorithm with improvement approach as described in the methodology section above by experimenting with several scenarios. The algorithm has been written in a test environment of MATLAB R2012a and IBM ILOG CPLEX v12.6 and tested on a Windows PC with specifications of 1.6GHz Pentium M processor and 512MB RAM.

5.5.1 Comparison to MIP

Both MIP in Section 5.2 and column generation approach described in Section 5.3 can be used to solve rectangular packing problems. In order to compare the two approaches to MIP, two test cases are set up. The first set is a set of 25 items with various heights and lengths. The second set is a set of 50 items. The test cases are from test cases J1 and J2 with known optimal solution from (Jakobs, 1996). All items should fit in the container dimensions for the optimal solution.

	MIP		Improvement Heuristic
Set	Utilization (%)	Time (CPU seconds)	Utilization (%)
1	100	2.16	100
2	100	4.69	98

Table 5.1 Comparison to MIP

5.5.2 Comparison to Maximal Rectangle Packing

For larger scale problems, the MIP will be harder to execute and therefore heuristics are used. In the improvement method, the initial column is obtained from the steplike stacking heuristic. Therefore we would like to know if changing the initial column method to the maximal rectangle heuristic will yield different results for the improvement approach described in Section 5.3. The following describes the comparison between the two as the starting heuristic for the improvement method. Table 5.2 shows the results obtained where the columns show improvement of average utilization of layers before and after performing column generation.

Instance	Average utilization before improvement	Average utilization after improvement
1	0.418575	0.794040
2	0.489320	0.775986
3	0.701556	0.775986
4	0.744508	0.748719
5	0.495216	0.765984
7	0.489320	0.813990
8	0.465739	0.812306
9	0.577752	0.803041
10	0.530588	0.747982

Table 5.2 Comparison of utilization before and after improvement

5.5.3 Varying Demand Profile

The method is tested against three types of demand profiles. The first demand profile is sampled from a uniform distribution of crate lengths with low variance, the second with medium variance and lastly, high variance. Next the demand profile is sampled from a set of crate widths that are low, medium and high in size. This affects the grouping of the demand into low, medium or high number of groups with the same packing height.

5.5.3.1 Steplike Stacking Algorithm with Improvement

CV	Max utilization (%)					Average
0.1	95.00	91.50	95.00	87.80	89.00	91.66
0.3	93.33	94.50	93.06	92.50	92.89	93.26
0.5	93.30	93.44	93.93	94.46	93.42	93.71

Table 5.3 Variance Level versus Packing Utilization Results I

Crate size	Max utilization (%)					Average
Small	92.00	90.08	90.61	91.00	84.80	89.70
Medium	90.33	90.29	91.03	87.45	89.50	89.72
Large	90.30	91.46	89.50	84.70	91.46	89.48

Table 5.4 Crate size versus Packing Utilization Results II

5.5.3.2 Maximal Rectangle Algorithm with Improvement

The demand from section 5.5.3.1 is repeated with the packing heuristic now changed to the maximal rectangle algorithm instead of the step like stacking heuristic. The following are the results from the change in heuristic. From the numerical results, it is shown that the improvement method can be independent of the starting heuristic.

CV	Max utilization (%)					Average
0.1	95.00	91.50	95.00	87.80	89.00	91.66
0.3	93.33	94.50	93.06	92.50	92.89	93.26
0.5	93.30	93.44	93.93	94.46	93.42	93.71

Table 5.5 Variance Level versus Packing Utilization Results II

Crate size	Max utilization (%)					Average
Small	92.00	90.08	90.61	91.00	84.80	89.70
Medium	90.33	90.29	91.03	87.45	89.50	89.72
Large	90.30	91.46	89.50	84.70	91.46	89.48

Table 5.6 Crate size versus Packing Utilization Results II

5.5.3.3 Multiple Height Packing

Lastly, multiple height packing is used to determine the number of containers needed from the packed layers used in the previous stage. Using MATLAB R2012a to construct the model for multiple height packing described in Section 5.4, the simulated scenarios involve random quantity of rectangles with shared height *h* for a few types of height and to be packed into containers. From the table below, *n* refers to the total quantity of layers with different heights of h to be packed into containers of a fixed size. The height of the container is set at 200.

