8-2013

Three-Dimensional Knapsack Problem with Pre-Placed Boxes and Vertical Stability

Hanan Mostaghimi Ghomi
Universty of Windsor

Three-Dimensional Knapsack Problem with Pre-Placed Boxes and Vertical Stability

By

Hanan Mostaghimi Ghomi

A Thesis
Submitted to the Faculty of Graduate Studies
through Industrial and Manufacturing Systems Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2013

Three-Dimensional Knapsack Problem with Pre-Placed Boxes and Vertical Stability

by

Hanan Mostaghimi Ghomi

APPROVED BY:

_____

Dr. I. Ahmad
Computer Science School

_____

Dr. R. Lashkari
Industrial & Manufacturing Systems Engineering

_____

Dr. W. Abdul-Kader, Advisor
Industrial & Manufacturing Systems Engineering

10 June 2013

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

A three-dimensional knapsack problem packs a subset of rectangular boxes inside a bin with fixed size such that the total value of packed boxes is maximized. Each box has its own value and size and can be freely rotated into any of the six positions while its edges are parallel to the bin's edges. A Mixed Integer Linear Programming is developed for the 3D knapsack problem, while some practical constraints such as vertical stability are considered. However, the given model can be applied to two dimensional problems as well. The proposed solution methodology is based on the sequence triple. Simulated annealing technique is used to model the heuristic approach. Moreover, the situation where some boxes are pre-placed in the bin is investigated. These pre-placed boxes represent potential obstacles. Numerical experiments are conducted for bins with and without obstacles. The results show that the heuristic approach is successful and can handle different kinds of instances.

DEDICATION

I dedicated this thesis

To my beloved parents,

the best people I know.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. I am thankful to my supervisor, Dr. W. Abdul-Kader, whose advice and knowledge added to my graduate experience. His insight has inspired me, and I would appreciate his support.

I would like to appreciate the outside reader, Dr. Imran Ahmad, for his great advice. I am grateful to Dr. R. Lashkari for his helpful suggestions which improved the quality of my thesis. I would also like to appreciate Dr. J. Urbanic for chairing the thesis defence.

Furthermore, a special thanks goes to my dear friend, Omid Beiraghi, who kindly corrected my writing. I am especially grateful to my beloved parents who encouraged me to come for the Master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS/SYMBOLS

C&P	Cutting and Packing
MILP	Mixed Integer Linear Programming
3D	Three-Dimensional
2D	Two-Dimensional
ILP	Integer Linear Programming

CHAPTER 1
Introduction

1.1.Background

Cutting and packing problems have been intensely studied as they have many applications in industrial and finance management. The three dimensional packing problem is essential for practical purposes such as container loading or scheduling which can be defined as a geometric assignment problem. The various packing problems can have different constraints and objectives. For instance, in the case of shipping, objects with different sizes have to be packed into a larger container. A topology of packing problems in general was defined by Dyckhoff et al. (1990) and a recent survey was defined by Wascher et al. (2007). Cutting and packing problems appear under several different names such as bin packing, multi-container loading problem, strip packing and knapsack problems, based on the objective function and the side constraints. All types of cutting and packing problems have some similar structures. They consist of two sets of elements, a set of large objects (called bins) and a set of small items (called boxes). The problem is to select some or all small items and assign them to one of the large objects while all selected small items are placed entirely in the large object and do not overlap and a given objective function is optimized. Thus, only some of the large objects and small items may be used in a solution of the problem. The packing problem considers optimal utilization of bin volume for goods distribution and is an important industrial problem. Filling a bin optimally decreases the shipping cost and increases the stability of the load. The large objects, which are called bins, can be homogeneous or heterogeneous. If the boxes placed in the given bin are identical it is called homogeneous; however, if various types of boxes are placed in it, it is considered as strongly heterogeneous.

Different kinds of cutting and packing problems can be divided to two categories. In the first category, sufficient bins are available to pack all the boxes; however, only a limited number of bins is available to pack a subset of boxes in the second category. The first type of problems are called an input minimization problem, and the second type are called an output maximization type. In the case of output maximization, a set of boxes has to be packed in a set of bins where the number of bins is not enough.

However, in the case of input minimization, all the boxes can be packed. In *strip packing* problem, a set of rectangular boxes are packed in a strip with certain width and height and variable length. The problem is how to place all the boxes inside the strip such that its length is minimized. In *bin-packing* problem, a set of items have to be packed in a set of bins of the same fixed sizes and costs, such that the number of used bins is minimized. Unlike bin-packing problem, in *multi-container loading* problem, the containers (or bins) do not essentially have equal sizes and costs. In *knapsack problem* each item has a profit and the problem is to choose the best subset of items that fits into the single bin or container such that the sum of the items profit is maximized. In this kind of problem, the availability of bins is limited so all items cannot be packed. (Leung, 2012; Fekete & Schepers 1997; Wei et al. 2009; Egeblad et al. 2010; Pisinger 2002).

1.2. Knapsack Problem

The knapsack problem is a problem in combinatorial optimization. The multidimensional knapsack Problem (MKP) is a strongly NP-hard optimization problem which can be show by reduction from the one-dimensional packing problem; it means that it is very unlikely to develop polynomial algorithms for these problems. Knapsack problems consist of three different types. The first one is *Single Knapsack Problem* (SKP), the problem of packing a subset of strongly heterogeneous boxes in a single container. *Multiple Identical Knapsack Problem* is the second type which considers packing a subset of strongly heterogeneous boxes in a set of identical bins. The last type is *Multiple Heterogeneous Knapsack Problem* (MHKP) which is the problem of packing a subset of strongly heterogeneous boxes in a set of weakly and strongly heterogeneous bins. Figure 1.1 shows the different types of knapsack problems in summary.

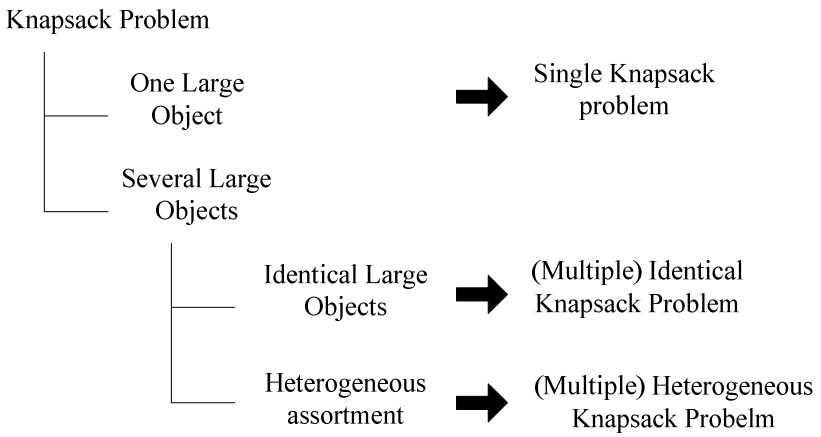


Figure 1.1 Knapsack Problem Types, Wascher et al. (2007)

Various practical constraints can be considered in the multidimensional knapsack problems. Some of these constraints are related to the bin, while some of them may refer to the boxes. Moreover, some constraints might be related to the relationship between the bin and boxes. One such constraint is the orientation constraint. Principally, each box dimension can be considered as height, thus three other orientations can easily be defined. Each box can have six orientations in order to orthogonally be placed in a bin. Moreover, one other practical constraint is the positioning constraint which limits the location of the boxes in the bin.

Load stability constraint is one of the most important issues in knapsack problems. In spite of its importance, load stability is often not studied explicitly in the literature. The stability is a direct consequence of load trimness when high bin utilization can be assured. This is typically true for knapsack problems in which only a subset of boxes can be packed as the bin availability is limited. Load stability can be divided into vertical and horizontal stability. Vertical stability prevents boxes from falling down onto bin floor or on top of other boxes. It deals with gravity force. In order to satisfy this kind of stability, the bottom of a box should be supported by the bin floor or other box tops. Horizontal stability or dynamic stability guarantees that boxes cannot shift notably when the bin is moving. Horizontal stability is satisfied when each packed box is adjacent to other boxes or to the bin wall.

In addition, another constraint which can be considered in knapsack problems is the guillotine cutting constraint. A packing is guillotineable if it is able to be reached

by a series of cuts which are in parallel to the bin walls. Guillotineable patterns are not always suitable for packing as the boxes tend to be more unstable while being transported. A robot packable packing is one which can be done by placing boxes starting from left-bottom-behind corner of a bin, while each box is placed in front, on the right or above the already packed boxes. Robot packable packing tackles a situation in which a robot with artificial hands packs the boxes into the bin.

Although technological knowledge has enhanced, solving real knapsack problems is still a challenge. The solution quality and computational efficiency are very sensitive to the box-positioning rule. Due to NP-hardness of the packing problem, only few exact algorithms and many heuristic methods have been presented which are based on the different strategies (Leung, 2012; Fekete & Schepers, 1997; Wei et al., 2009; Egeblad et al., 2010; Pisinger, 2002; Bortfeldt & Wascher, 2012).

The problem addressed here, in the topology suggested by Dyckhoff (1990), belongs to 3/B/O/F (3: three-dimensional, B/O: one object/bin and items selection, F: few items of different types) while Wascher et al. (2007) classify it as the three-dimensional single orthogonal knapsack problem. As well as non-overlapping constraints, some other constraints should be considered in practice, such as bin stability and pre-placed boxes. The given problem considers the packing of rectangular items in a rectangular bin in order to maximize the total value of the packed items (minimize the amount of space loss).The value of boxes is assumed to be equal to their volume. The rotation of the boxes is taken into account as well. Since the three-dimensional knapsack problem is NP-hard, it is difficult to solve. In addition, the difficulty of finding optimal solution is enhanced as the box rotations increase the search space significantly. Some exact algorithms as well as heuristic methods are proposed in the published literature. Since exact algorithms need more time to find a solution, heuristic approaches are more popular and can be used as an alternative to find near optimal solutions. A mixed integer linear model is developed for the given knapsack problem. The model considers vertical stability and pre-placed constraints which were not studied in Egeblad and Pisinger (2009). These practical constraints as well as the box rotations are added to the model in order to study a realistic knapsack problem. The proposed three-dimensional solution methodology is based on the sequence triple representation proposed by Egeblad and Pisinger (2009).

The developed algorithm also considers box rotation, pre-placed boxes and vertical stability. Simulated annealing is used as a heuristic method.

1.3. Simulated Annealing

Simulated annealing (SA) is a general optimization method to solve combinatorial optimization problems. It belongs to the class of local search algorithms. Simulated annealing algorithm has been used to handle many NP-hard problems. It was developed in 1983 to solve nonlinear problems. The inspiration comes from annealing in metallurgy, a technique of heating and controlled cooling of material in order to enhance the size of its crystal and decrease their defects, so that its structure is finally frozen which occurs at a minimum energy configuration. Simulated annealing algorithm is based on the very important fact that even in low temperature it is probable to have a particle with high internal energy. This fact shows the possibility of jumping out of the local minimum. While the temperature is reduced, the possibility of jumping out decreases. The basic elements of simulated annealing are as follows:

1. A finite set S.
2. A cost function which is defined on S.
3. A set $S(i) \subset S - \{i\} \forall i \in S$ which is the set of the neighbours of i.
4. Cooling schedule T which is a non-increasing function. T(t) is the temperature at time t.
5. An initial state.

The slow cooling is applied to the simulated annealing method as a slow reduction in the probability of accepting worse solutions. At each step, the algorithm considers some neighbouring states of the current state, and decides whether to stay at the current state or move to a neighbouring state. The probability of moving from a current state to a new neighbouring state is called acceptance probability which depends on the energies of the two states and a control parameter known as temperature. If the energy of the new state is better than the current one, the acceptance probability is equal to one. However, when the energy of the new state is

worse, the move to the new state is accepted if $e^{(-\Delta/temperature)} > R$, where $\Delta = \frac{(current\_state\_energy - new\_state\_energy)}{current\_state\_energy}$, and R=Uniform(0,1). At first, T has a relatively high value, so the chance to accept the new state is higher. T is slowly decreased to values such that most new states will not be accepted. The algorithm is repeated until it achieves a state that is good enough for the given application or until a given computation time is exhausted. It has been proved that by controlling cooling rate of temperature this algorithm can find the global optimum, although it needs infinite time. Like all other algorithms, simulated annealing has some strengths and weaknesses. It can deal with chaotic data, highly nonlinear problems and many constraints. It is able to reach global optimality. Simulated annealing algorithm is relatively flexible as it does not depend on any restrictive model's properties. However, as SA is a metaheuristic algorithm, so many choices are required to consider in the actual algorithm. Obviously, there is a trade-off between the quality of the solutions and computation time. Figure 1.1 shows the block diagram of simulated annealing (Bertsimas & Tsitsiklis, 1993; Dowhan et al., 2009).

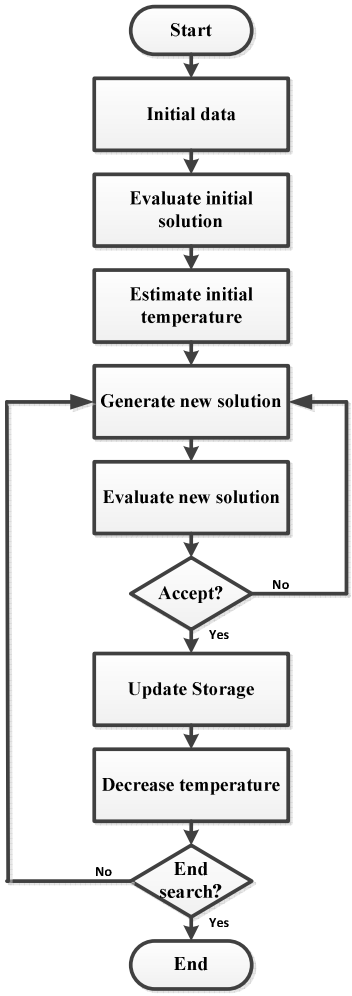


Figure 1.2. Simulated Annealing Block Diagram (Dowhan et al., 2009)

CHAPTER 2

Literature Review

2.1. Two Dimensional Knapsack Problem

Some papers in this area focus on two-dimensional packing problem. Leung et al. (2001) present a genetic algorithm and a simulated annealing approach to solve the two-dimensional non-guillotine cutting stock problem. They aim to find a cutting pattern which minimizes trim loss. The authors apply the genetic algorithm and simulated annealing to determine the permutations of small trim loss; then they use different packing approaches to pack the items corresponding to a special permutation. The proposed heuristic cannot produce all the feasible packings.

Capara and Monaci (2004) consider upper bounds and exact algorithm for the two-dimensional orthogonal knapsack problem. The authors present an approximation algorithm and four exact algorithms based on the enumeration scheme, and mainly focus on upper bounds. They claim their algorithm has similar performance to Fekete and Schepers' (1997) algorithms in most instances.

Clautiaux et al. (2007) consider the two-dimensional orthogonal knapsack problem and propose two exact methods to solve the problem. In the first algorithm, they improve the classic branch and bound method; however, the second one is on the basis of a new relaxation of the problem. They, moreover, define the reduction procedures and lower bounds used within both enumerative methods. The first algorithm is called LMAO (Leftmost Active Only) which counts the packing of items only in the left-most-downward position and tests the possibility of not packing any item in that position. By using this algorithm the same packing is not counted twice. The second algorithm called Two Step Branching Procedure (TSBP) is based on cutting each item with w_i and height h_i into h_i strips with width w_i. All strips relating to the given item must be packed at the same coordinate even if they are not similar. The proposed lower bounds increase the computing time in some instances.

Goncalves (2007) proposes combination of the placement procedure and a genetic algorithm based on random keys to solve a two-dimensional orthogonal knapsack problem. The objective function is minimizing the amount of trim loss. The proposed algorithm is relatively complex and time consuming.

Bortfeldt and Winter (2009) propose a genetic algorithm for the two dimensional orthogonal knapsack problems. The proposed algorithm considers both guillotine and non-guillotine variant of the problem and an orientation constraint also may be considered. The items which have to be placed in the container can be constrained as well as unconstrained. The authors claim that for large instances of the non-guillotine constrained 2D knapsack, GA solution is significant.

Joncour et al. (2010) suggest a method for finding a feasible solution for a two dimensional orthogonal knapsack problem which is based on the characterization of the interval graph. The problem is packing the rectangular items in a big rectangular container without overlapping. It is assumed that the rotation of the items is not allowed. In order to find infeasible solutions earlier, they used a method similar to Clautiaux et al. (2007). The approach suggested in this paper is superior to the Fekete and Schepers' (1997) method since by creating MPQ-trees, the search space stays within the set of interval graphs.

Dolatabadi et al. (2012) propose a recursive exact algorithm to solve the two-dimensional guillotine knapsack problem. The problem is packing small rectangular items in a bigger rectangular sheet. The packing is orthogonal and the rotation of the items is not allowed. At first, the sets of associated guillotine packing are built; then, the algorithm is divided into two exact algorithms in order to solve the two-dimensional knapsack problem. The first algorithm is on the basis of iterative implementation of recursive method with different input parameters, and the second one is based on an ILP model. The branch-and-cut method is used to confirm the optimality of the solution.

Leung et al. (2012) propose a hybrid simulated annealing metaheuristic for the two-dimensional knapsack problem. The authors first define a fitness strategy to identify which item has to be packed first in a given position. A heuristic algorithm generates

the solution based on this fitness strategy. Finally, the simulated annealing approach is used to jump out of the greedy strategy's local optimal trap. The items are packed into stock sheet one at a time for a given sequence of items. For any available position, the fitness value of each item, which has to be packed, is calculated and then the item with maximum fitness value is selected. If more than one item has the same maximum fitness value, the algorithm selects the one by the input order of the items. The proposed hybrid algorithm combines the greedy strategy approach and simulated annealing to gain a better solution. The greedy algorithm is used to search a good sequence of items; then a simulated annealing heuristic is applied to do a broader search to gain a better solution.