Types of height	n	h	Number of containers
2	50	25	10
		50	
3	50	25	13
		50	
		75	
4	50	25	16
		50	
		75	
		100	
5	100	20	30
		40	
		60	
		80	
		100	

Table 5.7 Multiple height packing

5.5.3.4 Varying the number of crate types

In this section, we would like to investigate the effect of having different number of crate types on the containerization. Random sets of data with two, three, and four types are packed to minimize the number of layers.

Number of types	Set	Min utilization	Max utilization	Average utilization
2	1	0.624915	0.848942	0.774266
	2	0.106118	0.848942	0.663236
	3	0.011791	0.848942	0.639654
	4	0.312458	0.848942	0.670114
	5	0.607229	0.848942	0.768371
	6	0.760510	0.848942	0.819464
	7	0.483425	0.848942	0.727103
	8	0.560066	0.848942	0.752650
	9	0.530588	0.848942	0.742824
	10	0.725138	0.848942	0.807674

Table 5.8 Packing of two types

Table 8 shows the packing of ten random demands with two types of crates. For each set of demand, the minimum utilization, maximum utilization and average utilization are obtained. The minimum utilization is often quite low, because this reflects the odd crates left behind in the last container. On the other hand, the maximum utilization appears stagnant at 0.848942 due to the reason that this packing pattern is the best and is used for all containers before coming to the odd crates which are left behind. The average utilization varies between 0.63654 and 0.819464.

Number of types	Set	Min utilization	Max utilization	Average utilization
3	1	0.489320	0.848942	0.758299
	2	0.660288	0.848942	0.801778
	3	0.795883	0.848942	0.835677
	4	0.253503	0.848942	0.728675
	5	0.053059	0.848942	0.689765
	6	0.621968	0.848942	0.792198
	7	0.792935	0.848942	0.834940
	8	0.784092	0.848942	0.832729
	9	0.288876	0.848942	0.708925
	10	0.837151	0.848942	0.845994

Table 5.9 Packing of three types

Table 5.9 shows the packing of ten random demands with three types of crates. Similar to the experiment conducted for two types, the results of the average utilization appears to be slightly better off in general with the average utilization ranging from 0.689765 to 0.845994. The maximum utilization remains at 0.848942 because this is the best packing pattern available to satisfy for the earlier containers.

Number of types	Set	Min utilization	Max utilization	Average utilization
4	1	0.212235	0.848942	0.741350
	2	0.347830	0.848942	0.764932
	3	0.100222	0.848942	0.724155
	4	0.165072	0.848942	0.734472
	5	0.760510	0.848942	0.830666
	6	0.271190	0.848942	0.751176
	7	0.394994	0.848942	0.772792
	8	0.807673	0.848942	0.840098
	9	0.598386	0.848942	0.798830
	10	0.035373	0.848942	0.711382

Table 5.10 Packing of four types

Finally Table 5.10 shows the packing of ten random demands with three types of crates. Similar to the previous two experiments, the results show that the average utilization ranges from 0.711382 to 0.840098. The maximum utilization also remains at 0.848942.

6 Conclusions and Future Research

The study of this thesis is on the problem of sizing of crates, with inventory consideration in addition to the packing of the crates into containers. The objective is to minimize total cost while addressing the issue of the optimal number of types of crates to use and the optimal sizes respectively. Besides that, the packing of the crates into containers is also considered. The study is important because it is based on a real industrial problem and there are practical results which can be applied to improve the various aspects of the problem.

6.1 Conclusions

Firstly, we are able to define and formalize an actual industrial problem where an MIP is formulated for the crate length optimization problem to minimize total loss of length and determine the optimal crate lengths. In the crate length optimization problem, historical data was used to find the optimal number of crate lengths given the number of crate types.

Next, we extend the problem to determine both the number of optimal crate types to use and also the optimal sizes using inventory consideration. Here we consider inventory and introduce safety stock into the problem. The problem is formulated as a non-linear MIP; however it has a good property which makes it suitable to solve efficiently using dynamic programming. A dynamic program is formulated for the problem which is able to determine both the number of crate types and optimal sizes at the same time.

A generalized crate sizing problem is then formulated to find optimal crate sizes in 3D and solved using Hungarian-based GA algorithm. As the width and the height are the same, the problem can be modelled in 2D. Using the Hungarian match for parent selection and crossover, the neighbourhood property can be preserved and the GA is used to find the crate sizes. The Hungarian match is needed due to the structure of the problem as the crate sizes are more likely to belong to one of its neighbours than a size that is far in distance.

Finally, we also consider the problem of packing the crates into containers using an improved bin packing algorithm. The actual 3D bin packing problem has been reduced to a 2D packing problem due to several properties of the problem. Moreover, packing by layer is more intuitive and easier to apply. When packing multiple different size items, they are pre-sorted into items of same height which then enables the problem to be solved using a rectangular packing algorithm. The improvement method uses readily available rectangular packing heuristics to generate the initial column. Then, new and improved columns are constructed from the information of the previous iteration.

6.2 Future Research Topics

There are several topics related to the scope of this thesis where future research can be conducted.

In the crate sizing problem, the GA was constructed to find the optimal crate length, width and height. The model is based on 2D because the problem has the property of rectangular cross section. However, a future research topic can be an extension of the problem to 3D. By varying the number of crate types for each run, it is also possible to find the optimal number of types to use. Instead, a future research topic can be variable chromosome length GA. In traditional GA, the chromosome length is determined when the solution is encoded into a chromosome. Subsequently, the chromosome length does not change. Varying the chromosome length allows for finding an optimal solution by starting with a shorter chromosome and is then transferring to the following stages with a longer chromosome to maintain diversity.

Additionally, the crate sizing problem and the bin packing problem are related. The crate sizes to pack the rolls are obtained from dynamic programming or GA and these crates are then packed into the containers. The crate sizes obtained from the earlier stage will influence the packing problem in the next stage. A potential research topic is to treat both problems together and investigate how the crate sizes affect the packing stage afterwards and use this information to improve the total cost of both stages. The problem can also be extended with other considerations such as rotations, weight or symmetry constraints.

In our problem, only one roll is packed into a crate. It would be interesting to study a different problem if multiple products are allowed. In this case, larger crate sizes may be more desirable and will not be penalized as much because it is able to contain more products in one crate.

References

Abidi, S., Krichen, S., Alba, E., & Molina, J. M. (2013, 28-30 Apr.*).* *Improvement heuristic for solving the one-dimensional bin-packing problem*. Paper presented at the 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), 2013. (pp. 1-5). IEEE.

Adelson, R. M., Norman, J. M., & Laporte, G. (1976). A dynamic programming formulation with diverse applications. *Operational Research Quarterly (1970-1977), 27*(1), 119-121. doi: 10.2307/3009216

Akeda, Y., & Hori, M. (1976). On random sequential packing in two and three dimensions. *Biometrika, 63*(2), 361-366. doi: 10.2307/2335631

Alves, C., & Carvalho, J. M. V. d. (2008). New integer programming formulations and an exact algorithm for the ordered cutting stock problem. *The Journal of the Operational Research Society, 59*(11), 1520-1531. doi: 10.2307/20202235

Anika, & Garg, D. (2014, 21-22 Feb.). *Parallelizing generalized one-dimensional bin packing problem using MapReduce*. Paper presented at the IEEE International Advance Computing Conference (IACC), 2014. (pp. 628-635). IEEE.

Baker, B. M. (1999). A spreadsheet modelling approach to the assortment problem. *European Journal of Operational Research, 114*(1), 83-92. doi: http://dx.doi.org/10.1016/S0377-2217(98)00097-6

Bansal, N., Lodi, A., & Sviridenko, M. (2005, 23-25 Oct.). *A tale of two dimensional bin packing*. Paper presented at the Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005. (pp. 657-666). IEEE.