2.2. Three Dimensional Knapsack Problem

Some papers consider the three dimensional cutting and packing problem (or container loading) and attempt to model it or propose solution methodology for such problems. The focus of most of these papers is on the rectangular bins. As multi dimensional C&P problems are strongly NP-hard, only very few exact algorithms have been proposed for such problems.

Fekete and Schepers (1997) propose a method for modeling more-dimensional packing problem based on the graph characterization of feasible packing. They define a graph based on the relative positions of boxes. The graph is proven to be an interval graph. The authors consider a set of boxes to be packed into a container and focus on an orthogonal packing problem. The method cannot handle further constraints like fixing the position of some items, and the results are limited to two dimensional problems. Fekete and Schepers (1997) present a method in order to gain lower bounds for more-dimensional knapsack problem. They, moreover, illustrate that all known lower bounds for such problems can be improved by this method. The authors describe heuristics for dismissing infeasible packings. Fekete and Schepers (1997) show how this method can be applied to more dimensional knapsack problem.

Fekete and Schepers (2004) propose a new method for obtaining classes of lower bound for higher-dimensional packing problem. The authors apply a number of volume tests after modifying the size of boxes. The relative bulkiness of the items and

the way that they can be combined is reflected by transformation. They present a combinatorial characterization of feasible packing as a basis for branch and bound approach. The major objective of this paper is to define good criteria for removing a candidate set of boxes. Dual feasible function is a way to build conservative scales. All known classes of lower bound for higher-dimensional packing problem can be improved by using the proposed approach. The authors suggest a strong method for solving higher dimensional problems by combining these classes of bounds and characterization of feasible packing as described in Fekete and Schepers (1997). The computational results are mainly limited to the two-dimensional packing problem.

Hifi (2004) proposes a dynamic algorithm and an exact depth-first search in order to solve the three dimensional cutting problem. Orientation and guillotine constraint are considered. Sixty four problem instances were tested which include up to 50 boxes. Optimal solutions are obtained for most of the instances but not all of them.

Although considerable advancement has been made in the development of exact algorithms, heuristic algorithms still play an important role in solving three-dimensional knapsack problems. Only heuristic methods can provide reasonable solutions within acceptable running times for problem instances of real-world size.

Martello et al. (2007) consider the orthogonal three-dimensional bin packing problem where box rotation is not allowed. Both general and robot packable variants of bin-packing problem are presented. The algorithm is on the basis of two-level decomposition approach and consists of two parts. In the first part the boxes are assigned to the bins. In the second part, a single bin is filled while the objective function is maximizing the filled volume. The proposed methodology can be used as a whole for solving the three-dimensional bin packing problem or just for filling a single bin.

Egeblad and Pisinger (2009) propose a simulated annealing based methodology for the two and three dimensional knapsack problems. A three-dimensional knapsack model is presented. New constraints can be added to this model such as fixing the position of items or rotation. The authors present an iterative heuristic for the two-dimensional knapsack problem which is based on the sequence pair. In each iteration,

the sequence pair is transformed to the packing. In order to control the heuristic method simulated annealing is used. For three-dimensional knapsack problem, sequence triple technique is used. The authors prove that a fully robot packable packing can be obtained through sequence triple representation. Robot packing is a packing obtained by locating items starting from left-bottom-behind (LBB) corner. It is represented in three sequences; for any sequence the relationship of each two items is defined. To find a placement for any given sequence, three constraint graphs are constructed. Like 2DKP, the meta-heuristic annealing is used to solve the three-dimensional knapsack problem. Rotation of boxes is not considered in the three-dimensional model and experiments.

Wu et al. (2010) consider the three-dimensional bin packing problem with variable bin height. The bins and boxes are rectangular and the object rotation is allowed. Guillotine constraint is not imposed. Moreover, bin heights can change in order to fit bin contents. A mixed integer programming model is proposed, and a bin packing algorithm which is based on packing index is used to develop the problem feature and as a building block for genetic algorithm. The authors also present the situation when more than one type of bin is used. A genetic algorithm-based heuristic is proposed for packing a batch of objects. The algorithm is on the basis of extreme point method. The authors consider both single bin packing and batch bin packing problems.

Amossen and Pisinger (2010) consider the multi-dimensional orthogonal bin-packing problem with guillotine constraints where rotation is not allowed. The authors experimentally evaluate three packing methods –unrestricted, robot packable, guillotine cuttable- based on the solution time and quality.

Models provide information on optimal objective function value and bounds. They are helpful to assess the solution quality of heuristic algorithms. Modeling three dimensional knapsack problems, while considering practical constraints, is still at its beginning.

Junqueira et al. (2012) present mixed integer linear programming models for the container loading problem. Vertical and horizontal stability of the cargo as well as cargo load bearing strength are taken into account in the proposed model. The models

can be extended in order to apply to other variants of container loading problem as well. However, the models are only able to handle moderate size problems.

In addition, container loading problems have been studied from a more general and practical view. Murty et al. (2005) propose a decision support system in order to develop optimal decisions. These decisions are used to route container trucks, find the storage place for containers, number of assigned container and truck scheduling. The proposed decision system is applied to the Hong Kong International Terminals. Murty et al. (2005) define a selection of inter-related decisions which is made at the container terminal during a day. The main goal of these decisions is minimizing the resource and the trucks waiting time, and maximizing the container volume utilization. The author use decision support systems to make these decisions since these kinds of decisions are complex and large scale. Petering and Murty (2009) develop a simulation study about terminal's average quay crane rate, and how the long-run performance of seaport container terminal is related to storage block length and yard crane deployment. Several scenarios are evaluated. These experiments are direct connection between length of the block and long-run performance in the container terminal.

As mentioned, both exact algorithms and heuristic methods are proposed in the published literature. Leung et al. (2001), Goncalves (2007), Bortfeldt & Winter (2009), Leung et al. (2012), Egeblad & Pisinger (2009) and Wu et al. (2010) propose heuristic algorithms for different types of packing problems. While, Fekete & Schepers (1997), and Hifi (2004) propose exact methods. The following table compares some relevant papers and models, and shows their similarities, differences and superiority.

Table 2.1. Summary of Some relevant Papers

Papers	Problem type	Assumption	What they do?	Solution Methodology	Superiority to other papers	Limitation
Egeblad & Pisinger (2009)	2D and 3D knapsack problem	Items are strongly heterogeneous, no rotation	Mathematical Model	sequence based representation (SA based approach)	Sequence pair and triple is one of the successful representations	Fixed orientation for 3D
Bortfeldt & Winter (2009)	2D Orthogonal knapsack problem	Guillotine & non-guillotine, orientation constraint may be considered	Heuristic algorithm	GA	GA is suitable for large instances of the non-guillotine constrained	compare to other methods GA is in the mid-table
Junqueira et al. (2012)	container loading problem	vertical and horizontal stability, load bearing strength	MILP	GAMS	extend in other variants of container loading problem	Only able to handle moderate size problems
Wu et al. (2010)	3D bin packing problem with variable bin height	Rectangular boxes, , Guillotine constraint is not imposed	Mathematical Model	GA & extreme point	both single bin packing and batch bin packing problem is considered, object rotation is allowed	
Amossen & Pisinger (2010)	multi-dimensional orthogonal bin-packing problem	Guillotine, no rotation	evaluate three packing methods	unrestricted, robot packable, guillotine cuttable		Fixed orientation
Martello et al. (2007)	3D orthogonal bin packing problem	rotation is not allowed, general and robot packable	Decomposition algorithm	two-level decomposition approach	can be used as a whole for solving three-dimensional bin packing problem or just for filling a single bin	Fixed orientation
Goncalves (2007)	2D knapsack problem	Orthogonal, fixed orientation	Solving 2D packing problem	Hybrid genetic algorithm		Relatively complex, long computational time compared to Leung et al. (2012)
Leung et al. (2001)	2D non-guillotine cutting stock	Fixed orientation, orthogonal,	Heuristic algorithm	Genetic algorithm and simulated annealing		cannot produce all feasible packing

Fekete & Schepers (1997)	More-dimensional packing problem	Fixed orientation, orthogonal	Modeling packing based on the graph characterization of feasible packing	Interval Graph		method cannot handle further constraints
Given Problem	3D knapsack problem	Rectangular boxes	Finding more practical packing, Mathematical formulation	SA and sequence triple	Rotation allowed, vertical stability, pre-placed boxes	

2.3. Research Gaps

According to the literature, not all papers consider box rotation since it increases the search space significantly. Moreover, bin stability is just taken into account in some of the container loading problems and it has not been considered in three-dimensional knapsack problem. Vertical stability is one of the realistic constraints which should be taken into account in 3D knapsack problems, so all the packed boxes are supported by the bin floor or other boxes top and do not fall down. In addition, to the best of our knowledge, pre-placed boxes (obstacles) has not been studied in three-dimensional knapsack problems, which is so essential for such problems since it is often required to place certain boxes in certain positions. Such a constraint can be also considered when the bin does not have rectangular shape. Therefore, it is important to study more practical constraints in the knapsack problem. In the given problem, box rotation is taken into account in order to find more practical packings. Also, preplaced boxes (bin with some obstacles) and vertical stability which are real-world constraints are studied.

CHAPTER 3
Problem Formulation

3.1. Problem Definition

In this study, the three-dimensional knapsack problem is considered where there is one bin with fixed size and a set of boxes; each box has an associated size. The aim is to find an efficient solution methodology in order to pack rectangular boxes in a single bin so that the total value of the packed boxes is maximized, or equivalently the empty spaces left are minimized. The boxes are assumed to be strongly heterogeneous which means there is a relatively high number of different types of boxes and a small number of boxes for each box type (Wascher et al., 2007). Moreover, the packing is considered feasible if each box lies entirely in the bin, and the packed boxes do not overlap. The edges of all boxes must be parallel to the edges of the bin (orthogonal packing). The bin and boxes are assumed to be of rectangular shape.

Some practical considerations which play an important role in modeling more realistic knapsack problems are presented such as box rotation and bin stability. Boxes are able to freely rotate in six different orientations, need not to be packed in layers, and the bottom of each box must be supported by the top of other boxes or the bin floor. In addition, some boxes are considered as pre-placed boxes or obstacles, whose left-bottom-behind (LBB) corner should be placed in a specific position. The value of each box is equal to its volume. It is assumed that the dimensions of all boxes and the bin are integers, thus the placement are to be done in integer steps. Let C be a rectangular container with width W, height H and depth D. The origin of the Cartesian coordinate system is located at the LBB corner of the container, and l_i, h_i, and w_i are respectively, the length, height and depth of box type *i*. For each packed box, (x_i, y_i, z_i) represents the coordinates of the LBB corner of the box.

A mixed integer programming formulation is presented for the given problem. Some real-world knapsack problem constraints are considered in the model which, to the best of our knowledge, have not been studied yet. These constraints are vertical stability and pre-placed boxes. Since the three-dimensional knapsack problem is NP-hard, it is difficult to solve. In addition, the flexibility of the orientation of boxes

significantly increases the search space, so the difficulty of finding the optimal solution is enhanced as well. Some exact algorithms as well as heuristic methods are proposed in the published literature. As exact algorithms require more time to find a solution, heuristic approaches are more popular and can be good alternatives to find optimal or near optimal solution. The proposed three-dimensional solution methodology is based on Egeblad and Pisinger's (2009) sequence triple representation. Simulated annealing is used as heuristic method.

3.2. Mathematical Formulation

A mixed-integer programming model of the 3D-knapsack problem is introduced in this section. The mathematical model is based on Egeblad and Pisinger (2009) and Wu et al. (2010). Some modifications are made in their model which include considering vertical stability and pre-placed boxes constraints. Egeblad and Pisinger (2009) and Wu et al. (2010) do not consider these important and practical constraints. Constraints (1) – (4) are based on Egeblad and Pisinger (2009); they did not consider the box orientation in their model. The binary position variables which show the orientation of the boxes are based on Wu et al. (2010). However, constraints (5) – (17) are new constraints added to the model which are described in the following sections.

3.2.1. Notations

The variables and parameters used in the mathematical formulation are introduced as follows:

- *Variables:*

(x_i,y_i,z_i): LBB coordinates of box i

Xw_i, Zw_i: $\begin{cases} 1 & \text{whether width of box i is parallel to the container's X and Z} \\ 0 & \text{otherwise} \end{cases}$

Yh_i: $\begin{cases} 1 & \text{if height of box i is parallel to the container's Y} \\ 0 & \text{otherwise} \end{cases}$

Zd_i: $\begin{cases} 1 & \text{if depth of box i is parallel to the container's Z} \\ 0 & \text{otherwise} \end{cases}$

r_{ij}, l_{ij}: $\begin{cases} 1 & \text{if box i is to the right of or to left of box j} \\ 0 & \text{otherwise} \end{cases}$

o_{ij}, u_{ij}: $\begin{cases} 1 & \text{if box i is over or under box j} \\ 0 & \text{otherwise} \end{cases}$

b_{ij}, f_{ij}: $\begin{cases} 1 & \text{if box i is behind or in-front-of box j} \\ 0 & \text{otherwise} \end{cases}$

s_i: $\begin{cases} 1 & \text{if box i is packed} \\ 0 & \text{otherwise} \end{cases}$

y^a_{ij}: $\begin{cases} 1 & \text{if } x_j \geq x_i \\ 0 & \text{otherwise} \end{cases}$

x^a_{ij}: $\begin{cases} 1 & \text{if } x_j < x'_i \\ 0 & \text{otherwise} \end{cases}$

y^b_{ij}: $\begin{cases} 1 & \text{if } z_j \geq z_i \\ 0 & \text{otherwise} \end{cases}$

x^b_{ij}: $\begin{cases} 1 & \text{if } z_j < z'_i \\ 0 & \text{otherwise} \end{cases}$

y^c_{ij}: $\begin{cases} 1 & \text{if } x'_j > x_i \\ 0 & \text{otherwise} \end{cases}$

x^c_{ij}: $\begin{cases} 1 & \text{if } x'_j \leq x'_i \\ 0 & \text{otherwise} \end{cases}$

y^{d}_{ij}: $\begin{cases} 1 & \text{if } z'_j > z_i \\ 0 & \text{otherwise} \end{cases}$

x^{d}_{ij}: $\begin{cases} 1 & \text{if } z'_j \leq z'_i \\ 0 & \text{otherwise} \end{cases}$

z^{a}_{ij}: $\begin{cases} 1 & \text{if } x_i \leq x_j < x'_i \\ 0 & \text{otherwise} \end{cases}$

z^{b}_{ij}: $\begin{cases} 1 & \text{if } z_i \leq z_j < z'_i \\ 0 & \text{otherwise} \end{cases}$

z^{c}_{ij}: $\begin{cases} 1 & \text{if } x_i < x'_j \leq x'_i \\ 0 & \text{otherwise} \end{cases}$

z^{d}_{ij}: $\begin{cases} 1 & \text{if } z_i < z'_j \leq z'_i \\ 0 & \text{otherwise} \end{cases}$

Cs_1: $\begin{cases} 1 & \text{if } x_i \leq x_j < x'_i \text{ and } z_i \leq z_j < z'_i \\ 0 & \text{otherwise} \end{cases}$

Cs_2: $\begin{cases} 1 & \text{if } x_i \leq x_j < x'_i \text{ and } z_i < z'_j \leq z'_i \\ 0 & \text{otherwise} \end{cases}$

Cs_3: $\begin{cases} 1 & \text{if } x_i < x'_j \leq x'_i \text{ and } z_i \leq z_j < z'_i \\ 0 & \text{otherwise} \end{cases}$

Cs_4: $\begin{cases} 1 & \text{if } x_i < x'_j \leq x'_i \text{ and } z_i \leq z_j < z'_i \\ 0 & \text{otherwise} \end{cases}$

$$x'_i = x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i)$$

$$z'_i = z_i + d_i Zd_i + h_i (1 - Zw_i - Zd_i) + w_i Zw_i$$

- *Parameters:*

(w_i,h_i,d_i): width, height and depth of box i

(W,H,D): width, height and depth of the container

(r,s,k): LBB coordinates of the pre-placed boxes

(a, b, c, d): Binary orientation parameters of the pre-placed boxes

P_i: value of box i

3.2.2. Assumptions

The following assumptions are considered for the mix integer linear model:

1. The boxes are strongly heterogeneous.
2. The boxes must be located orthogonally
3. The boxes are able to freely rotate
4. The box and bin dimensions are assumed to be non-negative integer
5. The value of a boxes is equal to its volume
6. The X, Y, and Z axes of the bin are shown in the following figure.