Bekrar, A., & Kacem, I. (2008, 30 Jun.-2 Jul.). *A comparison study of heuristics for solving the 2D guillotine strip and bin packing problems*. Paper presented at the International Conference on Service Systems and Service Management, 2008. (pp. 1-6). IEEE.

Berkey, J. O., & Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. *The Journal of the Operational Research Society, 38*(5), 423-429. doi: 10.2307/2582731

Berkey, J. O., & Wang, P. Y. (1991, 30 Apr.-2 May). *A parallel approximation algorithm for solving one-dimensional bin packing problems*. Paper presented at the Proceedings of Fifth International Parallel Processing Symposium, 1991. (pp. 138-143). IEEE.

Bhatia, A. K., Hazra, M., & Basu, S. K. (2009, 6-7 Mar.). *Better-fit heuristic for one-dimensional bin-packing problem*. Paper presented at the IEEE International Advance Computing Conference IACC, 2009. (pp. 193-196). IEEE.

Bongers, C. (1980). *Standardization: Mathematical methods in assortment determination*. Dordrecht, Netherlands: Springer Netherlands.

Bongers, C. (1982). Optimal size selection in standardization: A case study. *The Journal of the Operational Research Society, 33*(9), 793-799. doi: 10.2307/2581209

Brusco, M. J., Thompson, G. M., & Jacobs, L. W. (1997). A morph-based simulated annealing heuristic for a modified bin-packing problem. *The Journal of the Operational Research Society, 48*(4), 433-439. doi: 10.2307/3010270

Cao, D. & Kotov, V. M. (2011, 12-14 Aug.). *A best-fit heuristic algorithm for two-dimensional bin packing problem*. Paper presented at the International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011. (Vol. 7, pp. 3789-3791). IEEE.

Caprara, A., Lodi, A., & Monaci, M. (2005). Fast approximation schemes for two-stage, two-dimensional bin packing. *Mathematics of Operations Research, 30*(1), 150-172. doi: 10.2307/25151644

Chen, M.-C., & Lin, C.-P. (2007). A data mining approach to product assortment and shelf space allocation. *Expert Systems with Applications, 32*(4), 976-986. doi: http://dx.doi.org/10.1016/j.eswa.2006.02.001

Coverdale, I., & Wharton, F. (1978). On cutting stock problems. *The Journal of the Operational Research Society, 29*(5), 503-504. doi: 10.2307/3009773

Cui, Y., & Zhou, R. (2002). Generating optimal cutting patterns for rectangular blanks of a single size. *The Journal of the Operational Research Society, 53*(12), 1338-1346. doi: 10.2307/822723

Dowsland, K. A. (1996). Genetic algorithms-A tool for OR? *The Journal of the Operational Research Society, 47*(4), 550-561. doi: 10.2307/3010730

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research, 29*(6), 1092-1104. doi: 10.2307/170363

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research, 44*(2), 145-159. doi: http://dx.doi.org/10.1016/0377-2217(90)90350-K

Flapper, S. D. P., González–Velarde, J. L., Smith, N. R., & Escobar-Saldívar, L. J. (2010). On the optimal product assortment: Comparing product and customer based strategies. *International Journal of Production Economics, 125*(1), 167-172. doi: http://dx.doi.org/10.1016/j.ijpe.2010.01.017

Gasimov, R. N., Sipahioglu, A., & Saraç, T. (2007). A multi-objective programming approach to 1.5-dimensional assortment problem. *European Journal of Operational Research, 179*(1), 64-79. doi: http://dx.doi.org/10.1016/j.ejor.2006.03.016

Gemmill, D. D. (1992). Solution to the assortment problem via the genetic algorithm. *Mathematical and Computer Modelling, 16*(1), 89-94. doi: http://dx.doi.org/10.1016/0895-7177(92)90080-5

George, J. A. (1996). Multiple container packing: A case study of pipe packing. *The Journal of the Operational Research Society, 47*(9), 1098-1109. doi: 10.2307/3010370

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research, 9*(6), 849-859. doi: 10.2307/167051

Gilmore, P. C., & Gomory, R. E. (1963). A linear programming approach to the cutting stock problem-Part II. *Operations Research, 11*(6), 863-888. doi: 10.2307/167827

Gilmore, P. C., & Gomory, R. E. (1965). Multistage cutting stock problems of 2 and more dimensions. *Operations Research, 13*(1), 94-120. doi: http://dx.doi.org/10.1287/opre.13.1.94

Gómez, A., & Fuente, D. d. l. (2000). Resolution of strip-packing problems with genetic algorithms. *The Journal of the Operational Research Society, 51*(11), 1289-1295. doi: 10.2307/254213

Haessler, R. W., & Sweeney, P. E. (1991). Cutting stock problems and solution procedures. *European Journal of Operational Research, 54*(2), 141-150.

Healy, P., & Moll, R. (1996). A local optimization-based solution to the rectangle layout problem. *The Journal of the Operational Research Society, 47*(4), 523-537. doi: 10.2307/3010728

Hifi, M., Kacem, I., Nègre, S., & Wu, L. (2010). A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics, 36*(0), 993-1000. doi: http://dx.doi.org/10.1016/j.endm.2010.05.126

Hinxman, A. I. (1980). The trim-loss and assortment problems: A survey. *European Journal of Operational Research, 5*(1), 8-18. doi: http://dx.doi.org/10.1016/0377-2217(80)90068-5

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research, 88*(1), 165-181. doi: http://dx.doi.org/10.1016/0377-2217(94)00166-9

Ji, J., & Jeng, M. (1990, 9-13 Dec.). *Bin-packing adjustable rectangles and applications to task scheduling on partitionable parallel computers*. Paper presented at the Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, 1990. (pp. 312-315). IEEE.

Jiang, J., & Cao, L. (2012, 19-20 May). *A hybrid simulated annealing algorithm for three-dimensional multi-bin packing problems*. Paper presented at the International Conference on Systems and Informatics (ICSAI), 2012. (pp. 1078-1082). IEEE.

Kao, C.-Y., & Lin, F.-T. (1992, 18-21 Oct.). *A stochastic approach for the one-dimensional bin-packing problems*. Paper presented at the IEEE International Conference on Systems, Man and Cybernetics, 1992. (pp. 1545-1551). IEEE.

Kasimbeyli, N., Sarac, T., & Kasimbeyli, R. (2011). A two-objective mathematical model without cutting patterns for one-dimensional assortment problems. *Journal of Computational and Applied Mathematics, 235*(16), 4663-4674. doi: http://dx.doi.org/10.1016/j.cam.2010.07.019

Korchemkin, M. B. (1983). A heuristic partitioning algorithm for a packaging problem. *Computing, 31*(3), 203-209. doi: 10.1007/BF02263431

Leung, S. Y. S., Wong, W. K., & Mok, P. Y. (2008). Multiple-objective genetic optimization of the spatial design for packing and distribution carton boxes. *Computers & Industrial Engineering, 54*(4), 889-902. doi: http://dx.doi.org/10.1016/j.cie.2007.10.018

Levine, J., & Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *The Journal of the Operational Research Society, 55*(7), 705-716. doi: 10.2307/4102017

Li, H.-L., & Chang, C.-T. (1998). An approximately global optimization method for assortment problems. *European Journal of Operational Research, 105*(3), 604-612. doi: http://dx.doi.org/10.1016/S0377-2217(97)00072-6

Li, H.-L., Chang, C.-T., & Tsai, J.-F. (2002). Approximately global optimization for assortment problems using piecewise linearization techniques. *European Journal of Operational Research, 140*(3), 584-589. doi: http://dx.doi.org/10.1016/S0377-2217(01)00194-1

Li, H.-L., & Tsai, J.-F. (2001). A fast algorithm for assortment optimization problems. *Computers & Operations Research, 28*(12), 1245-1252. doi: http://dx.doi.org/10.1016/S0305-0548(00)00035-6

Lin, C.-C. (2006). A genetic algorithm for solving the two-dimensional assortment problem. *Computers & Industrial Engineering, 50*(1–2), 175-184. doi: http://dx.doi.org/10.1016/j.cie.2006.03.002

Lin, J. -L., Foote, B., Pulat, S., Chang, C. –H., & Cheung, J. Y. (1993, 1-5 Mar.). *Hybrid genetic algorithm for container packing in three dimensions*. Paper presented at the Proceedings of the Ninth Conference on Artificial Intelligence for Applications, 1993.