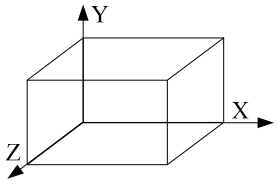


Figure 3.1. The X, Y, and Z axes of the bin

3.2.3. MILP

The objective Function is maximizing the value of packed boxes:

$$Max \sum_{i=1}^{n} P_i s_i$$

Subject to:

$$r_{ij} + l_{ij} + b_{ij} + f_{ij} + u_{ij} = s_i + s_j - 1 \qquad \forall i,j \quad i \neq j \qquad (1)$$

$$x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \leq x_j + M(1-l_{ij})$$

$$\forall i,j \quad i \neq j \qquad (2a)$$

$$x_j + w_j Xw_j + h_j(Zw_j - Yh_j + Zd_j) + d_j(1 - Xw_j - Zw_j + Yh_j - Zd_j) \leq x_i + M(1-r_{ij})$$

$$\forall i,j \quad i \neq j \qquad (2b)$$

$$z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i \leq z_j + M(1-b_{ij}) \qquad \forall i,j \quad i \neq j \qquad (2c)$$

$$z_j + d_j Zd_j + h_j(1 - Zw_j - Zd_j) + w_j Zw_j \leq z_i + M(1-f_{ij}) \qquad \forall i,j \quad i \neq j \qquad (2d)$$

$$y_i + h_i Yh_i + w_i(1 - Xw_i - Zw_i) + d_i(Xw_i + Zw_i - Yh_i) \leq y_j + M(1-u_{ij})$$

$$\forall i,j \quad i \neq j \qquad (2e)$$

$$y_j + h_j Yh_j + w_j(1 - Xw_j - Zw_j) + d_j(Xw_j + Zw_j - Yh_j) \leq y_i + M(1-o_{ij})$$

$$\forall i,j \quad i \neq j \qquad (2f)$$

$$x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \leq W \qquad (3a)$$

$$y_i + h_i Yh_i + w_i(1 - Xw_i - Zw_i) + d_i(Xw_i + Zw_i - Yh_i) \leq H \qquad (3b)$$

$$z_i + d_i Zd_i + h_i(1 - Zw_i - Zd_i) + w_i Zw_i \leq D \qquad (3c)$$

$$Xw_i + Zw_i \leq 1 \qquad (4a)$$

$$Zw_i + Zd_i \leq 1 \qquad (4b)$$

$$0 \leq Zw_i - Yh_i + Zd_i \leq 1 \qquad (4c)$$

$$0 \leq 1 - Xw_i - Zw_i + Yh_i - Zd_i \leq 1 \qquad (4d)$$

$$0 \leq Xw_i + Zw_i - Yh_i \leq 1 \qquad (4e)$$

$$(x_i, y_i, z_i) = (r, s, k) \qquad \forall i \in Pb \qquad (5)$$

$$(Xw_i, Zw_i, Zd_i, Yh_i) = (a,b,c,d) \qquad \forall i \in Pb \qquad (6)$$

$$x_j - x_i \le M.\ y^a_{ij} \qquad x_j - x_i \ge M\,(y^a_{ij} - 1) \tag{7a}$$

$$x'_i - x_j \le M.\ x^a_{ij} \qquad x'_i - x_j \ge M\,(x^a_{ij} - 1) + 0.5 \tag{7b}$$

$$(y^a_{ij} + x^a_{ij} - 1)/2 \le z^a_{ij} \le (y^a_{ij} + x^a_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{7c}$$

$$z_j - z_i \le M.\ y^b_{ij} \qquad z_j - z_i \ge M\,(y^b_{ij} - 1) \tag{8a}$$

$$z'_i - z_j \le M.\ x^b_{ij} \qquad z'_i - z_j \ge M\,(x^b_{ij} - 1) + 0.5 \tag{8b}$$

$$(y^b_{ij} + x^b_{ij} - 1)/2 \le z^b_{ij} \le (y^b_{ij} + x^b_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{8c}$$

$$x'_j - x_i \le M.\ y^c_{ij} \qquad x'_j - x_i \ge M\,(y^c_{ij} - 1) + 0.5 \tag{9a}$$

$$x'_i - x'_j \le M.\ x^c_{ij} \qquad x'_i - x'_j \ge M\,(x^c_{ij} - 1) \tag{9b}$$

$$(y^c_{ij} + x^c_{ij} - 1)/2 \le z^c_{ij} \le (y^c_{ij} + x^c_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{9c}$$

$$z'_j - z_i \le M.\ y^d_{ij} \qquad z'_j - z_i \ge M\,(y^d_{ij} - 1) + 0.5 \tag{10a}$$

$$z'_i - z'_j \le M.\ x^d_{ij} \qquad z'_i - z'_j \ge M\,(x^d_{ij} - 1) \tag{10b}$$

$$(y^d_{ij} + x^d_{ij} - 1)/2 \le z^d_{ij} \le (y^d_{ij} + x^d_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{10c}$$

$$(z^a_{ij} + z^b_{ij} - 1)/2 \le Cs_1 \le (z^a_{ij} + z^b_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{11}$$

$$(z^a_{ij} + z^d_{ij} - 1)/2 \le Cs_2 \le (z^a_{ij} + z^d_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{12}$$

$$(z^c_{ij} + z^b_{ij} - 1)/2 \le Cs_3 \le (z^c_{ij} + z^b_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{13}$$

$$(z^c_{ij} + z^d_{ij} - 1)/2 \le Cs_4 \le (z^c_{ij} + z^d_{ij})/2 \qquad \forall\, i,j \qquad i \ne j \tag{14}$$

$$Cs_1 + Cs_2 + Cs_3 + Cs_4 = u_{ij} + o_{ij} \qquad \forall\, i,j \qquad i \ne j \tag{15}$$

$$x'_i = x_i + w_i Xw_i + h_i(Zw_i - Yh_i + Zd_i) + d_i(1 - Xw_i - Zw_i + Yh_i - Zd_i) \tag{16}$$

$$z'_i = z_i + d_i Zd_i + h_i\,(1 - Zw_i - Zd_i) + w_i Zw_i \tag{17}$$

$$r_{ij},\ l_{ij},\ b_{ij,}\ f_{ij},\ u_{ij} \in \{0,1\} \tag{18}$$

$$Xw_i,\ Zw_i,\ Zd_i,\ Yh_i \in \{0,1\} \tag{19}$$

$x^a_{ij}, x^b_{ij}, x^c_{ij}, x^d_{ij}, y^a_{ij}, y^b_{ij}, y^c_{ij}, y^d_{ij}, z^a_{ij}, z^b_{ij}, z^c_{ij}, z^d_{ij} \in \{0,1\}$ (20)

$s_i, Cs_1, Cs_2, Cs_3, Cs_4 \in \{0,1\}$ (21)

$(x_i, y_i, z_i) \geq 0$ (22)

Constraint (1) ensures that if box i and box j are packed then they must be placed left, right, under, over, behind or in-front-of each other. Constraints (2) guarantee that any two boxes i and j do not overlap, while considering the box rotation. It includes six parts; constraint (2a) and (2b) find the x coordinate of the box to be packed; constraint (2c) and (2d) are used to find its z coordinate, and constraint (2e) and (2f) calculate its y coordinate. The binary position variables (Xw_i, Zw_i, Yh_i, Zd_i) are used to allow box rotations. Constraint set (3) ensures that all boxes are placed within the bin's dimensions. Constraint (3a) makes sure that the box dimensions do not exceed the bin's width; while constraints (3b) and (3c) are related to the bin's height and depth. Constraint set (4) is used to make sure that the binary variables which show the position of the boxes are controlled to represent practical positions. Constraint (4a) guarantees the width of the packed box is not parallel to both X and Z axis. Constraint (4b) ensures that the width and depth of each packed box are not parallel to Z axes simultaneously. Constraint (4c) shows that the height of box i cannot be parallel to both Z and Y axes. Constraints (4d) and (4e) also control the orientation of the packed boxes, and ensure that the width, height, and depth of each packed box are not parallel to two axes simultaneously. Constraint (5) and (6) are used to fix the coordinates and orientation of the pre-placed boxes, where Pb is a set of preplaced boxes. Constraints (7)–(10) ensure vertical stability. These constraints compare the four corners of each newly packed box with the points that cover the top of other packed boxes. If one of the corners has the same x and z coordinates as one of the mapped points, it means that the new box is located under or above that box. Constraint set (7) is used to define the binary variable z^a_{ij} and includes three parts. Constraint (7a) ensures that if $x_j \geq x_i$, then y^a_{ij} is equal to one; otherwise it is equal to zero. Constraint (7b) makes sure that if $x_j < x_i$, then x^a_{ij} is one; otherwise it is equal to zero. Constraint (7c) guarantees when y^a_{ij} and x^a_{ij} are both equal to one, then z^a_{ij} is equal to one. Similarly, constraint sets (8), (9), and (10) are used to define the binary variables z^b_{ij}, z^c_{ij}, and z^d_{ij}. Constraints (11)-(14) show whether the x and z coordinates of the new box's corner are equal to x and z coordinates of the mapped points on the top of the packed

boxes. Constraint (15) ensures that if these coordinates are the same, the new box should be located on top of or under the packed box. Constraints (16) and (17) define x'_i and z'_i. Constraints (18) - (21) represent the binary variables, and constraint (22) represents the integer variables.

The given mathematical model has $21n^2+9n$ binary variables and 3n integer variables. It was coded in GAMS/Cplex, and the computational tests run on an Intel® Core™ i5 CPU @ 2.67GHz processor with 4.0 GB RAM. The model at first was run for an instance with 5 boxes; it reached the optimal solution in 53 seconds. Then the instance with 6 boxes has been considered, the solution time is equal to 6 minutes and 14 seconds. However, the solution time for the instance with 7 boxes increased significantly to 4 hours and 4 minutes; the number of variables in such instance is 1113. The optimal results for instance with 8 boxes- 1440 variables- was obtained after 21 hours and 39 minutes. GAMS was not able to reach optimal solution for instance with 9 boxes – 1809 variables- even after 3 days, thus the algorithm was terminated before reaching the solution. According to the results, optimal solutions only for small size instances (up to 8 boxes) were possible in a reasonable time. Thus, heuristic algorithm is required to get faster solutions for larger instances.

3.3. Two-dimensional Model

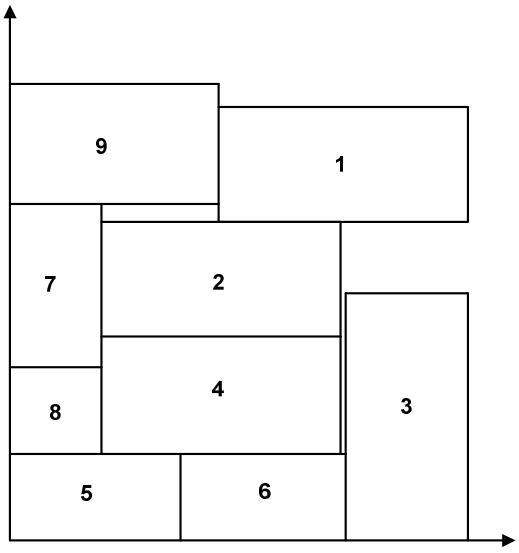
Although the proposed model is considered a three-dimensional knapsack problem it can be modified in order to solve two-dimensional problems as well. The z axis should be omitted in order to adjust the model. Since two dimensional problems are simpler than three-dimensional ones they can be solved in a shorter time. As an example, the instance of 4 different types of rectangles (totally 10 rectangles) is studied. The dimensions and maximum allowed number of these rectangles are shown in table 3.1. The dimensions of the bin, which is two dimensional as well, are equal to 900×900 (mm^2).

Table 3.1. 2D Rectangles Dimensions and Maximum Allowed Number

Rectangle type	Width(mm)	Height(mm)	Max. allowed no.
1	229	483	4
2	165	330	3
3	165	165	1
4	229	406	1

The optimal solution is obtained after 3 hours and 37 minutes. Figure 3.1 shows the obtained result. Compared to the three dimensional instances, the optimal solution can be obtained sooner. However, the solution time is not reasonable for the 2D instances as well, thus it is better to use a heuristic algorithm to reach the results in a shorter time.

Figure 3.2. 2D Instance Result



CHAPTER 4
Solution Methodology

4.1. Three Dimensional Algorithm

Based on Egeblad and Pisinger's work (2009), the three sequences considered for the boxes must be packed. These sequences show the relative box locations. They are known as sequence triple. Sequence triple is one of the most successful representations in the literature and defines the packing order. As mentioned in Egeblad and Pisinger (2009), the sequence triple does not create all three-dimensional packing; however, it is proved that a fully robot packable packing is obtainable with this representation. A *robot packing* is a packing that can be obtained by placing boxes from the LBB corner of the bin while each box is in-front-of, on the right side, or above other boxes. If all six rotations of the packing are robot packable, the packing is known as a fully robot packable packing. Although Egeblad and Pisinger (2009) claim that their algorithm creates normalized packings, their results are not normalized. Normalized packing is a packing when all boxes are placed as far left, down, and back as possible without overlapping, and every new box touches an already placed box on its left, lower, and back side. However, according to their results some of the packed boxes are placed in the air.

The solution methodology section is organized as follows: first, sequence triple is described in section 4.1.1 which is used in section 4.1.2 in order to place the boxes. Simulated annealing is defined in section 4.1.3 to control the local neighbourhood search. Orthogonal rotation, pre-placed boxes (obstacles), four-corner packing, and box insertion order are explained in sections 4.1.4, 4.1.5, 4.1.6, and 4.1.7, respectively.

4.1.1. Sequence Triple

Three sequences A, B, and C represent the fully robot packable packing, where A, B, and C are permutations of the numbers 1 ... n, and n is the total number of boxes to be placed in the bin. These sequences denote the relative placement of each of the two i and j boxes with respect to each other. Each sequence is defined as follows:

- *A-chain*: If box i appears before box j in the A-chain, then box i is located to the left of, on top of, or in front of box j.
- *B-chain*: if box i appears before box j in the B-chain, then box i is located behind, to the left of, or below, box j.
- *C-chain*: If box i appears before box j in the C-chain, then box i is located to the right, under, or in front of box j.

4.1.2. Placement algorithm

Based on the given three sequences, box i is located on the left side of box j if it appears before box j in A-chain and B-chain and after box j in C-chain. Box i is located below box j if it appears before box j in B-chain and C-chain and after box j in A-chain. Moreover, box i is placed behind box j if it appears after box j in B-chain and before it in A-chain and C-chain, or if box i is placed after box j in all sequences. It is observed that box i always appears before box j in B-chain for all three given placements. Thus, the order of placement of the boxes in the bin can be based on the order of B-chain. The first box is placed at the origin, and the succeeding boxes are placed according to their relative position to already packed boxes. The coordinates of each new box are calculated based on the following formula:

$$x_i = \max(0, \max_{j \in P_x}(x_j + w_j))$$

$$y_i = \max(0, \max_{j \in P_y}(y_j + h_j))$$

$$z_i = \max(0, \max_{j \in P_z}(z_j + d_j))$$

where P_x, P_y, and P_z are the subsets of packed boxes located on the left, below, and behind the new box. In order to consider vertical stability and reduce the gap between the boxes, some modifications have been applied to Eglebad and Pisinger's (2009) procedure. These modifications are explained in the following section.

- ***Vertical Stability***

As it is assumed that (x,y,z) coordinates of boxes and their dimensions are integer, it is possible to map a set of points that a certain box covers.. Let (x_i, y_i, z_i) be the LBB

coordinates of each to be packed box. The algorithm considers four corners of the given box. If x and z coordinates of one of these corners are equal to the coordinates of one of the points at the top of any packed box, it returns the height of that box. Then, the y coordinate of the new box would be equal to maximum of those values. The proposed approach is illustrated in the following:

1. Consider (x_i, y_i, z_i)

   $\forall j \in P_y$: compute x'_j and z'_j

   Where $x_j \le x'_j \le x_j+w_j-1$ and $z_j \le z'_j \le z_j+d_j-1$

   If ($x_i = x'_j$ and $z_i = z'_j$) then

   Return $y_j+ h_j$

   Else Go to 2

2. Consider $(x_i + w_i, y_i, z_i)$

   $\forall j \in P_y$: compute x'_j and z'_j

   Where $x_j+1 \le x'_j \le x_j+w_j$ and $z_j \le z'_j \le z_j+d_j-1$

   If ($x_i+ w_i = x'_j$ and $z_i = z'_{j)}$ then

   Return $y_j+ h_j$

   Else Go to 3

3. Consider $(x_i, y_i, z_i + d_i)$

   $\forall j \in P_y$: compute x'_j and z'_j

   Where $x_j \le x'_j \le x_j+w_j-1$ and $z_j+1 \le z'_j \le z_j+d_j$

   If ($x_i = x'_j$ and $z_i+ d_i = z'_j$)

   Return $y_j+ h_j$

   Else Go to 4

4. Consider $(x_i + w_i, y_i, z_i + d_i)$

   $\forall j \in P_y$: compute x'_j and z'_j

   Where $x_j+1 \le x'_j \le x_j+w_j$ and $z_j+1 \le z'_j \le z_j+d_j$

   If ($x_i+ w_i = x'_j$ and $z_i+d_j = z'_j$) then

   Return $y_j+ h_j$

   Else Return 0

Return $y_i = \max(0, \max_j(y_j + h_j))$

The algorithm pushes each packed box downward where possible such that its bottom can be supported by the bin floor or by the top of other packed boxes.

4.1.3. Simulated annealing

Although it is relatively simple to develop a simulated annealing heuristic, choosing a good neighborhood and cooling procedure, which itself depends on several different parameters, is usually necessary for the algorithm to work efficiently. The cooling procedure is different for various types of problem and even between instances of the same problem. Therefore, it is difficult to find out a good cooling procedure. In the proposed simulated annealing algorithm, the temperature is reduced when a new solution is accepted, according to the following function:

$$t \rightarrow t/(1+ \beta t)$$

where β is the cooling parameter. Besides the cooling down procedure, the process is allowed to heat up again whenever it is appeared be getting trapped. The heating up function is:

$$t \rightarrow t/(1- \alpha t)$$

where α is the heating parameter. The temperature is reduced when the solution is accepted and increased when the solution is rejected. α must be smaller than β as the number of acceptances is small relative to number of rejections (Dowsland, 1993).