Lins, L., Lins, S., & Morabito, R. (2003). An L-approach for packing (ℓ, w)-rectangles into rectangular and L-shaped pieces. *The Journal of the Operational Research Society*, *54*(7), 777-789. doi: 10.2307/4101727

Liu, F.-H. F., & Hsiao, C. -J. (1997). A three-dimensional pallet loading method for single-size boxes. *The Journal of the Operational Research Society, 48*(7), 726-735. doi: 10.2307/3010061

Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research, 48*(2), 256-267. doi: 10.2307/223143

Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management Science, 44*(3), 388-399. doi: 10.2307/2634676

Mrad, M., Meftahi, I., & Haouari, M. (2013). A branch-and-price algorithm for the two-stage guillotine cutting stock problem. *The Journal of the Operational Research Society, 64*(5), 629-637. doi: 10.2307/23407008

Oliveira, J. F., & Wäscher, G. (2007). Cutting and packing. *European Journal of Operational Research, 183*(3), 1106-1108. doi:10.1016/j.ejor.2006.04.022

Omar, M. K., & Ramakrishnan, K. (2011, 6-9 Dec.). *EPSO for solving non-oriented two-dimensional bin packing problem.* Paper presented at the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2011. (pp. 106-110). IEEE.

Ong, H. L., Magazine, M. J., & Wee, T. S. (1984). Probabilistic analysis of bin packing heuristics. *Operations Research, 32*(5), 983-998. doi: 10.2307/170649

Pargas, R. P., & Jain, R. (1993, 1-5 Mar.). *A parallel stochastic optimization algorithm for solving 2D bin packing problems*. Paper presented at the Proceedings of the Ninth Conference on Artificial Intelligence for Applications, 1993. (pp. 18-25). IEEE.

Pentico, D. W. (1986). Comments on "On the optimal choice of sizes" by Peter Tryfos. *Operations Research, 34*(2), 328-329. doi: 10.2307/170829

Pentico, D. W. (2008). The assortment problem: A survey. *European Journal of Operational Research, 190*(2), 295-309. doi: http://dx.doi.org/10.1016/j.ejor.2007.07.008

Pimpawat, C., & Chaiyaratana, N. (2001, 7-30 May). *Using a co-operative co-evolutionary genetic algorithm to solve a three-dimensional container loading problem.* Paper presented at the Proceedings of Congress on Evolutionary Computation, 2001. (Vol. 2, pp. 1197-1204). IEEE.

Puchinger, J., & Raidl, G. R. (2007). Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational*

Research, 183(3), 1304-1327. doi: http://dx.doi.org/10.1016/j.ejor.2005.11.064

Rajaram, K. (2001). Assortment planning in fashion retailing: methodology, application and analysis. *European Journal of Operational Research, 129*(1), 186-208. doi: http://dx.doi.org/10.1016/S0377-2217(99)00406-3

Rhee, W. T., & Talagrand, M. (1991). Multidimensional optimal bin packing with items of random size. *Mathematics of Operations Research, 16*(3), 490-503. doi: 10.2307/3690035

Rhee, W. T., & Talagrand, M. (1993). On line bin packing with items of random size. *Mathematics of Operations Research, 18*(2), 438-445. doi: 10.2307/3690289

Salma, M., & Ahmed, F. (2011, May 31-Jun. 3). *Three-dimensional bin packing problem with variable bin length application in industrial storage problem.* Paper presented at the 4th International Conference on Logistics (LOGISTIQUA), 2011. (pp. 508-513). IEEE.

Savelsbergh, M. (1997). A branch-and-price algorithm for the generalized assignment problem. *Operations research, 45*(6), 831-841. doi: 10.2307/172068

Shi, W., & Xue, Z. (2009, 19-20 Dec.). *A steplike stacking heuristic algorithm for solving rectangle packing problem.* Paper presented at the International Conference on Information Engineering and Computer Science (ICIECS), 2009. (pp. 1-4). IEEE.

Sinuany-Stern, Z., & Weiner, I. (1994). The one dimensional cutting stock problem using two objectives. *The Journal of the Operational Research Society*, *45*(2), 231-236. doi: 10.2307/2584129

Toledo Suarez, C. D., Gonzlez, E. P., & Rendon, M. V. (2006, 13-17 Nov.). *A heuristic algorithm for the offline one-dimensional bin packing problem inspired by the point Jacobi matrix iterative method.* Paper presented at the Fifth Mexican International Conference on Artificial Intelligence (MICAI), 2006. (pp. 281-286). IEEE.

Tryfos, P. (1985). Technical note—On the optimal choice of sizes. *Operations Research*, *33*(3), 678-684. doi: 10.2307/170565

Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, *3*(2), 111-130. doi: 10.1007/bf01300970

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming, 86*(3), 565-594. doi: 10.1007/s101070050105

Vanderbeck, F. (2000). Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, *48*(6), 915-926. doi: 10.2307/222998

Vanderbeck, F. (2001). A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. *Management Science*, *47*(6), 864-879.

doi: 10.2307/2661644

Van De Vel, H., & Shijie, S. (1991). An application of the bin-packing technique to job scheduling on uniform processors. *The Journal of the Operational Research Society*, *42*(2), 169-172. doi: 10.2307/2583183

Verma, V., & Singh, B. (2010). Genetic-algorithm-based design of passive filters for offshore applications. *Industry Applications, IEEE Transactions on Industry Applications*, *46*(4), 1295-1303. doi: 10.1109/TIA.2010.2049629

Vidal, R. V. V. (1994). On the optimal sizing problem. *The Journal of the Operational Research Society*, *45*(6), 714-719. doi: 10.2307/2584462

Wang, H., & Chen, Y. (2010, 23-26 Sept.). *A hybrid genetic algorithm for 3D bin packing problems*. Paper presented at the IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010. (pp. 703-707).

Wang, L., Wang, D., Ni, H., & Cheng, J. (2011, 15-17 Apr.). *On the genetic-search-algorithm-based multi-parameter optimization design system of packaging container size*. Paper presented at the International Conference on Electric Information and Control Engineering (ICEICE), 2011. (pp. 3399-3402). IEEE.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*(3), 1109-1130. doi: http://dx.doi.org/10.1016/j.ejor.2005.12.047

Wilson, R. C. (1965). A packaging problem. *Management Science, 12*(4), B135-B145. doi: 10.2307/2627807

Wong, W. K., & Leung, S. Y. S. (2006, 21-23 June). *Carton box optimization problem of VMI-based apparel supply chain.* Paper presented at the IEEE International Conference on Management of Innovation and Technology, 2006. (Vol. 2, pp. 911-915). IEEE.

Xu, J., Qin, H., Shen, R., & Shen, C. (2008, 12-15 Oct.). *An optimization framework for the box sizing problem*. Paper presented at the IEEE International Conference on Service Operations and Logistics, and Informatics (IEEE/SOLI), 2008. (Vol. 2, pp. 2872-2877). IEEE.

Yanasse, H. H. (1994). A search strategy for the one-size assortment problem. *European Journal of Operational Research, 74*(1), 135-142. doi: http://dx.doi.org/10.1016/0377-2217(94)90211-9

Yang, H., & Shi, J. (2010, 22-24 Jan.). *A hybrid CD/VND algorithm for three-dimensional bin packing*. Paper presented at the Second International Conference on Computer Modeling and Simulation (ICCMS), 2010. (Vol. 3, pp. 430-434). IEEE.

Zhang, X., Yuan, Y., & Yuan, X. (2012, 25-27 July). *Correlation between average waste space and box size in online next-fit bin-packing.* Paper presented at the 31st Chinese Control Conference (CCC), 2012. (pp. 2498-2502). IEEE.