The neighbourhood of each solution is defined as one of these five permutations: either exchange two boxes from one of the sequences; exchange two boxes in sequences A and B; exchange two boxes in sequences A and C; exchange two boxes in sequences C and B; or exchange two boxes in all sequences. An overview of the simulated annealing algorithm is as follows:

```
// Prepare the initial state and volume
temperature := initial_temperature
initial_state := randomly generated state
best_state := initial_state
best_volume := volume_utilized(best_state)
```

```
while (time is not up) do
   neighbours := generate_neighbourhood(best_state)
   neighbour := randomly select an element from neighbours
neighbour_volume := volume_utilized(neighbour)
found_better := false
   if (neighbour_volume>best_volume) then
found_better := true
   else
      // We accept a worse solution at random, but the chance of
      // doing so decreases with the temperature.
      temperature := temperature / (1+β*temperature)
      delta := (best_volume – neighbour_volume) / best_volume
i := random number between 0 and 1
      if (i< e^( -delta / temperature ) ) then
found_better := true
      else
     //increase temperature
      temperature := temperature / (1-α*temperature)
      end if
   end if
if (found_better) then
      selected := selected + 1
best_state := neighbour
best_volume := neighbour_volume
   end if
end while
return best_state
```

The solutions are compared based on the bin utilization. The formula used for calculating the utilization percentage is as follows:

$$utilization\_percentage = \frac{total\_volume\_of\_packed\_boxes}{volume\_of\_bin} \times 100$$

4.1.4. Orthogonal Rotation

The boxes are allowed to be rotated orthogonally with respect to the bin. Suppose the width, height, and depth of all boxes are respectively parallel to x, y, and z axis, and w_i, h_i, and d_i represents the width, height, and depth of box i, respectively. It is possible to obtain better packings if the boxes were rotated in different directions. Egeblad and Pisinger (2009) considered box rotation only for the two dimensional instances but neglected to include it in the three dimensional experiments. Boxes are allowed to be rotated in one of the following orientation:

WHD: Standard orientation.
WDH: Swap the height and the depth.
HWD: Swap the width and the height.
HDW: Swap the width and the height, and then swap the height with the depth.
DHW: Swap the depth with the width.
DWH: Swap the depth with the width, and then swap the depth with the height.

The given rotation is applied to the simulated annealing by adding an additional transformation to the neighbourhood generating routine. The orientation of the boxes is generated randomly at first. Thus, an additional vector R which shows the orientation of the boxes is stored as well as the sequence triple.

4.1.5. Obstacles

Suppose *O* is a set of rectangular obstacles with known coordinates (x, y, z) and known dimensions (w, h, d). At the beginning of the algorithm, the obstacles are fixed into the bin. The packing is created from the sequence triple and those boxes that overlap with any obstacles in the set are removed. The container free volume is calculated as follows:

Bin free volume = volume of bin – total volume of obstacles

4.1.6. Four-corner packing

Four packing schemes, one for each corner are created. First, the coordinates of the boxes are calculated relative to the current origin. Then, their real (x, y, z) coordinates are calculated relative to the real origin of the container which is its LBB corner. The processing technique is as follows:

```
W := bin width
H := bin height
D := bin depth
w := box width
h := box height
d := box depth
if (loading from front) then
  // No change needed: this is the default loading method.
return <x,y,z>
else if (loading from rear) then
return<W – x – w, y, D – z – d>
else if (loading from left side) then
return<W – z – w, y, x>
else if (loading from right side) then
return<z, y, D – x – w>
end if
```

4.1.7. Order of box insertion

As mentioned earlier, the order of inserting boxes into the container is based on B-chain. The order of the boxes in B-chain can be created randomly or can be based on the volume of the boxes which means ones with larger volume are packed first.

4.2. Two Dimensional Algorithm

Although the algorithm is proposed for the three dimensional knapsack problem, it can also be used to solve two dimensional instances as solving a two-dimensional knapsack problem is simpler than three-dimensional one. The algorithm must be

modified in order to apply to the two-dimensional instances. These modifications are as follows:

Instead of defining three sequences, a pair of sequences commonly known as sequence pair is defined. The definitions are as follows:

- *A-chain*: If rectangle i appears before rectangle j in A-chain, then rectangle i is located left of or on top of rectangle j.
- *B-chain*: If rectangle i appears before rectangle j in B-chain, then rectangle i is located left of or under rectangle j.

Based on these two sequences, rectangle i is located on the left of rectangle j if it appears before box j in both A-chain and B-chain. However, rectangle i is located under rectangle j if it appears before box j in A-chain and after box j in B-chain. These implications are used for the placement algorithm. The first rectangle is placed in the origin, and the succeeding rectangles are placed according to their relative position to the already placed rectangles. The coordinates of each new rectangle are calculated based on the following formula:

$$x_i = \max(0, \max_{j \in P_x}(x_j + w_j)) \qquad y_i = \max(0, \max_{j \in P_y}(y_j + h_j))$$

where P_x and P_y are the subsets of the placed rectangles located on the left and below the new rectangle, respectively. Same simulated annealing scheme is used here but with two-dimensional sequences. The neighborhood of each state is defined as one of these three permutations: either exchange two rectangles in A-chain; exchange two rectangles in B-chain; or exchange two rectangles in both sequences. The rectangles are allowed to be rotated in the following two orientations: WH which is the standard orientation, and HW which is obtained by swapping the width and height. Pre-placed rectangles with known coordinates (x,y) and known dimensions (w, h) are fixed into the bin. Two packing schemes, one for each corner, are created. First, the coordinates of the rectangle are calculated relative to the current origin. Then, their real (x,y) coordinates are calculated relative to the real origin of the bin. Similar to the three-dimensional problems, the order of inserting the rectangles into the bin is based on the order of rectangles in B-chain which can be created randomly or can be based on the area of the rectangles.

CHAPTER 5
Numerical Analysis

5.1. Introduction

This chapter presents some numerical experiments for the proposed solution methodology in order to assess its practicability. The numerical examples are illustrated in section 5.2. Section 5.3 presents the parameter setting for the heuristic algorithm. The results are discussed in section 5.4, and the algorithm verification is illustrated in section 5.5.

5.2. Numerical Experiments

The proposed methodology is implemented in C++. The code is tested using two different sets of boxes. The first set is based on SAE J1100 – Section 9 – Standard which includes 7 types of boxes. The dimensions of these boxes are illustrated in table 5.1. Twelve instances are created by using the first set of boxes. These instances contain 36 and 70 boxes. The maximum allowed number of the boxes for both types of instances is also shown in table 5.1. The second set of the boxes is generated randomly based on Uniform distribution and includes 10 types of boxes. The width, height, and depth of these boxes are selected from the intervals [50, 300], [100, 50], [100, 300] respectively. Two instances are created by using this set of boxes, which includes 50 boxes. The dimensions of the boxes and their maximum allowed number are shown in table 5.2. Thus, fourteen instances are tested in total. In case of not considering pre-placed boxes, the dimensions of the bin for instances containing 36 is equal to 800×700×1000 (mm^3); however, for instances with 70 boxes, it is equal to 1100×900×1400 (mm^3), and in the case of having instances with 50 boxes is equal to 600×500×700 (mm^3). In the case of having obstacles, the bin dimension is equal to 1350×540×890 (mm^3) in instances with 36 boxes, and it is equivalent to 1100×900×1400 (mm^3) in other instances. The profits of the boxes are set to be equal to their volume.

Table 5.1. Information on the First Set of Boxes

Box Type	Width (mm)	Height (mm)	Depth (mm)	Max. no.-instances with 36 boxes	Max. no.-instances with 70 boxes
1	229	483	610	4	7
2	165	330	457	4	7
3	229	406	660	2	5
4	216	457	533	2	5
5	203	229	381	2	5
6	178	356	533	2	6
7	152	114	325	20	35

Table 5.2. Information on the Second Set of Boxes

Box Type	Width (mm)	Height (mm)	Depth (mm)	Max. no.
1	138	182	285	6
2	126	240	135	5
3	108	222	165	4
4	140	80	246	5
5	105	234	272	3
6	153	237	159	6
7	216	229	272	6
8	188	124	236	5
9	137	100	167	4
10	103	104	222	6

The instance names are *Mst-n-o-c-v*, where $n \in \{36, 70, 50\}$ is the number of boxes to be packed, *o* is the order of boxes in B-chain which can be based on the boxes volume (v) or randomly created (R), *c* shows whether or not the obstacles are considered and can be set as (obs) or (wo) respectively, and *v* represents the volume of the bin.

The number and dimensions of the obstacles (pre-placed boxes) differ in various instances. Eight obstacles are defined for cases with 36 and 70 boxes. The dimensions of the obstacles and their coordinates are described in table 5.3. For the instances where there are 70 boxes, four obstacles are defined in case of ceiling obstacles, and two obstacles are defined for middle ones. The dimensions and coordinates of these obstacles are illustrated in table 5.4.

Table 5.3. Obstacles Dimensions and Coordinates for Instances with 36 and 70 Boxes

Obstacle dimensions (mm)	Obstacle coordinates Instance of 36 boxes	Obstacle coordinates Instance of 70 boxes
{180;220;250}	<1170;0;160>	<920;0;160>
{320;220;160}	<0;0;0>	<0;0;0>
{320;220;160}	<1030;0;0>	<780;0;0>
{125;220;160}	<0;0;160>	<0;0;160>
{200;320;320}	<0;220;0>	<0;580;0>
{200;320;320}	<1150;220;0>	<900;580;0>
{160;208;240}	<0;332;320>	<0;692;320>
{160;208;240}	<1190;332;320>	<940;692;320>

Table 5.4. Information on Ceiling and Middle Obstacles

Ceiling Obstacles		Middle Obstacles	
Dimensions (mm)	**Coordinate**	**Dimensions (mm)**	**Coordinate**
{200;320;320}	<0;580;0>	{500;220;160}	<300;300;0>
{200;320;320}	<900;580;0>	{500;220;160}	<300;300;1240>
{160;208;240}	<0;692;320>		
{160;208;240}	<940;692;320>		

5.3. Parameter Setting

As previously mentioned, choosing a suitable cooling procedure and parameters is essential for the algorithm to work efficiently. After testing different cooling procedures (Egeblad and Pisinger, 2009; Pisinger, 2007; Dowsland, 1993) the one proposed by Dowsland (1993) works best. The given cooling process has been explained in section 4.1.3. β is selected to be 0.2, and α is equal to 0.002. Values for initial temperature are selected from {0.5, 0.4, 0.3, 0.2}, and based on the results, t_0=0.2 is the most suitable.

5.4. Results and Sensitivity Analysis

Ten runs were conducted for each case. The worst, best, and average solutions are shown in table 5.5. The values in the table illustrate the bin percentage of the utilization- see section 4.1.3 for formula. In addition, *time* represents the running time for each case in minutes.

Table 5.5. Worst, Best, and Average Utilization

Case	Time (min)	Best (%)	Average (%)	Worst (%)
Mst-36-v-wo-560	10	88.49	86.19	83.92
	20	87.72	85.29	80.45
	30	88.08	86.23	83.43
	120	88.07	85.83	84.81
Mst-36-R-wo-560	10	83.51	80.83	77.31
	20	88.43	85.00	78.26
	30	86.51	83.65	80.19
	120	87.93	87.05	84.81
Mst-36-v-obs-649	10	76.42	74.54	70.76
	20	80.60	78.5	75.63
	30	81.06	79.55	77.64
	120	79.10	77.33	75.13

Table 5.5. (Continued) Worst, Best, and Average Utilization

Case	Time (min)	Best (%)	Average (%)	Worst (%)
Mst-36-R-obs-649	10	82.23	79.15	77.14
	20	82.80	80.03	77.50
	30	80.77	79.22	77.58
	60	80.35	79.24	78.48
	120	80.79	78.88	77.21
Mst-70-v-wo-1386	20	86.34	84.33	82.02
	30	85.99	84.24	82.17
	60	86.29	84.56	82.68
	120	86.44	84.96	82.71
Mst-70-R-wo-1386	20	84.13	80.92	77.27
	30	84.80	83.39	82.49
	60	84.61	81.89	81.64
	120	85.59	83.59	79.57
Mst-70-v-obs-1386	30	79.74	77.24	75.73
	60	82.09	79.14	75.53
	120	80.12	78.93	76.84
Mst-70-R-obs-1386	30	78.12	75.59	75.06
	60	80.24	78.01	76.50
	120	83.66	79.67	78.34
Mst-70-v-obs[1]-1386	30	85.97	84.37	82.88
	60	85.05	83.30	82.06
	120	82.70	81.74	80.18
Mst-70-R-obs[1]-1386	30	82.31	80.68	78.39
	60	82.66	79.75	77.26
	120	83.09	80.09	78.65

[1]Ceiling obstacles

Table 5.5. (Continued) Worst, Best, and Average Utilization

Case	Time (min)	Best (%)	Average (%)	Worst (%)
Mst-70-v-obs[2]-1386	30	79.29	77.66	76.66
	60	78.97	78.46	77.74
	120	79.86	77.80	76.15
Mst-70-R-obs[2]-1386	30	79.74	77.89	76.00
	60	78.96	77.35	76.45
	120	82.50	78.75	76.15
Mst-50[2]-v-wo-210	20	85.49	84.02	82.95
	30	88.58	86.45	84.39
	60	86.56	85.36	83.97
	120	89.68	87.58	85.91
	180	88.31	87.02	85.93
Mst-50[3]-R-wo-210	20	86.79	84.70	82.87
	30	86.41	84.89	83.56
	60	88.07	85.53	84.20
	120	89.72	87.42	85.83
	180	88.06	86.55	85.56

Based on Egeblad & Pisinger (2009), the minimum running time for instances with 36 boxes (*Mst-36-o-c-v*) was set to 10 minutes. Although the heuristic often reached the best solution in less than 10 minutes, the running time was increased to see whether the algorithm is able to jump out of the local optimal and find a better solution. Thus, the instances were run for 20, 30, and 120 minutes as well. Based on the results, increasing time does not significantly affect the solutions. It can be concluded that 10 minutes is sufficient for the heuristic to find the final solution.

For scenarios that contain 70 boxes and where pre-placed boxes are neglected the algorithm was run for at least 20 minutes. The running time was increased to 30, 60, and 120 minutes. The results indicate that 20 minutes is sufficient to reach a good

[2]Boxes with different dimensions

solution in these scenarios. However, when considering obstacles, the algorithm was tested for at least 30 minutes. This is because dealing with the obstacles increases the solution time. The running time was increased to 60 and 120 minutes. The results show that increasing the running time to 60 minutes allows the algorithm to reach better solutions; however, increasing the running time to 120 minutes does not improve the utilization significantly. Therefore, 60 minutes can be a sufficient running time to reach the final solution. In these cases, according to the results, when including ceiling obstacles the reasonable running time is equal to 30 minutes since handling the ceiling obstacles is easier than floor obstacles. In the case of having middle obstacles, the bin utilization is less than other instances. These kinds of instances are run for 30, 60, and 120 minutes. Based on the obtained utilizations shown in table 5.5, 30 minutes can be considered as a reasonable running time. In case of *Mst-70-R-obs(middle)-1386*, the algorithm jumps out of the local minimum after 120 minutes and is able to obtain better solution (higher bin utilization). Nevertheless, only the best utilization enhances, and the average and worst results do not change significantly. Moreover, the instances in which 50 boxes should be packed were run for 20, 30, 60, 120, and 180 minutes; 30 minutes is observed to be enough if it is required to obtain a satisfying solution in a short time. However, it seems that the algorithm is able to jump out of the local optimal and find a better solution after 120 minutes.

The five best solutions for each instance and the number of packed boxes of each type are shown in appendix A. Table 5.6 presents the summary of the results. As it is illustrated in the table, in the most instances the best utilization is obtained when the order of the boxes in B-chain is based on their volume. Appendix B shows the coordinates of the packed boxes at best results.

Table 5.6. Summary of Results (based on the utilization)

Instance	Best (%)	Average (%)	Worst (%)
Mst-36-v-wo-560	88.49	86.19	83.92
Mst-36-R-wo-560	83.51	80.83	77.31
Mst-36-v-obs-649	76.42	74.54	70.76
Mst-36-R-obs-649	82.23	79.15	77.14
Mst-70-v-wo-1386	86.34	84.33	82.02
Mst-70-R-wo-1386	84.13	80.92	77.27
Mst-70-v-obs-1386	82.09	79.14	75.53
Mst-70-R-obs-1386	80.24	78.01	76.50
Mst-70-v-obs-1386 (ceiling)	85.97	84.37	82.88
Mst-70-R-obs-1386 (ceiling)	82.31	80.68	78.39
Mst-70-v-obs-1386 (middle)	79.29	77.66	76.66
Mst-70-R-obs-1386 (middle)	79.74	77.89	76.00
Mst-50-v-wo-210	85.49	84.02	82.95
Mst-50-R-wo-210	86.79	84.70	82.87

The best results for some of the instances are shown in the following figures.

Figure 5.1. Best Result for *Mst-36-v-wo-560*

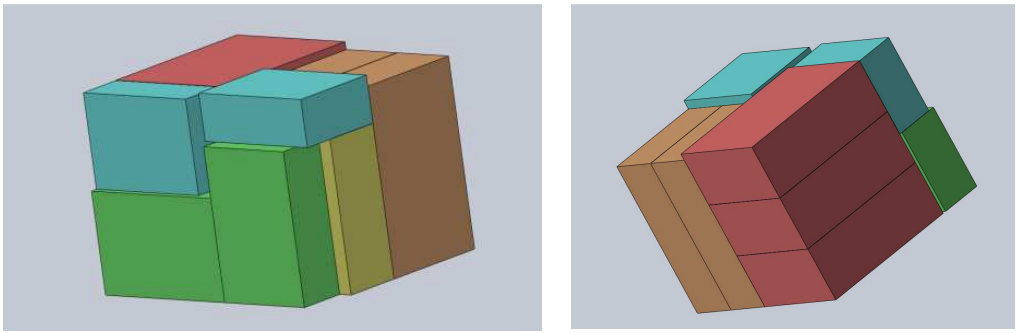


Figure 5.2. Best Result for *Mst-36-R-wo-560*

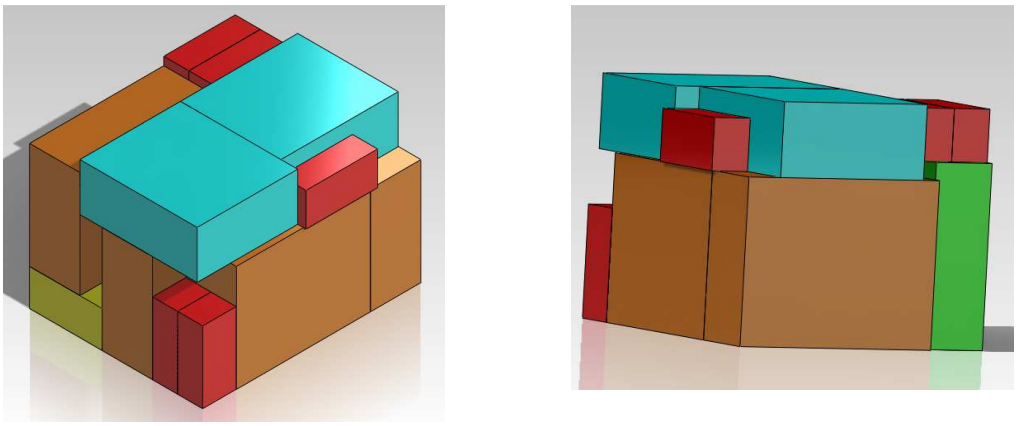


Figure 5.3. Best Result for *Mst-36-R-obs-649*

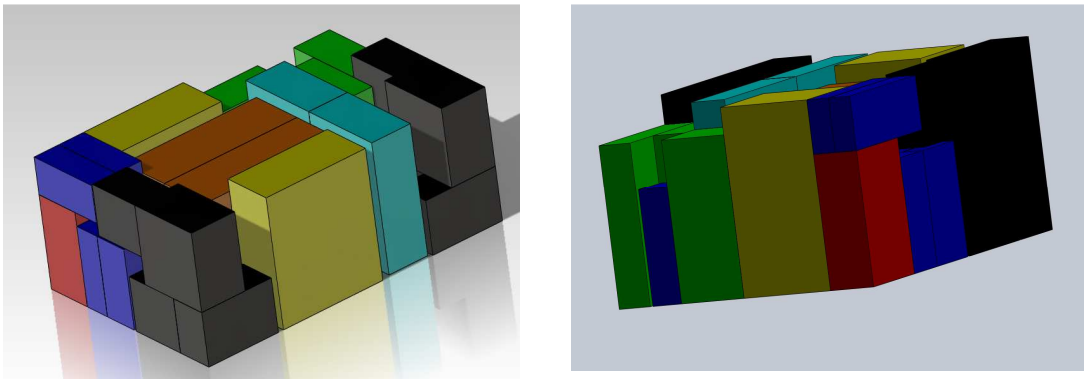


Figure 5.4. Best Result for *Mst-36-v-obs-649*

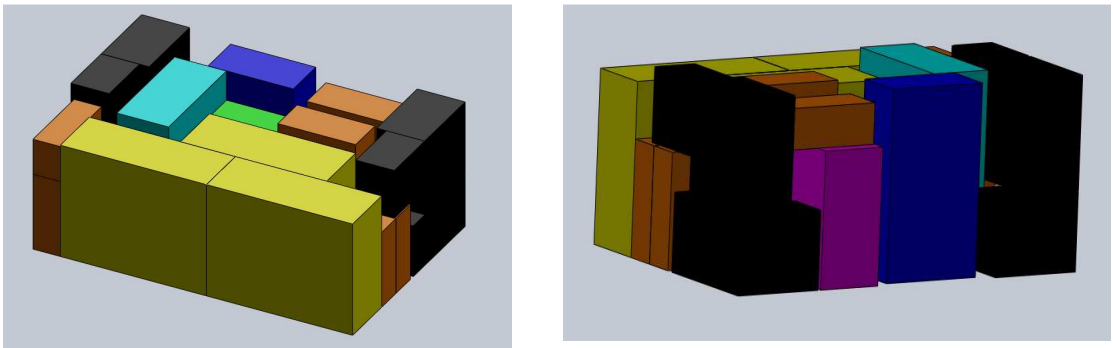


Figure 5.5. Best Result for *Mst-70-v-wo-1386*

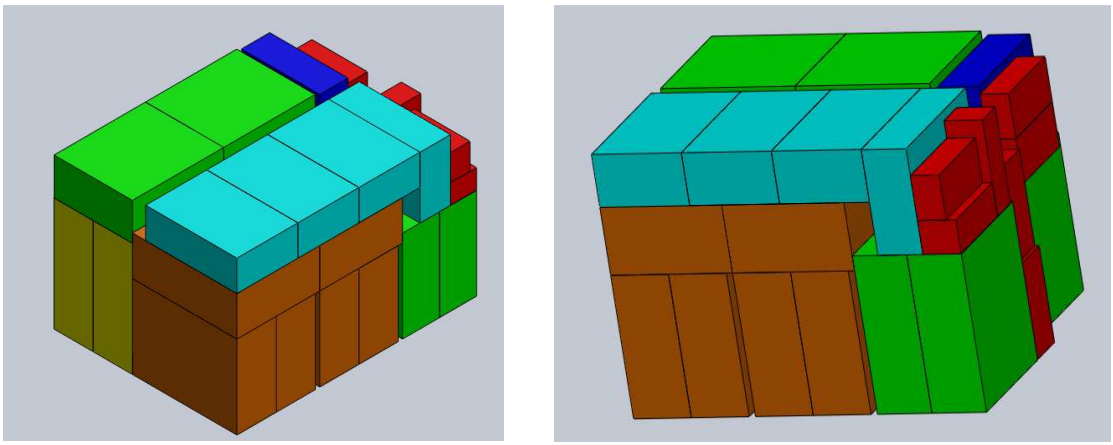


Figure 5.6. Best Result for *Mst-70-R-wo-1386*

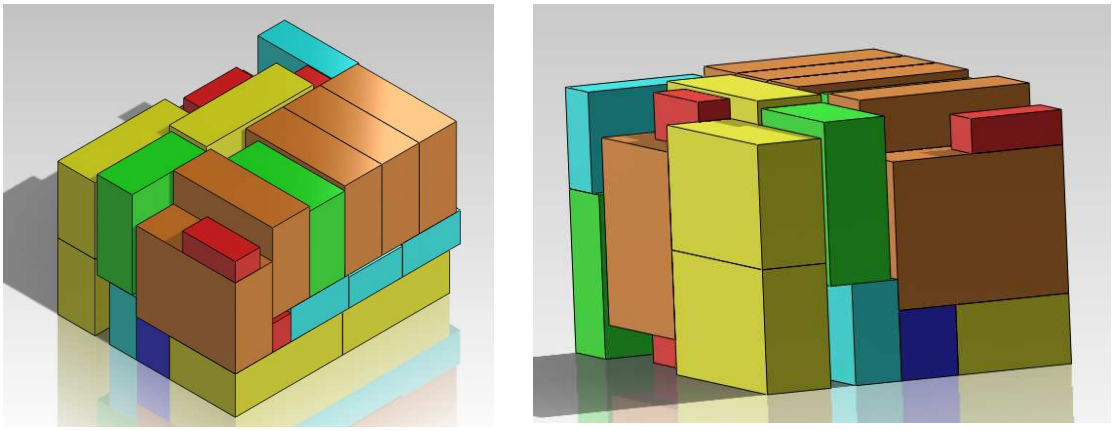


Figure 5.7. Best Result for *Mst-70-R-obs-1386*

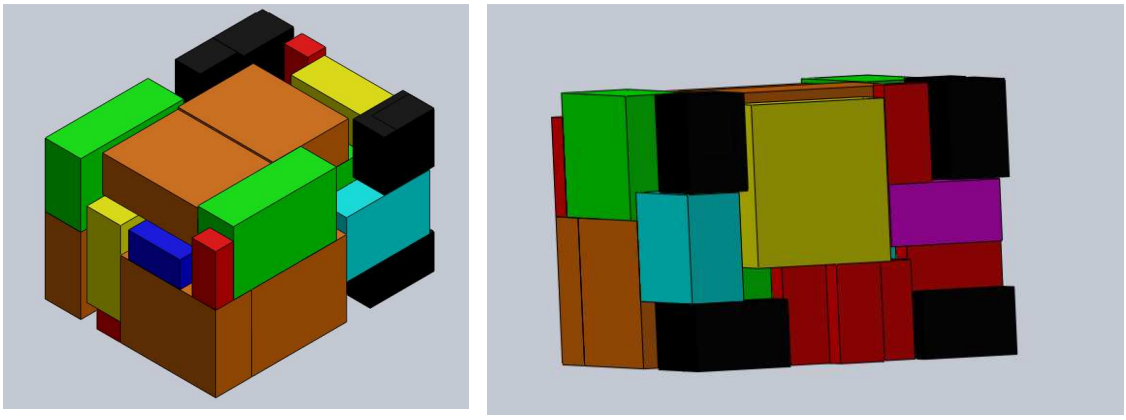


Figure 5.8. Best Result for *Mst-70-v-obs-1386*

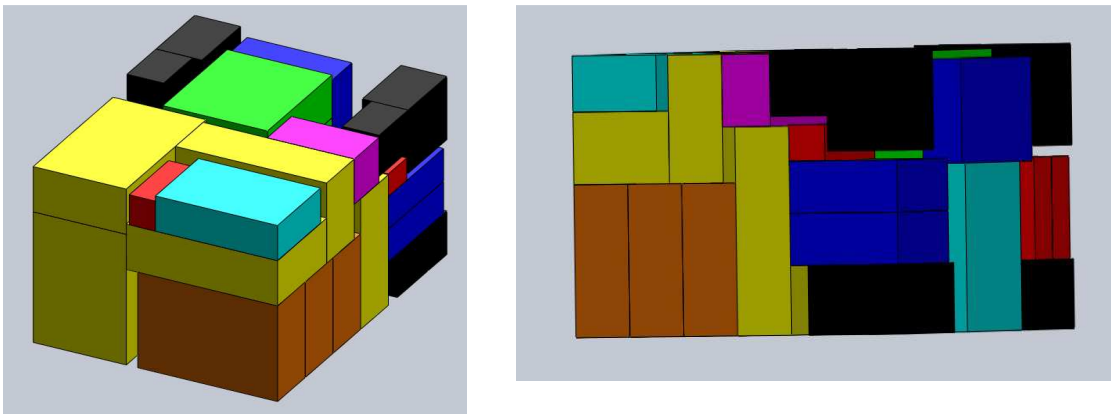


Figure 5.9. Best Result for *Mst-70-v-obs(ceiling)-1386*

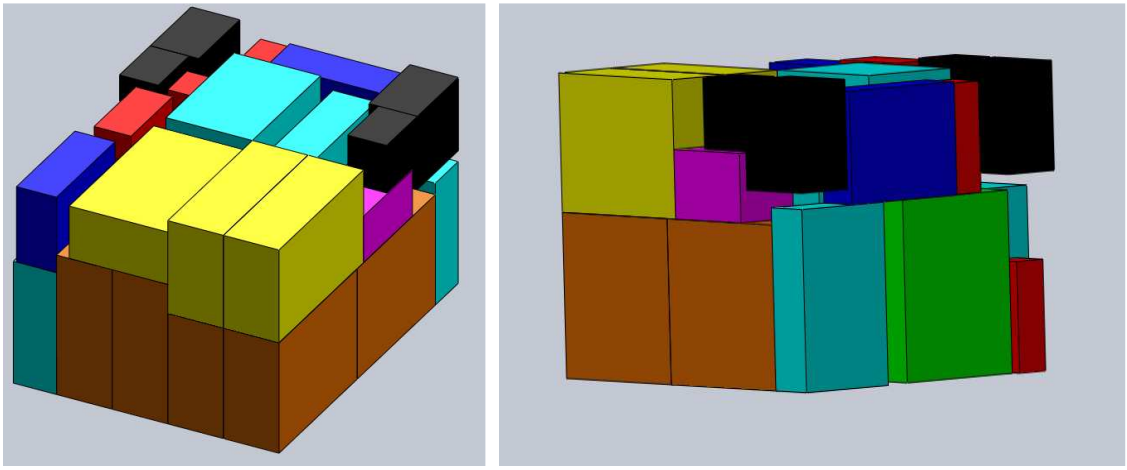


Figure 5.10. Best Result for *Mst-70-R-obs(ceiling)-1386*

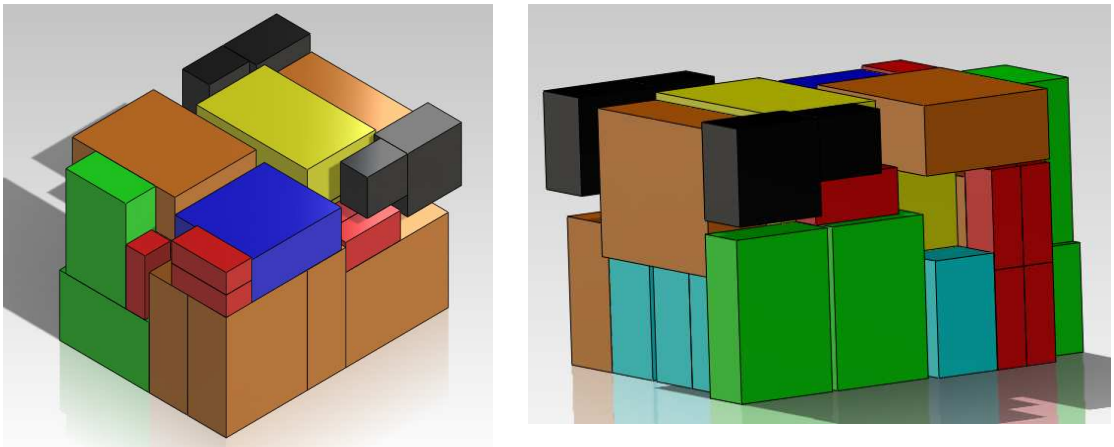


Figure 5.11. Best Result for *Mst-70-v-obs(middle)-1386*

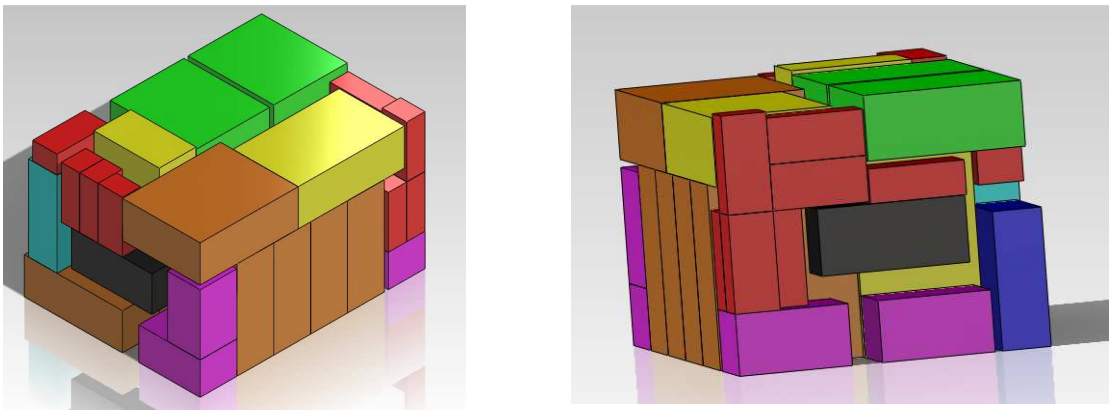


Figure 5.12. Best Result for *Mst-70-R-obs(middle)-1386*

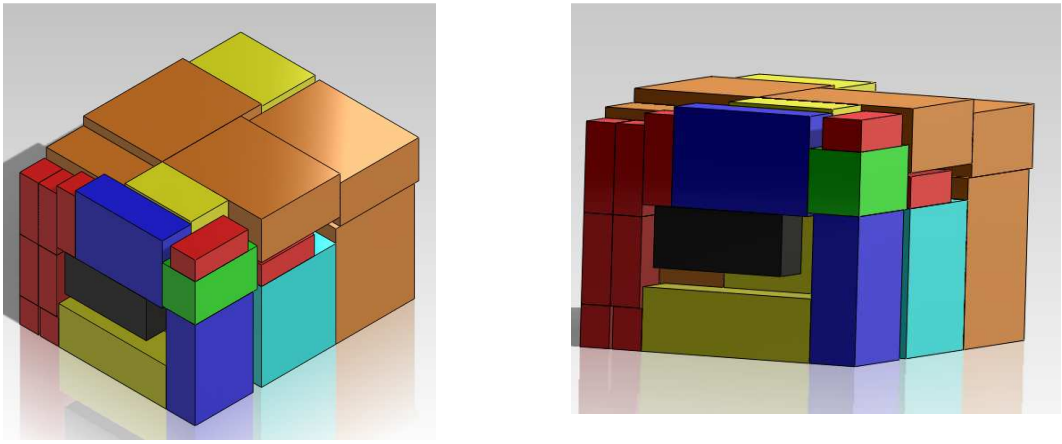


Figure 5.13. Best Result for *Mst-50-v-wo-210*

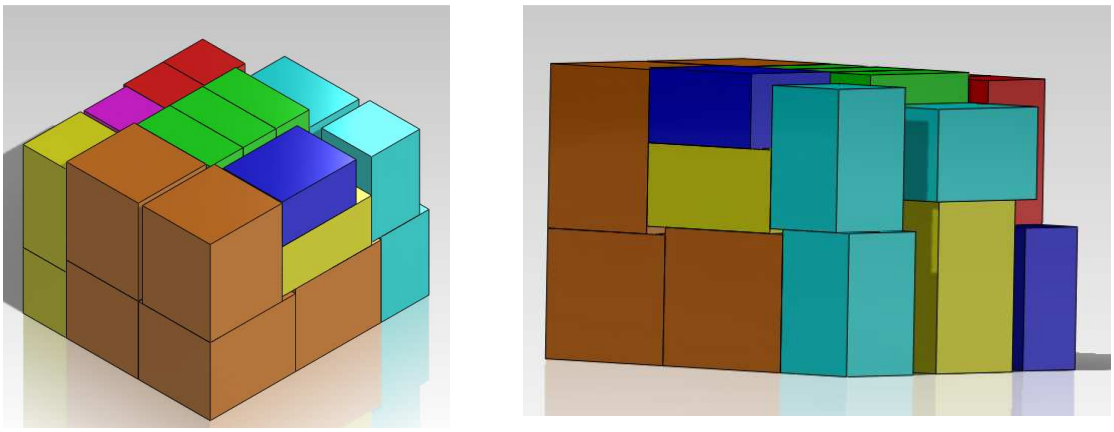
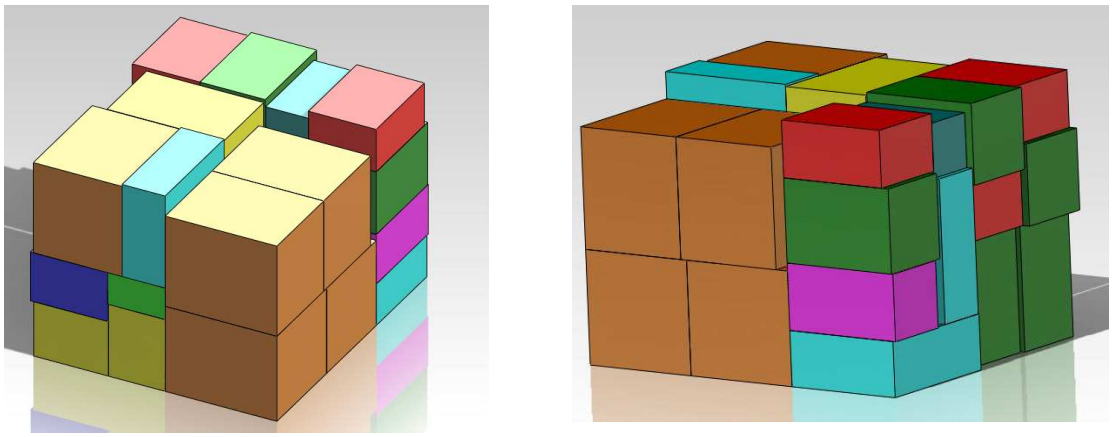


Figure 5.14. Best Result *for Mst-50-R-wo-210*



For the instances with obstacles, pre-placed boxes are shown in black. As shown in figures 5.1-5.14, the vertical stability is satisfied for all instances, and there is no box placed in the air anymore. The bottom of all packed boxes is placed on the bin floor or top of other packed boxes.

5.5. Algorithm Verification

In order to verify the proposed methodology, the *Mst-36-R-obs-649* instance is run without considering vertical stability constraint; the best, worst and average results obtained in this case are equal to 77.38%, 75.19% and 76.2% which are less than the utilizations obtained by considering the vertical stability constraint (82.23%, 77.14% and 79.15%). The result for this case is illustrated in figure 5.15. As shown in the figure some of the boxes are placed in the air.

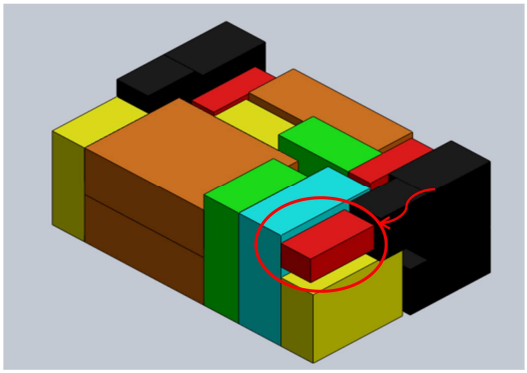


Figure 5.15. Result without Vertical Stability

5.6. Conclusion

Various experiments with different kinds of boxes and obstacles have been executed. Moreover, two different kinds of box insertions have been considered. According to the results, it is evident that the proposed heuristic approach has been successful. Usually the algorithm can achieve the final solution in a very short time. The approach is capable to handle different kinds of instances, and it is not limited to some special instances.

The algorithm is able to deal with different kinds of obstacles such as floor, ceiling and middle obstacles. The position of each packed box should be defined relative to the floor and middle obstacles as well as other packed boxes. Therefore, dealing with such obstacles is more difficult compared to ceiling obstacle. In such instances, the algorithm requires more time to reach the solution. In addition, the results illustrate that the obtained percentage of utilization is decreased in the case of having obstacles in the middle of the bin. Furthermore, the solution time increases for instances created from the second set of boxes as it contains more box types.

The results and the figures in section 5.5 conclude that the vertical stability constraint is satisfied, and there is no box placed in the air. The bottom of all the packed boxes are supported by the bin floor or by the top of other packed boxes. The boxes have been placed into the bin either in a random order or based on their volume. According to the results, in most instances volume-based order leads to better final solutions and higher utilizations. However, by using random order, the results are still satisfying.

At the end, the algorithm has been implemented on one of the instances without considering the vertical stability constraint to verify its success. The results show the proposed approach has been successful.

CHAPTER 6
Conclusions and Future Works

6.1. Conclusions

Packing problems have been extensively studied as they are so essential for operating supply chains and reducing unnecessary cost, such as cost of additional shipment. Packing problems appear under several names and each one has different constraints and objective functions. One of the cutting and packing problems with maximization output is knapsack problem. Multi-dimensional knapsack problem is strongly NP-hard. Some exact algorithms, as well as heuristic approaches, have been considered in the published literature for these problems. As exact algorithms need more time to find a solution, heuristic algorithms are more popular and can be used as an alternative to find optimal or near optimal solution.

A three-dimensional knapsack problem with pre-placed boxes and vertical stability has been presented and discussed. The packing must be orthogonal; boxes are rectangular and can be freely rotated. The mixed integer linear programming model has been proposed for the problem, which considers some practical and real-world constraints such as box rotations, vertical stability, and pre-placed boxes. According to the results obtained from GAMS, optimal solution can only be possible for small instances. Thus, in order to solve the large instances in a reasonable time, a heuristic algorithm has been proposed based on the simulated annealing technique. The methodology is based on the sequence triple representation; moreover, box rotations, vertical stability, and pre-placed boxes are considered in the heuristic approach as well.

Various experiments have been conducted with different sets of boxes. In addition, different cases and multiple kinds of pre-placed boxes have been considered in order to ensure that the solution methodology is able to tackle any kinds of problems, and it is not limited to a special case. The order of box insertion in the bin can be random or based on the box volumes. The found solutions were compared based on the bin utilization. Sensitivity analysis has been done based on the running time in order to find out whether the algorithm can jump out of the local optimal by increasing time and reach a better solution. Although the algorithm was just applied to the three

dimensional knapsack problem it can easily be used for the two dimensional instances since the complexity of these types of instances is less. The algorithm was verified by applying the algorithm not considering the vertical stability to one of the instances.

The results illustrate that the proposed algorithm is successful. Good quality results can be obtained for large instances in a reasonable time. The algorithm is able to handle various instances and get satisfactory utilizations. According to the final results, better solutions can be obtained if the order of inserting boxes in the bin is based on the volume of the boxes. Moreover, the results show that the proposed approach is compatible with pre-placed boxes, and vertical stability is satisfied as well. No box is placed in the air. In addition, the methodology can be used in order to deal with irregular bins- where the bin is not rectangular- by considering the irregular parts as pre-placed boxes.

6.2. Future Works

The proposed mixed integer linear programming model is limited to some practical constraints. The model can be a motivation for future research in a way to extend it to consider more practical and real-world constraints beyond vertical stability, pre-placed boxes, and box rotations. Horizontal stability or loading priorities can be some examples of such constraints. Horizontal stability guarantees that the boxes do not move notably in the middle of transportation. As the number of available bins in knapsack problems is limited, and it should be decided which boxes have to be packed, the loading priorities constraint can play an important role in such problems. The loading of some boxes might be more advantageous than others. These priorities can be consequences of delivery deadlines or freshness desires.

Moreover, the dimensions of the boxes can be considered as non-integer for further research, since in most of real problems boxes do not necessarily have integer dimensions. In addition, non-rectangular and irregular shape boxes can be taken into account in the future. Other heuristic approaches might be studied in the future which are able to tackle more realistic constraints such as weight limits and weight distribution constraints.

REFERENCES/BIBLIOGRAPHY

Amossen, R.R. & Pisinger D., 2010, 'Multi-dimensional Bin Packing Problems with Guillotine Constraints', *Computers & Operations Research*, vol. 37, no. 11, pp. 1999-2006.

Bertsimas, D. & Tsitsiklis, J., 1993, 'Simulated Annealing', *Statistical Science*, vol. 8, no. 1, pp. 10-15.

Bortfeldt, A. & Wascher, G., 2012, 'Container Loading Problems – A State-of-the-Art Review', *FEMM Working Papers from Otto-von-Guericke University Magdeburg, Faculty of Economics and Management*, no. 120007.

Bortfeldt, A. & Winter T., 2009, 'A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces', *International Transactions in Operational Research*, vol. 16, pp. 685-713.

Capara, A. & Monaci, M., 2004, 'On the two-dimensional knapsack problem', *Operation Research Letters*, vol. 32, pp. 5-14.

Clautiaux F., J. Carlier & A. Moukrim, 2007, 'A new exact method for the two dimensional orthogonal packing problem', *European Journal of Operational Research*, vol. 183, pp. 1196-1211.

Dowhan, L., Wymyslowski, A. & Urbanski, K., 2009, 'Simulated annealing as a global optimization algorithm used in numerical prototyping of electrical packaging', *10th International Conference on Thermal, Mechanical and Multi-Physics simulation and Experiments in Microelectronics and Microsystems*, pp. 1-5.

Dolatabadi, M., Lodi, A. & Monaci, M., 2012, 'Exact algorithm for the two-dimensional guillotine knapsack', 2012, *Computer & Operations Research*, vol. 39, pp. 48-53.

Dowsland, K.A., 1993, 'Some experiments with simulated annealing techniques for packing problems', *European Journal of Operational Research*, vol. 68, no. 3, pp. 389-399.

Dyckhoff, H., 1990, 'A typology of cutting and packing problems', *European Journal of Operational Research*, vol. 44, pp. 145-159.

Egeblad, J., Garavelli, C., Lisi, S. & Pisinger, D., 2010, 'Heuristic for container loading of furniture', *European Journal of Operational Research*, vol. 12, pp. 881-892.

Egeblad, J. & Pisinger, D., 2009, 'Heuristic approaches for two- and three-dimensional knapsack packing problem', *Computer & Operations Research*, vol. 36, pp. 1026-1049.

Fekete, S.P. & Schepers, J., 1997, 'On more dimensional Packing i: Modeling', Technical Report, University of Koln, Germany.

Fekete, S.P. & Schepers, J., 1997, 'On more dimensional Packing ii: bounds', Technical Report, University of Koln, Germany.

Fekete, S.P. & Schepers, J., 1997, 'On more dimensional Packing iii: exact algorithm', Technical Report, University of Koln, Germany.

Fekete, S.P. & Schepers J., 2004, 'A General Framework for Bounds for Higher-dimensional Orthogonal Packing Problems', *Mathematical Methods of Operation Research*, vol. 60, no. 2, pp. 311-329.

Goncalves, J.F., 2007, 'A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem', *European Journal of Operation Research*, vol. 183, pp. 1212-1229.

Hifi, M., 2004, 'Exact algorithms for unconstrained three-dimensional cutting problems', *Computers & Operations Research*, vol. 31, no. 5, pp. 657-674.

Joncour, C., Pecher, A. & Valicov, P., 2010, 'MPQ-trees for orthogonal packing problem', *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 423-429.

Junqueira, L., Morabito, R. & Yamashita D.S., 2012, 'Three-dimensional container loading models with cargo stability and load bearing constraints', *Computers & Operations Research*, vol. 39, pp. 74-85.

Leung, T.W., Yung, C.H. & Troutt, M.D., 2001, 'Application of genetic research and simulated annealing to the two-dimensional non-guillotine cutting stock problem', *Computers & Industrial Engineering*, vol. 40, pp. 201-214.

Leung, S.C.H, Zhang, D., Zhou, Ch. & Wu, T., 2012, 'A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem', *Computer & Operations Research*, vol. 39, pp. 64-73.

Martello, S., Pisinger, D., Vigo, D., Den Boef, E. & Korst, J., 2007, 'Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem', *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 1, pp. 1-7.

Murty, K.G., Liu, J. Wan, Y. & Linn, R., 2005, 'A decision support system for operations in container terminal', *Decision support Systems*, vol. 39, no. 3, pp. 309-332.

Murty, K.G., Wan Y., Liu, J., Tseng, M.M., Leung, E., Lai, K. & Chiu, H.W.C., 2005 'Hongkong International Terminals Gains Elastic Capacity Using a Data-Intensive Decision-Support System', *informs*, vol. 35, no. 1, pp. 61-75.

Petering, M.E.H. & Murty, K.G., 2009, 'Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal', *Computers & Operations Research*, vol. 36, no. 5, pp. 1711-1725.

Pisinger, D., 2007, 'Denser Packings Obtained in $O(n \log \log n)$', *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 395-406.

Pisinger, D., 2002, 'Heuristics for the container loading problem', *European Journal of Operational Research*, vol. 141, no. 2, pp. 382-392.

Wascher, G., Haußner, H. & Schumann, H., 2007, 'An improved typology of cutting and packing problems', *European Journal of Operational Research*, vol. 183, pp. 1109-1130.

Wei, L., Zhang, D. & Chen, Q., 2009, 'A least wasted first heuristic algorithm for the rectangular packing problem', *Computers & Operations Research*, vol. 36, pp. 1608-1614.

Wu, Y., Li, W., Goh, M. & de Souza, R., 2010, 'Three Dimensional Bin Packing Problem with Variable Bin Height', *European Journal of Operational Research*, vol. 202, no. 2, pp. 347-355.

APPENDICES

Appendix A

The five best solutions for each instance and the number of packed boxes of each type are shown in the following:

Case	Utilization %	Box Type						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Mst-36-v-wo-560	88.49	3	2	2	1	0	2	0
	87.88	4	4	2	0	0	0	0
	87.49	4	2	1	1	0	1	4
	86.66	4	1	0	2	1	1	6
	85.50	4	1	0	2	1	0	8
Mst-36-R-wo-560	83.51	2	3	0	2	1	1	18
	83.15	3	3	1	1	0	2	8
	82.05	2	2	2	0	0	2	15
	80.98	3	0	1	2	0	0	15
	81.42	1	3	1	2	1	1	17
Mst-36-v-obs-649	70.86	3	1	0	1	1	2	5
	75.13	3	1	0	2	0	1	9
	75.57	2	2	0	2	0	2	11
	76.42	3	0	1	1	1	1	10
	74.70	2	2	0	1	1	1	13
Mst-36-R-obs-649	82.23	2	3	0	2	1	2	10
	80.35	3	1	0	2	1	1	11
	78.82	4	1	0	1	0	1	10
	80.20	1	2	0	3	1	2	8
	76.88	2	2	0	0	2	2	18
Mst-70-v-wo-1386	86.34	6	1	5	5	0	4	11
	85.38	7	2	4	5	1	3	6
	85.92	7	1	5	5	0	3	4
	82.02	7	0	5	4	1	2	11
	82.63	7	0	4	4	1	4	7
Mst-70-R-wo-1386	84.80	7	3	5	2	3	3	10
	83.44	7	3	5	2	5	2	4
	82.49	7	4	5	2	2	3	10
	84.07	7	3	5	3	2	2	9
	83.04	7	3	4	2	0	6	15
Mst-70-v-obs-1386	82.09	3	5	5	5	1	3	8
	81.73	7	2	4	2	4	2	8
	79.74	7	1	4	2	3	2	11
	76.67	5	1	5	2	3	2	17
	80.32	7	4	5	0	2	2	10

case	Utilization %	Box Type						
		1	2	3	4	5	6	7
Mst-70-R-obs-1386	80.24	5	1	5	3	2	3	13
	78.77	6	3	2	2	1	6	16
	78.55	7	4	1	4	3	3	3
	77.60	7	2	3	2	1	2	15
	78.39	7	3	3	1	5	2	13
Mst-70-v/R-obs-1386 (ceiling)	83.52	6	0	4	4	2	5	8
	85.97	6	3	4	3	1	6	7
	84.04	7	2	2	5	1	4	10
	82.06	6	5	4	2	3	3	10
	78.39	6	0	4	3	3	3	13
Mst-70-v/R-obs-1386 (middle)	79.29	6	1	5	2	5	2	13
	77.83	7	3	4	2	2	2	9
	77.64	6	0	4	3	5	2	15
	79.74	6	0	5	1	4	2	19
	77.54	6	3	1	2	1	6	23

case	Utilization %	Box Type									
		1	2	3	4	5	6	7	8	9	10
Mst-50-v-wo-210	85.49	4	4	0	0	3	3	6	3	0	0
	83.95	4	3	1	0	3	3	6	2	0	1
	84.14	4	5	1	1	1	2	6	4	0	0
	84.46	3	0	3	2	1	6	6	3	0	0
	84.50	6	0	2	1	3	1	6	1	1	4
Mst-50-R-wo-210	85.21	0	1	3	4	3	3	6	4	0	5
	86.79	1	5	1	5	3	3	6	3	0	1
	84.86	5	2	1	2	0	3	6	4	0	2
	85.16	1	1	1	5	3	4	6	3	0	4
	83.32	4	2	1	3	3	3	6	1	0	1

Appendix B

-Mst-36-v-wo-560:

Utilization=88.49%

Box type	Box coordinate	Box dimensions
1	<190;0;771>	{610;483;229}
1	<190;0;542>	{610;483;229}
1	<190;0;59>	{610;229;483}
1	<190;229;59>	{610;229;483}
3	<140;458;136>	{660;229;406}
4	<267;483;543>	{533;216;457}
6	<12;0;467>	{178;356;533}
2	<25;356;543>	{165;330;457}
2	<25;0;10>	{165;330;457}
7	<475;458;22>	{325;152;114}
7	<26;356;391>	{114;325;152}
7	<26;356;239>	{114;325;152}
7	<26;356;87>	{114;325;152}
7	<145;458;22>	{325;152;114}

Utilization= 88.40%

Box type	Box coordinate	Box dimensions
1	<0;0;517>	{610;229;483}
1	<0;229;517>	{610;229;483}
1	<0;458;517>	{610;229;483}
3	<0;0;288>	{406;660;229}
3	<0;0;59>	{406;660;229}
4	<406;0;60>	{216;533;457}
6	<622;0;467>	{178;356;533}
6	<622;0;111>	{178;533;356}
2	<610;356;543>	{165;330;457}
2	<406;533;60>	{330;165;457}

- *Mst-36-R-obs-649:*

Utilization= 80.35%

Box type	Box coordinate	Box dimensions
1	<0;0;661>	{610;483;229}
7	<0;0;509>	{114;325;152}
1	<610;0;661>	{610;483;229}
7	<0;0;357>	{114;325;152}
7	<114;0;509>	{114;325;152}
7	<114;0;357>	{114;325;152}
4	<228;0;204>	{216;533;457}
1	<444;0;432>	{610;483;229}
2	<330;0;39>	{330;457;165}
6	<660;0;26>	{356;533;178}
7	<1054;0;547>	{152;325;114}
7	<1220;0;565>	{114;152;325}
4	<444;0;216>	{457;533;216}
7	<1054;0;433>	{152;325;114}
7	<1220;152;565>	{114;152;325}
7	<1206;0;413>	{114;325;152}
7	<1220;304;565>	{114;152;325}
5	<901;0;210>	{229;381;203}
7	<1016;381;88>	{114;152;325}

Utilization= 82.23%

Box type	Box coordinate	Box dimensions
5	<1121;0;687>	{229;381;203}
4	<664;0;674>	{457;533;216}
2	<334;0;725>	{330;457;165}
1	<511;0;445>	{610;483;229}
7	<1198;0;573>	{152;325;114}
1	<511;0;216>	{610;483;229}
6	<333;0;369>	{178;533;356}
7	<181;0;776>	{152;325;114}
4	<550;0;0>	{457;533;216}
2	<168;0;446>	{165;457;330}
2	<3;0;560>	{165;457;330}
7	<1236;0;421>	{114;325;152}
7	<16;0;446>	{152;325;114}
7	<181;0;332>	{152;325;114}
6	<333;0;13>	{178;533;356}
7	<1236;381;565>	{114;152;325}
7	<1122;0;421>	{114;325;152}
7	<1122;381;565>	{114;152;325}
7	<29;0;332>	{152;325;114}
7	<219;0;180>	{114;325;152}

- *Mst-70-v-wo-1386:*

Utilization= 86.34%

Box type	Box coordinate	Box dimensions
1	<490;0;1171>	{610;483;229}
1	<490;0;942>	{610;483;229}
1	<490;483;917>	{610;229;483}
1	<490;0;688>	{610;483;229}
1	<490;483;434>	{610;229;483}
1	<490;0;459>	{610;483;229}
3	<261;0;994>	{229;660;406}
3	<32;0;994>	{229;660;406}
3	<84;0;765>	{406;660;229}
3	<84;0;536>	{406;660;229}
3	<84;0;307>	{406;660;229}
4	<33;660;867>	{457;216;533}
4	<643;0;218>	{457;533;216}
4	<643;0;2>	{457;533;216}
4	<33;660;334>	{457;216;533}
4	<33;0;91>	{457;533;216}
6	<567;712;1044>	{533;178;356}
6	<567;712;688>	{533;178;356}
6	<567;712;332>	{533;178;356}
6	<567;533;154>	{533;356;178}
2	<33;533;142>	{457;330;165}
7	<775;533;2>	{325;114;152}
7	<661;533;2>	{114;325;152}
7	<165;533;28>	{325;152;114}
7	<775;647;40>	{325;152;114}
7	<165;685;28>	{325;152;114}
7	<491;0;345>	{152;325;114}
7	<491;0;231>	{152;325;114}
7	<491;0;117>	{152;325;114}
7	<529;325;134>	{114;152;325}
7	<491;0;3>	{152;325;114}
7	<491;325;20>	{152;325;114}

- *Mst-70-R-wo-1386:*

Utilization= 84.13%

Box type	Box coordinate	Box dimensions
3	<694;0;740>	{406;229;660}
1	<490;229;1171>	{610;483;229}
2	<694;0;80>	{406;229;660}
5	<491;0;1019>	{203;229;381}
2	<325;0;943>	{165;330;457}
5	<491;0;638>	{203;229;381}
5	<465;0;257>	{229;203;381}
2	<249;330;943>	{216;533;457}
1	<236;0;333>	{229;483;610}
7	<775;712;1248>	{325;114;152}
3	<236;483;283>	{229;406;660}
3	<7;0;740>	{229;406;660}
7	<84;0;415>	{152;114;325}
7	<775;229;1057>	{325;152;114}
6	<567;229;701>	{533;178;356}
3	<7;406;740>	{229;406;660}
2	<643;229;371>	{457;165;330}
1	<7;114;257>	{229;610;483}
7	<122;724;415>	{114;152;325}
6	<567;229;15>	{533;178;356}
1	<490;407;942>	{610;483;229}
7	<313;0;219>	{152;325;114}
4	<8;0;3>	{457;533;216}
4	<567;407;726>	{533;457;216}
1	<490;407;497>	{610;483;229}
1	<490;407;268>	{610;483;229}
7	<313;533;169>	{152;325;114}
2	<8;533;4>	{457;330;165}
1	<490;407;39>	{610;483;229}

- *Mst-70-v-obs-1386:*

Utilization= 82.09%

Box type	Box coordinate	Box dimensions
1	<490;0;1171>	{610;483;229}
1	<490;0;942>	{610;483;229}
1	<490;0;713>	{610;483;229}
3	<440;483;994>	{660;229;406}
3	<34;0;1171>	{406;660;229}
3	<34;660;740>	{406;229;660}
3	<440;483;765>	{660;406;229}
3	<694;0;484>	{406;660;229}
4	<478;0;180>	{216;457;533}
4	<224;0;180>	{216;457;533}
4	<237;457;180>	{457;216;533}
4	<237;673;180>	{457;216;533}
4	<224;0;714>	{216;533;457}
6	<338;0;2>	{356;533;178}
6	<46;0;357>	{178;533;356}
6	<567;712;1044>	{533;178;356}
2	<59;0;714>	{165;330;457}
2	<770;220;27>	{330;165;457}
2	<237;533;15>	{457;330;165}
2	<770;385;27>	{330;165;457}
2	<59;330;714>	{165;330;457}
5	<719;660;562>	{381;229;203}
7	<123;533;388>	{114;152;325}
7	<9;533;388>	{114;152;325}
7	<775;550;332>	{325;114;152}
7	<288;533;750>	{152;114;325}
7	<453;712;1075>	{114;152;325}
7	<72;220;160>	{152;325;114}
7	<123;220;8>	{114;325;152}
7	<9;220;8>	{114;325;152}

- *Mst-70-v-obs-1386 (ceiling):*

Utilization= 85.97%

Box type	Box coordinate	Box dimensions
1	<871;0;790>	{229;483;610}
1	<871;0;180>	{229;483;610}
1	<642;0;790>	{229;483;610}
1	<413;0;917>	{229;610;483}
1	<642;0;180>	{229;483;610}
1	<184;0;917>	{229;610;483}
3	<236;610;740>	{406;229;660}
3	<871;483;740>	{229;406;660}
3	<642;483;740>	{229;406;660}
3	<7;0;511>	{229;660;406}
4	<426;0;460>	{216;533;457}
4	<185;0;244>	{457;533;216}
4	<185;0;28>	{457;533;216}
6	<6;0;1044>	{178;533;356}
6	<286;533;207>	{356;178;533}
6	<286;711;207>	{356;178;533}
6	<6;0;155>	{178;533;356}
6	<744;0;2>	{356;533;178}
6	<693;483;207>	{178;356;533}
2	<414;533;15>	{457;330;165}
2	<261;0;587>	{165;457;330}
2	<19;533;943>	{165;330;457}
5	<871;483;359>	{229;203;381}
7	<172;533;359>	{114;325;152}
7	<300;533;28>	{114;325;152}
7	<58;0;3>	{114;325;152}
7	<84;660;592>	{152;114;325}
7	<84;774;592>	{152;114;325}
7	<32;0;930>	{152;325;114}
7	<274;0;473>	{152;325;114}

- *Mst-70-R-obs-1386 (ceiling):*

Utilization= 82.31%

Box type	Box coordinate	Box dimensions
1	<871;0;917>	{229;610;483}
1	<642;0;917>	{229;610;483}
6	<109;0;1222>	{533;356;178}
1	<159;0;993>	{483;610;229}
7	<7;0;1108>	{152;325;114}
1	<617;0;688>	{483;610;229}
7	<7;325;1108>	{152;325;114}
1	<871;0;78>	{229;483;610}
7	<986;483;363>	{114;152;325}
7	<7;0;994>	{152;325;114}
7	<7;325;994>	{152;325;114}
3	<211;0;764>	{406;660;229}
5	<8;0;764>	{203;381;229}
3	<642;0;282>	{229;660;406}
3	<211;0;535>	{406;660;229}
3	<211;0;306>	{406;660;229}
7	<503;356;1248>	{114;325;152}
5	<642;0;79>	{229;381;203}
6	<147;356;1222>	{356;533;178}
1	<7;660;739>	{610;229;483}
5	<414;0;77>	{203;381;229}
5	<211;0;77>	{203;381;229}
1	<261;381;53>	{610;483;229}
3	<211;660;282>	{660;229;406}
6	<33;0;383>	{178;533;356}
7	<872;483;363>	{114;152;325}
7	<97;533;363>	{114;152;325}
7	<775;610;1248>	{325;114;152}
4	<643;610;715>	{457;216;533}
7	<775;724;1248>	{325;114;152}
6	<33;0;7>	{178;533;356}

- *Mst-70-v-obs-1386 (middle):*

Utilization= 79.29%

Box type	Box coordinate	Box dimensions
1	<617;0;942>	{483;610;229}
1	<617;0;713>	{483;610;229}
1	<617;0;484>	{483;610;229}
1	<617;0;255>	{483;610;229}
1	<7;0;917>	{610;229;483}
1	<617;610;790>	{483;229;610}
3	<694;610;130>	{406;229;660}
3	<211;0;688>	{406;660;229}
3	<211;0;459>	{406;660;229}
3	<211;0;230>	{406;660;229}
3	<211;229;942>	{406;660;229}
4	<84;660;485>	{533;216;457}
4	<84;660;2>	{533;216;457}
6	<33;0;332>	{178;533;356}
6	<33;229;1044>	{178;533;356}
2	<46;0;2>	{165;457;330}
5	<719;0;1>	{381;203;229}
5	<236;0;1>	{381;203;229}
5	<719;0;1171>	{381;203;229}
5	<897;203;1171>	{203;381;229}
5	<8;0;688>	{203;381;229}
7	<465;520;1286>	{152;325;114}
7	<351;520;1248>	{114;325;152}
7	<948;203;116>	{152;325;114}
7	<834;203;78>	{114;325;152}
7	<948;203;2>	{152;325;114}
7	<623;528;2>	{325;152;114}
7	<948;528;2>	{152;325;114}
7	<623;680;2>	{325;152;114}
7	<59;762;1075>	{152;114;325}
7	<59;533;160>	{152;114;325}
7	<292;520;8>	{325;114;152}

- *Mst-70-R-obs-1386 (middle):*

Utilization= 82.50%

Box type	Box coordinate	Box dimensions
6	<922;0;1044>	{178;533;356}
3	<262;0;994>	{660;229;406}
5	<897;533;1019>	{203;229;381}
7	<148;0;1075>	{114;152;325}
6	<364;520;1222>	{533;356;178}
7	<250;520;1248>	{114;325;152}
7	<136;152;1248>	{114;325;152}
7	<136;477;1248>	{114;325;152}
7	<22;0;1075>	{114;152;325}
7	<22;152;1248>	{114;325;152}
7	<22;477;1248>	{114;325;152}
3	<491;229;993>	{406;660;229}
4	<884;0;536>	{216;533;457}
3	<655;0;587>	{229;660;406}
1	<8;229;993>	{483;610;229}
1	<871;0;53>	{229;610;483}
1	<172;0;764>	{483;610;229}
7	<330;0;650>	{325;152;114}
7	<330;0;536>	{325;152;114}
2	<325;152;599>	{330;457;165}
4	<414;0;3>	{457;216;533}
2	<414;216;371>	{457;330;165}
2	<414;216;206>	{457;330;165}
2	<7;0;663>	{165;457;330}
7	<948;533;668>	{152;114;325}
7	<948;762;1075>	{152;114;325}
7	<211;0;612>	{114;325;152}
7	<173;325;650>	{152;325;114}
6	<236;0;180>	{178;533;356}
7	<58;457;668>	{114;152;325}
1	<490;660;510>	{610;229;483}
1	<7;0;2>	{229;483;610}
1	<490;610;27>	{610;229;483}
2	<33;546;41>	{457;165;330}
7	<165;546;384>	{325;114;152}
1	<7;660;383>	{483;229;610}
2	<33;711;53>	{457;165;330}

- *Mst-50-v-wo-210:*

Utilization= 85.49%

Box type	Box coordinate	Box dimensions
7	<371;0;428>	{229;216;272}
7	<142;0;428>	{229;216;272}
7	<142;216;484>	{229;272;216}
7	<384;216;471>	{216;272;229}
7	<384;0;156>	{216;229;272}
7	<155;0;156>	{229;216;272}
1	<4;0;415>	{138;182;285}
1	<4;182;518>	{138;285;182}
1	<418;229;186>	{182;138;285}
1	<189;0;18>	{182;285;138}
5	<150;216;379>	{234;272;105}
5	<150;216;274>	{234;272;105}
5	<150;216;169>	{234;272;105}
6	<147;285;10>	{237;153;159}
6	<441;0;3>	{159;237;153}
6	<441;237;33>	{159;237;153}
8	<412;367;235>	{188;124;236}
8	<18;0;227>	{124;236;188}
8	<18;0;39>	{124;236;188}
2	<16;236;383>	{126;240;135}
2	<7;236;257>	{135;240;126}
2	<7;236;131>	{135;240;126}
2	<7;236;5>	{135;240;126}

Mst-50-v-wo-210:

Utilization= 85.49%

Box type	Box coordinate	Box dimensions
8	<364;0;576>	{236;188;124}
4	<354;188;560>	{246;80;140}
2	<228;0;565>	{126;240;135}
6	<363;0;423>	{237;159;153}
8	<40;0;576>	{188;236;124}
7	<384;268;428>	{216;229;272}
10	<378;159;456>	{222;103;104}
7	<82;240;484>	{272;229;216}
5	<82;0;460>	{272;234;105}
7	<138;0;231>	{216;272;229}
7	<138;0;2>	{216;272;229}
8	<14;0;272>	{124;236;188}
2	<12;0;137>	{126;240;135}
2	<360;0;288>	{240;126;135}
1	<462;0;3>	{138;182;285}
2	<12;0;2>	{126;240;135}
3	<354;0;66>	{108;165;222}
2	<360;126;297>	{240;135;126}
4	<354;182;157>	{246;80;140}
7	<82;272;255>	{272;216;229}
5	<82;272;21>	{272;105;234}
5	<82;377;21>	{272;105;234}
4	<2;240;454>	{80;140;246}
7	<371;262;156>	{229;216;272}
6	<363;182;3>	{237;159;153}
4	<2;240;208>	{80;140;246}
6	<363;341;3>	{237;159;153}
4	<2;240;68>	{80;246;140}

Appendix C

(http://www.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-fivese4.html#Q1-60004-22)

In order to prove the NP-hardness of the knapsack problem, it is required to explain some useful definitions:

“

- *Turing Machine:* A Turing machine is a theoretical machine that is used in thought experiments to study the computers borders and capabilities.
- *Boolean expression:* A Boolean expression is an expression which is defined inductively in the following way:
  - ✓ The constants 0 (false) and 1 (true) are Boolean expressions.
  - ✓ Each variable x is a Boolean expression.
  - ✓ If E_1 and E_2 are Boolean expressions, then so are the negation $\neg E_1$, the conjunction $E_1 \wedge E_2$, the disjunction $E_1 \vee E_2$, and the parenthesizing (E_1).

Each assignment of 0's and 1's to the variables of a Boolean expression provides a value to the expression. If E is a Boolean expression, then (E) has the same value as E. ¬E has the value 0 if E has the value 1, and ¬E has the value 1 if E has the value 0. If E_1 and E_2 are Boolean expressions, then $E_1 \vee E_2$ has the value 1 whenever E_1 or E_2 has the value 1. $E_1 \vee E_2$ has the value 0 whenever both E_1 and E_2 have the value 0. The value of $E_1 \wedge E_2$ is 1 if both E_1 and E_2 have the value 1, otherwise $E_1 \wedge E_2$ has the value 0. It is assumed that among the Boolean operations of ¬, ∧, and ∨, the operation ¬ has the highest precedence, followed by ∧, and then ∨.

A Boolean expression is said to be *satisfiable* if its variables can be assigned 0's and 1's so as to provide the value 1 to the expression. The *satisfiability problem* asks for any given Boolean expression whether it is satisfiable, that is, whether the instance is in the set L_{sat} = {E | E is a satisfiable Boolean expression}.

Theorem 1. The satisfiability problem is *NP*-complete.

Proof The satisfiability of any Boolean expression can be checked in polynomial time by nondeterministically assigning some values to the variables of the given expression and then evaluating the expression for such an assignment. Consequently, the problem is in *NP*.

To show that the satisfiability problem is *NP*-hard, it is sufficient to demonstrate that each problem K in *NP* has a polynomially time-bounded, deterministic Turing transducer T_K, such that T_K reduces K to the satisfiability problem. For the purpose of the proof consider any problem K in *NP*. Assume that M = <Q, Σ, Γ, δ, q_0, B, F> is a nondeterministic Turing machine with Q∩(Σ ∪Γ∪{¢, $}) = Ø that decides K in T(n) = $O(n^k)$ time. Let m denote the number of auxiliary work tapes of M; then T_K can be a Turing transducer that on input x outputs a Boolean expression E_x of the following form.

The Structure of E_x: The Boolean expression E_x describes how an accepting computation of M on input x should look. E_x is satisfiable by a given assignment if and only if the assignment corresponds to an accepting computation $C_0 \vdash C_1 \vdash \ldots \vdash C_{T(|x|)}$ of M on input x. The expression has the following structure, where t = T(|x|).

$E_{conf0} \wedge \ldots \wedge E_{conft} \wedge E_{init} \wedge E_{rule1} \wedge \ldots \wedge E_{rulet} \wedge E_{accept} \wedge E_{follow1} \wedge \ldots \wedge E_{followt}$

$E_{conf0} \wedge \ldots \wedge E_{conft}$ states that an accepting computation consists of a sequence $C_0, \ldots, C_t$ of t + 1 configurations. E_{init} states that C_0 is an initial configuration.

$E_{rule1} \wedge \ldots \wedge E_{rulet}$ states that an accepting computation uses a sequence Ψ of t transition rules. E_{accept} states that the last transition rule in Ψ enters an accepting state. With no loss of generality it is assumed that a transition rule can also be "null", that is, a transition rule on which M can have a move without a change in its configuration. Such an assumption allows us to restrict the consideration only to computations that consist of exactly T(|x|) moves.

$E_{followi}$ states that M by using the *i*th transition rule in Ψ reaches configuration C_i from configuration C_{i-1}, $1 \leq i \leq t$.

The Variables of E_x: The Boolean expression E_x uses variables of the form $w_{i,r,j,X}$ and variables of the form $w_{i,\tau}$. Each variable provides a statement about a possible

property of an accepting computation. An assignment that satisfies E_x provides the value 1 to those variables whose statements hold for the computation in question, and provides the value 0 to those variables whose statements do not hold for that computation.

$w_{i,r,j,X}$ states that X is the *j*th character of the *r*th tape in the *i*th configuration $0 \leq r \leq m$. $r = 0$ refers to the input tape, and $1 \leq r \leq m$ refers to the *r*th auxiliary work tape. $w_{i,\tau}$ states that τ is the transition rule in the *i*th move of the computation.

The Structure of E_{confi} : The expression E_{confi} is the conjunction of the following Boolean expressions.

a. $\vee\{ w_{i,0,j,X} \mid$ X is in $\Sigma \cup \{¢, \$\} \cup Q$ for $1 \leq j \leq |x| + 3$.

   This expression states that a configuration has an input segment with $|x| + 3$ entries, with each entry having at least one symbol from $\Sigma \cup \{¢, \$\} \cup Q$.

b. $\wedge\{ \neg(w_{i,0,j,X} \wedge w_{i,0,j,Y}) \mid$ X and Y are in $\Sigma \cup \{¢, \$\} \cup Q$ and $X \neq Y \}$ for $1 \leq j \leq |x| + 3$.

   This expression states that each entry in the input segment has at most one symbol.

c. $\vee\{ w_{i,r,j,X} \mid$ X is in $\Gamma \cup Q \}$ for $1 \leq r \leq m$ and $1 \leq j \leq t + 1$.

   This expression states that a configuration has m auxiliary work-tape segments, each segment having $t + 1$ entries, and each entry having at least one symbol from $\Gamma \cup Q$.

d. $\wedge\{ \neg(w_{i,r,j,X} \wedge w_{i,r,j,Y}) \mid$ X and Y are in $\Gamma \cup Q$ and $X \neq Y \}$ for $1 \leq r \leq m$ and $1 \leq j \leq t + 1$.

   This expression states that each entry in an auxiliary work-tape segment has at most one symbol.

Each assignment that satisfies the expressions in parts (a) and (b) above implies a string of length $|x| + 3$. The string corresponds to the input tape of M, and consists of

input symbols, end-marker symbols ¢ and $, and state symbols. In particular, the symbol X is at location j in the string if and only if $w_{i,0,j,X}$ is assigned the value 1.

Similarly, each assignment that satisfies the expressions in parts (c) and (d) above for a specific value r, provides a string of length t + 1 that corresponds to the *r*th auxiliary work tape of M. The string consists of auxiliary work tape symbols and state symbols. In particular, the string consists of the symbol X at location j if and only if $w_{i,r,j,X}$ is assigned the value 1.

The Structure of E_{init}: The expression E_{init} is the conjunction of the following three Boolean expressions.

a. $w_{0,0,1,q0} \wedge w_{0,0,2,q0} \wedge \{ w_{0,0,j+2,aj} \mid 1 \leq j \leq |x| \} \wedge w_{0,0,|x|+3,\$}$.

   This expression states that in the initial configuration the input segment consists of the string $¢q_0a_1 \ldots a_n\$$, where a_j denotes the *j*th input symbol in x.

b. $\vee \{ w_{0,r,j,q0} \mid 1 \leq j \leq t + 1 \}$ for $1 \leq r \leq m$.

   This expression states that in the initial configuration each auxiliary work-tape segment contains the initial state q_0.

c. $w_{0,r,j,B} \vee w_{0,r,j,q0} \wedge \{ w_{0,r,s,B} \mid 1 \leq s \leq t+1 \text{ and } s \neq j \}$ for $1 \leq j \leq t+1$ and $1 \leq r \leq m$.

   This expression states that in the initial configuration each auxiliary work-tape segment consists of blank symbols B and at most one appearance of q_0.

Each assignment that satisfies E_{init} corresponds to an initial configuration of M on input x. Moreover, each also satisfies E_{conf0}.

The Structure of E_{rulei} and E_{accept} : The expression E_{rulei} is the conjunction of the following two Boolean expressions.

a. $\bigvee \{ w_{i,\tau} \mid \tau \text{ is in } \delta \}$
b. $\bigwedge \{ \neg(w_{i,\tau 1} \wedge w_{i,\tau 2}) \mid \tau_1, \tau_2 \text{ are in } \delta \text{ and } \tau_1 \neq \tau_2 \}$.

The expression in part (a) implies, that for each assignment that satisfies E_{rulei}, at least one of the variables $w_{i,\tau}$ has the value 1. The expression in part (b) implies, that for

each assignment that satisfies E_{rulei}, at most one of the variables $w_{i,\tau}$ has a value 1. Hence, each assignment that satisfies E_{rulei} assigns the value 1 to exactly one of the variables $w_{i,\tau}$, namely, to the variable that corresponds to the transition rule τ used in the *i*th move of the computation in question.

The expression E_{accept} is of the form $\vee\{ w_{t,\tau} \mid \tau$ takes M into an accepting state $\}$.

The Structure of $E_{followi}$: The expression $E_{followi}$ is the conjunction of the following Boolean expressions.

a. $\vee \{ (w_{i,0,j,X} \wedge w_{i-1,0,j-1,Y} \wedge w_{i-1,0,j,Z} \wedge w_{i-1,0,j+1,W} \wedge w_{i,\tau}) \mid$ X, Y, Z, W, and τ such that $X = f_0(Y, Z, W, \tau) \}$ for $1 \leq j \leq |x| + 3$.
b. $\vee\{ (w_{i,r,j,X} \wedge w_{i-1,r,j-1,Y} \wedge w_{i-1,r,j,Z} \wedge w_{i-1,r,j+1,W} \wedge w_{i,\tau}) \mid$ X, Y, Z, W, and τ such that $X = f_r(Y, Z, W, \tau) \}$ for $1 \leq r \leq m$ and $1 \leq j \leq t + 1$.

Where, $\mathbf{f_r(Y, Z, W, \tau)}$ is a function that determines the replacement X for a symbol Z in a configuration, resulting from the application of the transition rule τ.

Z is assumed to be enclosed between Y on its left and W on its right.

$\mathbf{w_{i-1,0,0,Y}, \ldots, w_{i-1,m,0,Y}, w_{i-1,0,|x|+4,W}, w_{i-1,1,t+2,W}, \ldots, w_{i-1,m,t+2,W}}$ are new variables. They are introduced to handle the boundary cases in which the symbol Z in $f_r(Y, Z, W, \tau)$ corresponds to an extreme (i.e., leftmost or rightmost) symbol for a tape.

If $\tau = (q, a, b_1, \ldots, b_m, p, d_0, c_1, d_1, \ldots, c_m, d_m)$, then the value X of the function $f_r(Y, Z, W, \tau)$ satisfies X = p whenever one of the following cases holds.

a. $Z = q$ and $d_r = 0$.
b. $Y = q$ and $d_r = +1$.
c. $W = q$ and $d_r = -1$.

Similarly, $X = c_r$ whenever one of the following cases holds, $1 \leq r \leq m$.

a. $Z = q$, $W = b_r$, and $d_r = +1$.
b. $Y = q$, $Z = b_r$, and $d_r = 0$.
c. $Y = q$, $Z = b_r$, and $d_r = -1$.

On the other hand,

a. X = W whenever Z = q, r = 0, and $d_0 = +1$.
b. X = Y whenever Z = q and $d_r = -1$.

In all the other cases X = Z because the head of the *r*th tape is "too far" from Z.

The result now follows because T_K on input x can compute t = T(|x|) in polynomial time and then output (the string that represents) E_x.

- *The 3-Satisfiability Problem:* A slight modification to the previous proof implies the *NP*-completeness of the following restricted version of the satisfiability problem.

Definitions A Boolean expression is said to be a *literal* if it is a variable or a negation of a variable. A Boolean expression is said to be a *clause* if it is a disjunction of literals. A Boolean expression is said to be in *conjunctive normal form* if it is a conjunction of clauses. A Boolean expression is said to be in *k-conjunctive normal form* if it is in conjunctive normal form and each of its clauses consists of exactly k literals. The *k-satisfiability problem* asks for any given Boolean expression in k-conjunctive normal form whether the expression is satisfiable.

With no loss of generality, in what follows it is assumed that no variable can appear more than once in any given clause.

Theorem 2. The 3-satisfiability problem is *NP*-complete.

Proof The expression E_x in the proof of Theorem 1 needs only slight modifications to have a 3-conjunctive normal form.

a. Except for the expressions $E_{followi}$ and part (c) of E_{init}, all the other expressions can be modified to be in conjunctive normal form by using the equivalence $\neg(w_1 \wedge w_2) \rightleftharpoons(\neg w_1) \vee (\neg w_2)$.
b. Each expression in $E_{followi}$ and part (c) of E_{init} can be modified to be in conjunctive normal form by using the equivalence $w_1 \vee (w_2 \wedge w_3) \equiv(w_1 \vee w_2) \wedge (w_1 \vee w_3)$.
c. Each disjunction $w_1 \vee ... \vee w_s$ with s > 3 clauses can be modified to be in 3-conjunctive normal form by repeatedly replacing sub-expressions of the form

$w_1 \vee ... \vee w_s$ with sub-expressions of the form $(w_1 \vee w_2 \vee w) \wedge (\neg w \vee w_3 \vee ... \vee w_s)$, where the w's are new variables.

- ❖ The proof of the Knapsack problem NP-hardness is as follows:

Consider a Turing machine M that on any instance $(a_1, \ldots, a_N, b)$ of the problem assigns value from {0,1} to $v_1, \ldots, v_N$ non-deterministically. Accept the input if and only if $a_1v_1+\ldots+a_nv_n=b$. Therefore the 0-1 knapsack problem is in NP.

In order to show that 0-1 knapsack problem is NP-hard, consider any given instance E of the 3-satisfiability problem. Let $x_1, \ldots, x_n$ indicate the variables in Boolean expression E. is a conjunction $c_1 \wedge ... \wedge c_k$ of some clauses $c_1, \ldots, c_k$. Each C_i is a disjunction $c_{i\,1} \vee c_{i\,2} \vee c_{i\,3}$ of some literals $c_{i\,1}, c_{i\,2}, c_{i\,3}$. Each $c_{i\,j}$ is a variable x_t, or a negation $\neg x_t$ of a variable x_t, for some $1 \leq t \leq m$.

The following system S of linear equations is developed from Boolean expression E:

$$\begin{aligned} x_1 + \bar{x}_1 &= 1 \\ &\vdots \\ x_m + \bar{x}_m &= 1 \\ c_{1\,1} + c_{1\,2} + c_{1\,3} + y_{1\,1} + y_{1\,2} &= 3 \\ &\vdots \\ c_{k\,1} + c_{k\,2} + c_{k\,3} + y_{k\,1} + y_{k\,2} &= 3 \end{aligned}$$

The variable x_t in system S corresponds to the literal x_t in E. The variable $\bar{x}_t$ in S corresponds to the literal $\bar{x}_t$ in E. $c_{i\,j}$ stands for the variable x_t in S, if x_t is the *j*th literal in C_i. $c_{i\,j}$ stands for the variable $\bar{x}_t$ in S, if $\neg x_t$ is the *j*th literal in C_i.

Each equation of the form $x_i + \bar{x}_i = 1$ has a solution over {0, 1} $\Leftrightarrow$ either $x_i = 1$ and $\bar{x}_i = 0$, OR $x_i = 0$ and $\bar{x}_i = 1$.

Each equation of the form $c_{i\,1} + c_{i\,2} + c_{i\,3} + y_{i\,1} + y_{i\,2} = 3$ has a solution over {0, 1} $\Leftrightarrow$ at least one of $c_{i\,1} = 1$, $c_{i\,2} = 1$, and $c_{i\,3} = 1$ is satisfied.

Therefore, system S has a solution over {0, 1} $\Leftrightarrow$ the Boolean expression E is satisfiable.

The vector form of system S is shown in the following:

$$\begin{pmatrix} a_{1\,1} \\ \vdots \\ a_{m+k\,1} \end{pmatrix} z_1 + \cdots + \begin{pmatrix} a_{1\,2m+2k} \\ \vdots \\ a_{m+k\,2m+2k} \end{pmatrix} z_{2m+2k} = \begin{pmatrix} b_1 \\ \vdots \\ b_{m+k} \end{pmatrix}$$

The variables $z_1, \ldots, z_{2m+2k}$ stand for the variables $x_1, \ldots, x_m$, $x_1, \ldots, x_m$ and $y_{11}, \ldots, y_{k2}$ respectively. A_{ij} is the coefficient z_j in the *i*th equation of S. b_i is the constant in the right-hand side of the *i*th equation in S.

System S can be shown by the equation H:

$$a_1 z_1 + \cdots + a_{2m+2k} z_{2m+2k} = b \qquad \text{(H)}$$

Each a_j for the integer whose decimal representation is $a_{1\,j}, \ldots, a_{m+k\,j}$. In addition, b stands for the integer whose decimal representation is $b_1, \ldots, b_{m+k}$. The representation is possible because the sum $a_{i\,1} + \ldots + a_{i\,2m+2k}$ is either equal to 2 or to 5 for each $1 \leq i \leq m+k$. Which means that the *i*th digit in the sum $c = a_1 + \ldots + a_{2m+2k}$ depends only on the *i*th digits of $a_1, \ldots, a_{2m+2k}$. Thus, S is satisfiable over {0, 1} if and only if H is satisfiable over {0, 1}.

Therefore, instance E of the 3-satisfiability problem is satisfiable ⇔ instance $(a_1, \ldots, a_{2m+2k}, b)$ of the 0 - 1 knapsack problem has a positive solution.

Furthermore, a polynomially time-bounded, deterministic Turing transducer can similarly construct corresponding instance of the 0 - 1 knapsack problem, from each instance E of the 3-satisfiability problem. As a result, the *NP*-hardness of the 0 - 1 knapsack problem follows from the *NP*-hardness of the 3-satisfiability problem." (http://www.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-fivese4.html#Q1-60004-22)

VITA AUCTORIS

NAME:	Hanan Mostaghimi Ghomi
PLACE OF BIRTH:	Tehran, Iran
YEAR OF BIRTH:	1988
EDUCATION:	Zahra High School, Tehran, Iran, 2006
	University of Tehran, B.Sc., Tehran, Iran, 2010
	University of Windsor, M.Sc. Candidate, Windsor, ON, 2013